

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Self-Immunity in Industrial Cyber-Physical Systems

Rui Pedro Ferreira Pinto

THESIS SUBMITTED TO THE UNIVERSITY OF PORTO FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY IN INFORMATICS ENGINEERING

Doctoral Program in Informatics Engineering

Supervisor: Gil Manuel Magalhães de Andrade Gonçalves

July 2022

Self-Immunity in Industrial Cyber-Physical Systems

Rui Pedro Ferreira Pinto

Doctoral Program in Informatics Engineering

Approved by the Doctoral Committee:

Chair: Dr. Carlos Miguel Ferraz Baquero-Moreno, Full Professor

External Examiner: Dr. Jerker Delsing, Full Professor

External Examiner: Dr. Vítor Manuel de Jesus Filipe, Associate Professor

Internal Examiner: Dr. Hugo José Sereno Lopes Ferreira, Assistant Professor

Supervisor: Dr. Gil Manuel Magalhães de Andrade Gonçalves, Assistant Professor



July 2022

Assinado por: **Gil Manuel Magalhães de Andrade
Gonçalves**
Num. de Identificação: 08710109
Data: 2022.07.28 08:04:04 +0100

Abstract

The premise of *Smart Manufacturing Systems* (SMSs) is the increasing interaction between machines, devices, sensors, and people, by merging typical *Information and Communication Technology* (ICT) and *Operational Technology* (OT), such as *Cyber-Physical Production Systems* (CPPSs). In such complex systems, where autonomic capabilities are highly desired, openness in information exchange can harm privacy and security. OT has distinct security priorities and a different infrastructure to defend when compared to ICT. First, because of their importance in monitoring and regulating critical industrial processes, OT systems may require enhanced security features. Second, most businesses rely largely on their OT providers, limiting their ability to adapt and implement security solutions. Finally, most OT components are insecure by design and vulnerable to attacks, because they are frequently created without basic ICT security requirements in mind.

Thus, ICT approaches are typically altered to comply with the OT context. The *Biological Immune System* (BIS) is a highly autonomous and decentralized system, extremely adaptable, self-organizing in nature, composed of multiple interacting immune agents. The BIS has a memory of previous contacts with *Pathogens*, and can constantly learn about new invaders, which means it has a lot to offer, in terms of computational inspiration, to modern SMSs. This field of research is referred to as *Artificial Immune Systems* (AISs). At the core of AISs research, there are several immune-inspired models, by considering the BIS ability to discriminate between the body's cells and foreign cells and recognizing *Pathogens* based on the analysis of the environmental context. One of the most important immune models nowadays is the *Dendritic Cell Algorithm* (DCA), which captures the functionality of the biologic *Dendritic Cells*.

The DCA exhibits several interesting features for Anomaly Intrusion Detection System (A-IDS), showing good detection performance and presenting reduced false-positive errors. However, the DCA also presents weaknesses that limit its application as an A-IDS for CPPSs. As a result, this Thesis aims to analyze the DCA's algorithmic aspects and address some of its limitations when applied in binary classification problems. This objective leads to the research question related to the suitability of using the DCA for anomaly intrusion detection in CPPSs.

To answer this question, a theoretical framework of the research study is developed by surveying different AISs models and their application in SMSs. Then, a requirement analysis and solution specification are provided, considering the research strategies related to DCA-based IDS. With this in mind, a hybridization of the DCA with incremental learning is proposed, leading to the so-called *Incremental Dendritic Cell Algorithm* (iDCA). The iDCA presents an improved detection speed while maintaining a reasonable detection accuracy compared to the original DCA version. Finally, an extension to the IEC 61499 standard is proposed, enabling a security-by-design development method for building CPPSs, by automatically embedding the iDCA for intrusion detection in CPPS.

This study's findings have practical implications on the *Industry 4.0* related community by making novel contributions to the intrusion detection problem in CPPSs considering immune-inspired solutions. According to the experimental data, enabling the iDCA by design when build-

ing a CPPS using the IEC 61499 standard can dramatically reduce design and code *Complexity* while improving application *Maintainability*.

Keywords: *Artificial Immune System • Embedded and Cyber-Physical Systems • IEC 61499 • Smart Manufacturing System • Intrusion/Anomaly Detection • Online Learning*

Resumo

A premissa dos *Sistemas de Produção Inteligentes* (SMSs) é a crescente interação entre ativos do chão de fábrica, considerando a fusão de *Tecnologias da Informação e Comunicação* (TIC) e *Tecnologia Operacional* (OT), como *Sistemas de Produção Ciber-Físicos* (CPPSs). Nestes sistemas complexos, onde autonomia é altamente desejada, a abertura nas comunicações tem impactos negativos na privacidade e segurança. Quando comparado com TIC, OT tem prioridades de segurança distintas e uma infraestrutura diferente para defender. Em primeira instância, devido à sua importância na monitorização e regulação de processos industriais críticos, os sistemas OT podem exigir recursos de segurança aprimorados. Em segundo lugar, a maioria das empresas está demasiado dependente dos seus fornecedores de OT, o que limita a capacidade de adotar e implementar soluções de segurança. Por último, a maioria dos componentes OT são inseguros por defeito e vulneráveis a ataques cibernéticos, sendo frequentemente desenvolvidos sem os requisitos básicos de segurança em mente.

Desta forma, as abordagens TIC são tipicamente modificadas para se adequarem a contextos OT. O *Sistema Imune Biológico* (BIS) é um sistema altamente autónomo e descentralizado, extremamente adaptável, auto-organizado por natureza e composto por múltiplos agentes imunológicos em constante interação. O BIS memoriza *Patógenos* encontrados anteriormente, e aprende constantemente sobre novos invasores, o que significa que tem muito a oferecer, em termos de inspiração computacional, aos modernos SMSs. Esta área de investigação é conhecida como *Sistemas Imunes Artificiais* (AIS). Existem vários modelos de inspiração imunológica, considerando a capacidade do BIS em diferenciar entre células estranhas de células do próprio corpo, bem como reconhecer *Patógenos* com base na análise do contexto imune. Um dos modelos imunológicos mais importantes atualmente é o *Dendritic Cell Algorithm* (DCA), inspirado na funcionalidade das *Células Dendríticas* biológicas.

O DCA exhibe várias características interessantes para um *Sistema de Detecção de Intrusos* (IDS), desde um bom desempenho de deteção, até à redução de falsos positivos. Contudo, o DCA apresenta também fragilidades que limitam a sua aplicação como um IDS para CPPSs. Como resultado, esta tese tem como objetivo analisar os aspetos algorítmicos do DCA e abordar algumas das suas limitações quando aplicado em problemas de classificação binária. Isso leva à questão de investigação que está relacionada à adequação do uso do DCA para deteção de intrusos em CPPSs.

Para responder a esta questão, foi desenvolvida uma framework teórica para este estudo, através do levantamento de diferentes modelos AIS e a sua aplicação em SMS. Em seguida, é fornecida uma análise de requisitos e especificação da solução, considerando as estratégias de investigação relacionadas com IDS baseados em DCA. Com isso em mente, é proposta uma hibridação do DCA com aprendizagem incremental, resultando no *Incremental Dendritic Cell Algorithm* (iDCA). O iDCA apresenta uma velocidade de deteção aprimorada, mantendo uma performance de deteção satisfatória quando comparado com a versão original do DCA. Finalmente, é proposta uma extensão para o standard IEC 61499, que permite desenvolver CPPSs seguros por defeito, ao

incorporar automaticamente o iDCA para detecção de intrusos.

As descobertas deste estudo têm implicações úteis na comunidade emergente de *Indústria 4.0*, ao realizar novas contribuições para o problema de detecção de intrusos em CPPSs considerando soluções inspiradas pelo BIS. De acordo com os dados experimentais, incorporar por defeito o iDCA durante a implementação de CPPSs, usando o standard IEC 61499, pode reduzir drasticamente a complexidade no desenho e desenvolvimento de código enquanto melhora a capacidade de manutenção do sistema criado.

Palavras-chave: *Sistema Imune Artificial • Sistemas Embebidos e Ciber-Físicos • IEC 61499 • Sistema de Produção Inteligente • Detecção de Intrusos/Anomalias • Online Learning*

Acknowledgments

I want to thank my supervisor *Gil Gonçalves* for introducing me to Smart Manufacturing-related research areas (even before starting this Ph.D.). I thank him for his continuous guidance and all the valuable discussions on the theoretical investigation of this Thesis. A word of appreciation to many of my colleagues and friends from *DIGI2* and *SYSTEC*, at the Faculty of Engineering of the University of Porto, which gave me all the conditions to develop this work. In particular, I would like to thank João Reis for his support and his constant encouragement. His comments and observations have always been of great interest. Also, a word of appreciation to Eliseu Pereira because of his efforts to build the *DINASORE* framework. Without this tool, I will not imagine how hard it would not be to implement and deploy the outputs resulting from this work.

I am deeply thankful to my parents, *Arlindo* and *Fernanda*, for raising and supporting me every step of the way about my education. You and I probably didn't imagine reaching this achievement, but that didn't stop you from encouraging me to reach for more.

Lastly, and most importantly, thanks to my family, my wife *Joana*, and my kids, *Gonçalo* and *Duarte*, as proof of my love and gratitude. It is only natural that a Ph.D. research work impacts one's personal life. For all those times that I was there, but my mind was elsewhere, for all those hours working instead of spending time with you, thank you, *Joana*, for your understanding and patience. On the other hand, one's personal life also dramatically impacts a Ph.D. research work. During this long journey, I got married, moved to a beautiful home, and became the father of two wonderful boys. I cannot imagine my life without them. So, thank you for bearing me, teaching me, and loving me. My personal life experiences also tough me essential lessons, which could not be learned any other way.

I dedicate this Thesis to you since I would never have come to its completion without your love and support.

Sincerely,

A handwritten signature in black ink that reads "Rui Pinto". The signature is written in a cursive, flowing style with a large initial 'R'.

About the Author

Rui Pinto was born in Santa Maria de Lamas, Santa Maria da Feira, Portugal on March 19th, 1989. In 2013, he received the M.Sc. degree in Electrical and Computer Engineering at the [FEUP](#)¹. After achieving his M.Sc. degree, he started working as a researcher at [SYSTEC](#)². He is currently an Auxiliary Researcher at [FEUP](#). Since 2013, as a Researcher, he has participated in at least 14 national and European Research & Development projects, targeting mainly the topics of digitalization of industrial processes, [WSAN](#)³, Edge/Cloud Computing, Smart Components, bio-inspired cyber-security, and Education 4.0. Moreover, he was an Invited Assistant at [FEUP](#) between 2017 and 2021, where he taught *Project Management Laboratory* in the *Master in Informatics and Computer Engineering*, *Programming Laboratories* and *Information Systems* in the *Master in Electrical and Computers Engineering*. He also had the opportunity to co-supervise 7 M.Sc. dissertations in smart manufacturing-related topics. A list of his publications and work experience can be found online⁴⁵⁶.

¹*Faculty of Engineering of the University of Porto*

²*Research Center for Systems and Technologies*

³*Wireless Sensor & Actuator Network*

⁴<https://www.researchgate.net/profile/Rui-Pinto-16>

⁵<https://orcid.org/my-orcid?orcid=0000-0002-0345-1208>

⁶<https://www.linkedin.com/in/rui-pinto/>

Publications

The following publications were produced during the course of this Thesis. When applicable, corresponding chapters indicate which publications are related and vice-versa.

- **Pinto, R., & Gonçalves, G. (2022).** Immune-based Industrial Network Intrusion Detection System. IEEE Access, (submitted). *This paper provides a comprehensive analysis of the deterministic and incremental versions of the Dendritic Cell Algorithm in intrusion detection problems. The work is featured in Chapter 7, Section 7.2. Rui Pinto contributed to the majority of the work.*
- **Pinto, R., & Gonçalves, G. (2021).** Application of Artificial Immune Systems in Advanced Manufacturing. Array, (In Press). [1] *This paper focused on a systematic literature review regarding the application of immune-based techniques in Advanced Manufacturing Systems. The work is featured in Chapter 4. Rui Pinto contributed to the majority of the work.*
- **Pinto, R., Gonçalves, G., Delsing, J., & Tovar, E. (2021).** Enabling Data-Driven Anomaly Detection by Design in Cyber-Physical Production Systems. Cybersecurity. [2]. *This paper focused on proposing a Model-based Engineering approach to implement intrusion detection, namely the incremental Dendritic Cell Algorithm, in Smart Manufacturing Systems. The work is featured in Chapter 6, Section 6.4 and Chapter 7, Section 7.3. Rui Pinto contributed to the majority of the work.*
- **Pinto, C., Pinto, R., & Gonçalves, G. (2021).** Towards Bio-inspired Anomaly Detection using the Cursory Dendritic Cell Algorithm. Algorithms. [3]. *This paper focused on a comprehensive review of Artificial Immune Systems and their application in anomaly detection problems. The work is featured in Chapter 3, Section 3.3. Rui Pinto contributed mainly with the mentorship and supervision of the work while formulating overarching research goals and aims and validating research outputs. He also managed and coordinated the research activity planning and execution and provided study materials or other analysis tools. Finally, he participated in the preparation, creation, and presentation of the published work, namely writing, reviewing, and editing the draft.*
- **Pinto, R., Gonçalves, G., Delsing, J., & Tovar, E. (2021).** Incremental Dendritic Cell Algorithm for Intrusion Detection in Cyber-Physical Production Systems. In Intelligent Computing (pp. 664-680). Springer, Cham. [4]. *This paper focused on proposing a new variant of the deterministic Dendritic Cell Algorithm, designated, incremental Dendritic Cell Algorithm, designed to tackle intrusion detection in an online fashion. The work is featured in Chapter 6, Section 6.3.2 and Chapter 7, Section 7.2. Rui Pinto contributed to the majority of the work.*

- **Pinto, R., Gonçalves, G., Tovar, E., & Delsing, J. (2020, September).** Attack Detection in Cyber-Physical Production Systems using the Deterministic Dendritic Cell Algorithm. In 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) (Vol. 1, pp. 1552-1559). IEEE. [5]. *This paper focused on the proposal of a new dataset for Intrusion Detection System validation while validating the performance of the deterministic Dendritic Cell Algorithm. The work is featured in Chapter 6, Section 6.2. Rui Pinto contributed to the majority of the work.*
- **Antao, L., Pinto, R., Reis, J., & Gonçalves, G. (2018, April).** Requirements for testing and validating the industrial internet of things. In 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 110-115). IEEE. [6]. *This paper consolidated knowledge regarding the Industrial Internet of Things and Cyber-Physical Production Systems' main requirements. The work is partially included in Chapter 2, Section 2.2. All authors contributed equally to this work.*
- **Pinto, R., Rossetti, R. J., & Gonçalves, G. (2016, July).** Wireless sensor network simulation for fault detection in industrial processes. In 2016 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH) (pp. 1-6). IEEE. [7]. *This paper was the first attempt to understand industrial anomalies and statistical anomaly detection techniques in industrial processes, which is not included in the Thesis. Rui Pinto contributed to the majority of the work.*

“Yo Adrian, I did it!”

Rocky II, 1979

Contents

List of Figures	xviii
List of Tables	xx
List of Algorithms	xxi
Abbreviations	xxiii
Symbols	xxix

I Background	1
1 Introduction	5
1.1 Context	5
1.2 Motivation	7
1.3 Aim & Scope	9
1.3.1 Research Questions	10
1.3.2 Main Contributions to Knowledge	11
1.4 Structure of the Thesis	12
1.5 How to Read this Thesis	13
2 Smart Manufacturing Systems	15
2.1 Introduction	15
2.2 Cyber-Physical Production Systems	16
2.2.1 Technology Origins	16
2.2.2 Reference Architecture	18
2.2.3 CPPS Requirements	20
2.3 Model-based Engineering	26
2.3.1 IEC 61499	27
2.3.2 DINASORE	28
2.3.3 Function Block Pipeline Metrics	30
2.4 Industrial Cyber-Security	33
2.4.1 Intrusion Detection Systems	35
2.4.2 Anomaly Network Detection	42
2.4.3 Networks and Anomalies	44
2.4.4 Network Attacks	48
2.4.5 Datasets	51
2.5 Conclusion	52

3	Autonomic Computing	55
3.1	Introduction	55
3.2	System Autonomy	56
3.2.1	Self-x Properties	57
3.2.2	Levels of Autonomy	59
3.3	Biologic Inspired Computation	59
3.3.1	Biologic Immune System	60
3.3.2	Artificial Immune Systems	63
3.4	The Dendritic Cell Algorithm	72
3.4.1	The deterministic Dendritic Cell Algorithm	75
3.5	Conclusion	78
II	Research	79
4	Application of AIS in SMS	83
4.1	Introduction	83
4.2	Systematic Literature Review	84
4.2.1	SLR Methodology	86
4.2.2	Characterization of AIS Applications in SMS	88
4.2.3	Conceptual Framework for Industrial AIS Applications	108
4.3	Conclusion	111
5	Immune-based Industrial IDS	113
5.1	Introduction	113
5.2	Context and Scope	114
5.2.1	Industrial Context	114
5.2.2	Immune Context	116
5.2.3	Danger Models for IDS	117
5.3	Requirement Analysis	120
5.3.1	SWOT Analysis	120
5.4	Research Strategies	122
5.4.1	Opportunities-Strengths	123
5.4.2	Opportunities-Weaknesses	124
5.4.3	Threats-Strengths	127
5.4.4	Threats-Weaknesses	129
5.5	Proposed Approach	133
5.6	Conclusion	134
6	IDS Proposal	137
6.1	Introduction	137
6.2	OPC UA Data Collection	138
6.2.1	M2M using OPC UA	140
6.3	Detection Approach	142
6.3.1	Object-Oriented Definition	142
6.3.2	incremental Dendritic Cell Algorithm	146
6.4	Security by Design	154
6.4.1	OPCUA Flow Sniffer	155
6.4.2	iDCA	156

6.5	Conclusion	157
7	DCA-based IDS Evaluation	159
7.1	Introduction	159
7.2	E1. iDCA vs. dDCA	161
7.2.1	Experimental Design	161
7.2.2	Pre-Processing	162
7.2.3	Experimental Setup	163
7.2.4	Methodology	163
7.2.5	M2M Using OPC UA	165
7.2.6	KDD Cup99	170
7.2.7	NSL-KDD	174
7.2.8	UNSW-NB15	178
7.2.9	Overall Results	182
7.3	E2. A-HIDS Pipeline	183
7.3.1	Experimental Design	183
7.3.2	Testing Methodology	184
7.3.3	Experimental Setup	185
7.3.4	Results & Discussion	187
7.3.5	Discussion	191
7.4	Conclusion	192
8	Conclusions	195
8.1	Main Achievements	196
8.2	Main Outputs & Impact	200
8.3	Limitations	201
8.4	Future Work	202
III	Appendix & Bibliography	205
A	Database Search Inputs	207
B	Industrial AIS Conceptual Framework	213
C	iDCA - System Modeling	215
D	A-HIDS Pipeline	219
E	SWOT Analysis	221
F	Cross-Correlation Results	223
G	IEC 61499-based Security Solutions	225
	References	227

List of Figures

1.1	Thesis structure.	13
2.1	<i>Cyber-Physical System</i> basic scheme.	17
2.2	Five layer <i>Internet of Things</i> architecture [8].	18
2.3	Generic interface of an <i>International Electrotechnical Commission 61499 Function Block</i>	28
2.4	General <i>Dynamic INtelligent Architecture for Software and MODular REconfiguration</i> architecture [9].	29
2.5	General <i>Dynamic INtelligent Architecture for Software and MODular REconfiguration</i> development process [9].	31
2.6	Main <i>Intrusion Detection System</i> components [10].	36
2.7	<i>Anomaly Intrusion Detection System</i> taxonomy [11].	44
2.8	Network reference models.	46
2.9	<i>OPC Unified Architecture</i> specifications [12].	47
2.10	Example of the <i>OPC Unified Architecture</i> service discovery.	48
3.1	<i>Monitor-Analyze-Plan-Execute-Knowledge</i> model for autonomic systems [13].	57
3.2	<i>Biological Immune System</i> main organs.	61
3.3	Simplified taxonomy of cells in the <i>Biological Immune System</i>	62
3.4	Simplified diagram of <i>Self/Nonsel</i> f theory [3]. It consists of 3 main steps: a) Immature <i>T-cells</i> with various <i>Antigen</i> receptors are produced in the bone marrow and migrate to the thymus; b) In the thymus, <i>T-cells</i> that recognize self <i>Antigens</i> undergo <i>Apoptosis</i> (programmed cell death). On the other hand, <i>T-cells</i> that don't recognize any self <i>Antigen</i> are allowed to live; c) Mature <i>T-cells</i> that don't recognize self <i>Antigens</i> are now used to recognize non self <i>Antigen</i> invading the body, such as virus and bacteria.	65
3.5	Flowchart of the <i>Negative Selection Algorithm</i>	66
3.6	Simplified diagram of the clonal selection theory [3]. A foreign organism contains specific <i>Antigen</i> that can be recognized by <i>B-cells'</i> receptors with high affinity. <i>B-cells</i> that recognize those <i>Antigen</i> are cloned, resulting in <i>Plasma B-cells</i> and <i>Memory Cells</i> . Short-lived <i>Plasma B-cells</i> release <i>Antibodies</i> that bond to the considered <i>Antigen</i> and inhibit the foreign organism. Long-lived <i>Memory Cells</i> will react quickly in future occurrences of the same <i>Antigens</i>	67
3.7	Flowchart of the <i>Clonal Selection Algorithm</i>	68

3.8	Simplified diagram of the immune network theory (Richter model [14]) [3]. <i>B-cells</i> with receptors of a given <i>Idiotyp</i> e (specific type of <i>Antibody</i> are capable of mutual interaction if their receptors have complementary specificity. Hence, the entirety of the <i>B-cell</i> system forms a functional network, which can stimulate or suppress a given immune response.	70
3.9	Simplified diagram of the <i>Danger Theory</i> [3]. Initially, <i>immature Dendritic Cells</i> navigate throughout the body, collecting encountered <i>Antigens</i> without discriminating between self or non-self. While navigating, <i>immature Dendritic Cells</i> also collect signals emitted by dying tissue cells. Damaged cells release chaotically their contents, which are identified as <i>Danger signals</i> by neighboring <i>immature Dendritic Cells</i> . On the other hand, when cell death is programmed, <i>immature Dendritic Cells</i> still collect <i>Antigens</i> while recognizing <i>Safe signals</i> . After collecting a sufficient amount of signals, <i>immature Dendritic Cells</i> evolve into either mature or semi-mature <i>Dendritic Cells</i> , depending if the overall context during <i>Antigens</i> collection was dangerous or safe. In both cases, <i>Dendritic Cells</i> migrate and present their collected <i>Antigens</i> to <i>T-Cells</i> . <i>T-Cells</i> that recognize the <i>Antigens</i> presented by <i>mature Dendritic Cells</i> and proceed to initiate an immune response, or they can tolerate <i>Antigens</i> if presented by <i>semi-mature Dendritic Cells</i>	71
3.10	Flow Chart of the main functionality of the <i>Dendritic Cell Algorithm</i>	73
4.1	<i>Preferred Reporting Items for Systematic Review and Meta-Analysis</i> flowchart of study inclusions and exclusions for the <i>Systematic Literature Review</i>	87
4.2	Year-wise publications.	89
4.3	Contributions from publishers.	91
4.4	Affiliation-wise publication analysis.	92
4.5	Frequently used keywords.	93
4.6	Characterization matrix of the studies selected.	94
5.1	<i>Dendritic Cell Algorithm</i> -based <i>Intrusion Detection System</i> in <i>Industrial Internet of Things</i> environments activity diagram.	134
6.1	<i>Cyber-Physical Production System</i> testbed for the <i>M2M</i> using <i>OPC UA</i> generation.	140
6.2	<i>Anomaly Host Intrusion Detection System</i> Pipeline.	155
6.3	<i>Execution Control Chart</i> of the <i>Anomaly Host Intrusion Detection System</i> Pipeline.	156
7.1	Experimental setup architecture.	186
B.1	Industrial <i>Artificial Immune System</i> applications conceptual framework.	214
C.1	<i>Incremental Dendritic Cell Algorithm Unified Modelling Language</i> class diagram.	216
C.2	Network data collection <i>Unified Modelling Language</i> sequence diagram.	217
C.3	<i>Incremental Dendritic Cell Algorithm Unified Modelling Language</i> sequence diagram.	218
D.1	Experimental setup - Function Block Pipeline.	220

List of Tables

2.1	<i>Cyber-Physical System and Internet of Things</i> generic structure, and information flow.	19
2.2	<i>Cyber-Physical Production System</i> testing requirements.	22
3.1	<i>Dendritic Cell Algorithm</i> main nomenclature.	74
3.2	Example of suggested weighting values in <i>Dendritic Cell Algorithm</i>	75
4.1	Comparison of the related work.	85
4.2	General string queries used in the search string.	88
4.3	Overview of publications included in the <i>Systematic Literature Review</i>	90
4.4	Journals encompassed in the study.	92
4.5	<i>Artificial Immune System</i> conceptualization.	104
5.1	<i>Dendritic Cell Algorithm</i> -based <i>Intrusion Detection System</i> literature gaps.	133
6.1	Comparison of different datasets for network intrusion detection.	142
7.1	Relationship between selected features and signal categories.	163
7.2	Experimental methodology: test case scenarios.	164
7.3	<i>deterministic Dendritic Cell Algorithm</i> validation results in the <i>M2M Using OPC UA</i>	165
7.4	<i>Incremental Dendritic Cell Algorithm</i> validation results in <i>M2M Using OPC UA</i>	167
7.5	<i>deterministic Dendritic Cell Algorithm</i> and <i>Incremental Dendritic Cell Algorithm</i> global performance comparison in <i>M2M Using OPC UA</i>	169
7.6	<i>deterministic Dendritic Cell Algorithm</i> validation results in <i>KDD Cup99</i> training set.	170
7.7	<i>Incremental Dendritic Cell Algorithm</i> validation results in <i>KDD Cup99</i>	172
7.8	<i>deterministic Dendritic Cell Algorithm</i> and <i>Incremental Dendritic Cell Algorithm</i> global performance comparison in <i>KDD Cup99</i>	173
7.9	<i>deterministic Dendritic Cell Algorithm</i> validation results in <i>NSL-KDD</i>	175
7.10	<i>Incremental Dendritic Cell Algorithm</i> validation results in <i>NSL-KDD</i>	176
7.11	<i>deterministic Dendritic Cell Algorithm</i> and <i>Incremental Dendritic Cell Algorithm</i> global performance comparison in <i>NSL-KDD</i>	177
7.12	<i>deterministic Dendritic Cell Algorithm</i> validation results in <i>UNSW-NB15</i> training set.	178
7.13	<i>Incremental Dendritic Cell Algorithm</i> validation results in <i>UNSW-NB15</i>	180
7.14	<i>deterministic Dendritic Cell Algorithm</i> and <i>Incremental Dendritic Cell Algorithm</i> global performance comparison in <i>UNSW-NB15</i>	181

7.15	<i>deterministic Dendritic Cell Algorithm and Incremental Dendritic Cell Algorithm</i> average global performance comparison considering all datasets.	183
7.16	Performance overhead evaluation methodology results.	188
7.17	Evaluation methodology results using <i>Function Block</i> Pipeline measures.	190
E.1	<i>Dendritic Cell Algorithm</i> SWOT analysis in its application in binary classification problems.	222
F.1	<i>M2M Using OPC UA</i> : feature cross-correlation with class label.	223
F.2	<i>KDD Cup99</i> : feature cross-correlation with class label.	223
F.3	<i>NSL-KDD</i> : feature cross-correlation with class label.	224
F.4	<i>UNSW-NB15</i> : feature cross-correlation with class label.	224
G.1	<i>International Electrotechnical Commission 61499</i> -based Security Solutions.	226

List of Algorithms

1	Pseudocode of the <i>deterministic Dendritic Cell Algorithm</i> implemented in this Thesis.	77
2	<i>Incremental Dendritic Cell Algorithm</i> pseudocode.	148

Abbreviations

A-HIDS	<i>Anomaly Host Intrusion Detection System</i>
A-IDS	<i>Anomaly Intrusion Detection System</i>
A-NIDS	<i>Anomaly Network Intrusion Detection System</i>
ACM	<i>Association for Computing Machinery</i>
ACO	<i>Ant Colony Optimization</i>
ACS	<i>American Chemical Society</i>
AEAD	<i>Authenticated Encryption with Associated Data</i>
AES	<i>Advanced Encryption Standard</i>
AGV	<i>Autonomous Guided Vehicles</i>
AI	<i>Artificial Intelligence</i>
AIN	<i>Artificial Immune Network</i>
aiNet	<i>The Artificial Immune Network algorithm</i>
AIRS	<i>The Artificial Immune Recognition System</i>
AIS	<i>Artificial Immune System</i>
AISFD	<i>Artificial Immune System-based Fault Diagnostic</i>
ANN	<i>Artificial Neural Network</i>
APC	<i>Antigen Presenting Cell</i>
ARFF	<i>Attribute Relationship File Format</i>
ARTIS	<i>ARTificial Immune System</i>
AWA	<i>Artificial Awareness Architecture</i>
BCA	<i>The B-Cell Algorithm</i>
BFB	<i>Basic Function Block</i>
BHS	<i>Baggage Handling System</i>
BIS	<i>Biological Immune System</i>
CDCA	<i>Cursory Dendritic Cell Algorithm</i>
CDF	<i>Cumulative Distribution Function</i>
CFB	<i>Composite Function Block</i>
CL4FB	<i>Confidentiality Layer for Function Blocks</i>
CLONALG	<i>The CLONal selection ALGORITHM</i>
COTS	<i>Commercial Off-The-Shelf</i>

CPU	<i>Central Processing Unit</i>
CPS	<i>Cyber-Physical System</i>
CPPS	<i>Cyber-Physical Production System</i>
CSA	<i>Clonal Selection Algorithm</i>
CSV	<i>Comma Separated Values</i>
DARPA	<i>Defense Advanced Research Projects Agency</i>
DCA	<i>Dendritic Cell Algorithm</i>
DCS	<i>Distributed Control System</i>
dDCA	<i>deterministic Dendritic Cell Algorithm</i>
DDoS	<i>Distributed Denial of Service</i>
DeepDCA	<i>Deep Learning and Dendritic Cell Algorithm</i>
DG	<i>Distribution Generation</i>
DNN	<i>Deep Neural Network</i>
DINASORE	<i>Dynamic INtelligent Architecture for Software and MODular REconfiguration</i>
DMS	<i>Distribution Management System</i>
DNP3	<i>Distributed Network Protocol 3</i>
DoS	<i>Denial of Service</i>
DT	<i>Danger Theory</i>
ERP	<i>Enterprise Resource Planning</i>
ECC	<i>Execution Control Chart</i>
FB	<i>Function Block</i>
FBDK	<i>Function Block Development Kit</i>
FBR-DCA	<i>Fuzzy boundary region-based Dendritic Cell Algorithm</i>
FBRT	<i>Function Block Run-Time Environments</i>
FDDR	<i>Fault Detection, Diagnosis and Recovery</i>
FDE	<i>Fault Diagnostic Engine</i>
FEUP	<i>Faculty of Engineering of the University of Porto</i>
FLA-DCA	<i>Fuzzy lower approximation-based Dendritic Cell Algorithm</i>
FLISR	<i>Fault Location, Isolation, and Service Restoration</i>
FPGA	<i>Field-programmable Gate Array</i>
FRST-DCA	<i>Fuzzy Rough Set Theory-based Dendritic Cell Algorithm</i>
GPS	<i>Global Positioning System</i>
GUI	<i>Graphical User Interface</i>
hDCA	<i>Haskell-based Dendritic Cell Algorithm</i>
HIDS	<i>Host-based Intrusion Detection Systems</i>
HMI	<i>Human-Machine Interaction</i>
HVAC	<i>Heating, Ventilation, and Air-Conditioning</i>

HTTP	<i>Hypertext Transfer Protocol</i>
I4.0	<i>Industry 4.0</i>
I²S	<i>Intelligent Immune System</i>
IBM	<i>International Business Machines Corporation</i>
ICMP	<i>Internet Control Message Protocol</i>
ICT	<i>Information and Communication Technology</i>
ICS	<i>Industrial Control Systems</i>
iDCA	<i>Incremental Dendritic Cell Algorithm</i>
IDE	<i>Integrated Development Environments</i>
IDPS	<i>Intrusion Detection and Prevention System</i>
IDS	<i>Intrusion Detection System</i>
IEC	<i>International Electrotechnical Commission</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IET	<i>Institution of Engineering and Technology</i>
IG	<i>Information Gain</i>
IIoE	<i>Industrial Internet of Everything</i>
IIoT	<i>Industrial Internet of Things</i>
IIRA	<i>Industrial Internet Reference Architecture</i>
IMCS	<i>Industrial Mixer Control System</i>
IoT	<i>Internet of Things</i>
IoT-A	<i>Internet of Things Architecture</i>
IP	<i>Internet Protocol</i>
ISO	<i>The International Organization for Standardization</i>
KE	<i>Key Exchange</i>
KNN	<i>K Nearest Neighbor</i>
LAN	<i>Local-Area Network</i>
LISYS	<i>Lightweight Intrusion detection SYStem</i>
LOA	<i>Level of Autonomy</i>
LVQ	<i>Learning Vector Quantization</i>
M2M	<i>Machine-to-Machine</i>
MAPE	<i>Monitor-Analyze-Plan-Execute</i>
MAPEK	<i>Monitor-Analyze-Plan-Execute-Knowledge</i>
MAS	<i>Multi-Agent System</i>
MBE	<i>Model-based Engineering</i>
MES	<i>Manufacturing Execution System</i>
MIC25	<i>Made In China 2025</i>
MITM	<i>Man-in-the-Middle</i>

ML	<i>Machine Learning</i>
MSR	<i>Modular Self-Reconfigurable</i>
MTTF	<i>Mean Time to Failure</i>
MTTR	<i>Mean Time to Repair</i>
NIDS	<i>Network-based Intrusion Detection System</i>
NIST	<i>National Institute of Standards and Technology</i>
NKC	<i>Natural Killer Cell</i>
NSA	<i>Negative Selection Algorithm</i>
OPC UA	<i>OPC Unified Architecture</i>
OOP	<i>Object-Oriented Programming</i>
OSI	<i>Open Systems Interconnection</i>
PCA	<i>Principal Component Analysis</i>
PCAP	<i>Packet Capture</i>
PID	<i>Proportional Integral Derivative</i>
PLC	<i>Programmable Logic Controller</i>
PoD	<i>Ping of Death</i>
PRISMA	<i>Preferred Reporting Items for Systematic Review and Meta-Analysis</i>
PSCC	<i>Power Systems Computation Conference</i>
QoS	<i>Quality of Service</i>
QR-DCA	<i>QuickReduct algorithm-based Dendritic Cell Algorithm</i>
R2L	<i>Remote to Local</i>
RAM	<i>Random Access Memory</i>
RAMI 4.0	<i>Reference Architectural Model Industrie 4.0</i>
RC-DCA	<i>REDUCT and CORE-based Dendritic Cell Algorithm</i>
RST	<i>Rough Set Theory</i>
RST-DCA	<i>Rough Set Theory-based Dendritic Cell Algorithm</i>
ROC	<i>Receiver Operating Characteristic</i>
RTE	<i>Runtime Environment</i>
RTOS	<i>Real-Time Operating System</i>
SCADA	<i>Supervisory Control and Data Acquisition</i>
SIFB	<i>Service Interface Function Block</i>
SKAB	<i>Skoltech Anomaly Benchmark</i>
SLR	<i>Systematic Literature Review</i>
SMLC	<i>Smart Manufacturing Leadership Coalition</i>
SMS	<i>Smart Manufacturing System</i>
SNN	<i>Self Normalizing Neural Network</i>
SOA	<i>Service-oriented Architecture</i>

SVM	<i>Support Vector Machine</i>
SWaT	<i>Secure Water Treatment</i>
SWOT	<i>Strengths, Weaknesses, Opportunities and Threats</i>
SYSTEC	<i>Research Center for Systems and Technologies</i>
TCP	<i>Transmission Control Protocol</i>
TLR	<i>Toll-Like Receptor</i>
TLS	<i>Transport Layer Security</i>
TORUS	<i>Traceability of Requirements Using Splices</i>
TRL	<i>Technology Readiness Level</i>
U2R	<i>User to Root</i>
UDCA	<i>Ubiquitous Dendritic Cell Algorithm</i>
UDP	<i>User Datagram Protocol</i>
UML	<i>Unified Modelling Language</i>
UNSW	<i>The University of New South Wales</i>
WIP	<i>Work-in-Progress</i>
WSN	<i>Wireless Sensor Network</i>
WSAN	<i>Wireless Sensor & Actuator Network</i>
XAI	<i>Explainable Artificial Intelligence</i>
XML	<i>Extensible Markup Language</i>

Symbols

$\emptyset$	<i>Number of Non-classifications</i>
Δt^D	<i>Intrusion Detection Duration</i>
$\#Ag$	<i>Quantity of Antigen Presented</i>
$\#Iter$	<i>Number of Iterations</i>
$\#Dc$	<i>Number of Dendritic Cells</i>
$\#mAg$	<i>Quantity of Mature Antigen</i>
Acc	<i>Accuracy</i>
Ag	<i>Antigen</i>
At	<i>Anomaly Threshold</i>
$AUC-ROC$	<i>Area Under the Receiver Operating Characteristic Curve</i>
$C(i)$	<i>Function Block System Complexity</i>
C^*	<i>System Complexity</i>
C_x	<i>Signal Concentration</i>
Ck	<i>Inflammatory Cytokines</i>
csm	<i>Co-stimulatory Signal</i>
D	<i>Difficulty</i>
$DC(i)$	<i>Function Block Data Complexity</i>
DC^*	<i>Data Complexity</i>
Dc	<i>Dendritic Cell</i>
Ds	<i>Danger Signal</i>
E	<i>Program Effort</i>
$F1$	<i>F-measure</i>
FN	<i>False Negative</i>
FNR	<i>False Negative Rate</i>
FP	<i>False Positive</i>
FPR	<i>False Positive Rate</i>
f_{out}	<i>Output Events</i>
k	<i>Context Output Value</i>
LOC	<i>Lines of Code</i>
$LOC(Alg_i)$	<i>Lines of Source Code</i>

$LOC(ECC)$	<i>Execution Control Chart States and Transitions</i>
M	<i>Number of Cell Migrations</i>
M_H	<i>Halstead's Metrics</i>
M_m	<i>McCabe's Metric</i>
$M_m(Alg_i)$	<i>Function Block Source Code Cyclomatic Complexity</i>
$M_m(ECC)$	<i>Function Block Execution Control Chart Cyclomatic Complexity</i>
mat	<i>Mature Dendritic Cell Cytokine</i>
$mcav$	<i>Mature Context Antigen Value</i>
MI	<i>Maintainability Index</i>
$\hat{N}$	<i>Estimated Length</i>
n	<i>Program Vocabulary</i>
n_1	<i>Total Number of Distinct Operators</i>
n_2	<i>Total Number of Distinct Operands</i>
N	<i>Program Length</i>
N_1	<i>Total Number of Operators</i>
N_2	<i>Total Number of Operands</i>
NI	<i>Data Inputs</i>
NO	<i>Data Outputs</i>
P	<i>Global System Performance</i>
$PAMP$	<i>Pathogen-Associated Molecular Pattern</i>
PR	<i>Purity Ratio</i>
$R_{\Delta D}$	<i>Detection rate</i>
$R_{\emptyset}$	<i>Classification rate</i>
$S(i)$	<i>Function Block Structural Complexity</i>
S^*	<i>Structural Complexity</i>
$semi$	<i>Semi-mature Dendritic Cell Cytokine</i>
Ss	<i>Safe Signal</i>
T	<i>Total Classification Time</i>
t_m	<i>Migration Threshold</i>
TN	<i>True Negative</i>
TNR	<i>True Negative Rate</i>
TP	<i>True Positive</i>
TPR	<i>True Positive Rate</i>
V	<i>Program Volume</i>
W_x	<i>Signal Weight</i>
W_{AUC}	<i>Area Under Curve Weight</i>
$W_{\Delta D}$	<i>Detection Rate Weight</i>

$W_{\emptyset}$ *Classification Rate Weight*

Part I

Background

Part I - Background: Introduction

Part I introduces the work developed in the Thesis and presents all the theoretical context. It provides the necessary background regarding the basic concepts in the research fields of *Smart Manufacturing System* and *Autonomic Computing*. Besides, it introduces the main techniques, theories, and technologies adopted in this work, such as *Intrusion Detection System*, *Dendritic Cell Algorithm* and *Dynamic INtelligent Architecture for Software and MODular REconfiguration*.

Chapter 1

Introduction

1.1 Context

The journey to the 4th Industrial revolution, also known as **I4.0**¹, is marked by significant and high-impact technological advances, which were and are being applied in industrial contexts. Starting from the First Industrial Revolution in the late 18th century, regarding mechanized manufacturing processes, and adopted steam and water power to increase productivity. Then, in the late 19th and early 20th century, the Second Industrial Revolution began with the adoption of innovations such as electricity, interchangeable parts, assembly lines, and mass production. Digital Revolution marked the 1950s. In this Third Industrial Revolution, there was a shift from mechanical technology to digital circuits and systems, by the transition to **PLCs**² [15]. Currently, connected devices enable an unprecedented capacity for data and information sharing, applying typical **IoT**³ scenarios to industrial contexts. This is explained by the incredible growth in the technological ability of computational (embedded) devices and networks.

I4.0 [16] corresponds to the new industrial paradigm where traditional methods of control are being transformed into novel mechanisms that allow remote and distributed digital control. These mechanisms make networking and distributed computing essential for modern automation systems. Besides, Edge devices or **IoT** enabled devices within an intelligent system are becoming increasingly common in the industrial context since they enhance overall efficiency in task performance. The realization of **IoT** functionalities are possible with the development of **CPSs**⁴, where the physical and virtual worlds are related and mutually dependent. This means that actuation decisions in the virtual world would impact the physical world, which in its turn, will influence back the virtual world for a new decision-making process.

CPPS⁵ have already found several applications in different manufacturing scenarios, such as supply chain management, production line, and operations maintenance. Regarding supply

¹Industry 4.0

²Programmable Logic Controllers

³Internet of Things

⁴Cyber-Physical Systems

⁵Cyber-Physical Production Systems

chain/warehouse management processes, CPPSs enable tracking in real-time to ensure the highest efficiency at each step in such applications as demand management, order fulfillment, manufacturing flow management, and return management. The production line and the operations related to its maintenance can also benefit from AI⁶ based wireless inspection tools, allowing for real-time control of performance, durability, and safety of the production and the resulting products. Manufacturing operations are, therefore, the number one use case in SMSs⁷ today in Europe, followed by production & asset management. These use cases use CPPSs to monitor and maintain assets (e.g., industrial manufacturing devices) that are part of the production value chain with the end goal of improving process efficiency for manufacturing.

In 2011, the *Industrie 4.0* term was first used at the Hannover Fair. In 2013, the German Industrie 4.0 Working Group presented a final report [17], stating that I4.0 is the strategic initiative to secure the future of the German manufacturing industry. However, Germany was not the only country that has recognized the trend to deploy smart CPPSs into the manufacturing industry. Similar initiatives were also found, such as the U.S. SMLC⁸ [18], the Chinese MIC25⁹ [19], the French *Usine du Futur* [20], the British *Catapult* [21], the Korean *Industry Innovation 3.0* [22], etc.

Later, in 2015, at the World Economic Forum in Davos, the German chancellor Angela Merkel urged all of Europe to embrace I4.0 [23]. With this message, it was introduced to European industries the urgency to deal with the fusion of the online world and the world of industrial production. The paradigm is changing, and those who are the leaders in the digital domain will take the lead in industrial production. The support of the research in this area increased dramatically over the years, and I4.0 became a hot topic among practitioners and academics in Europe and one of the top priorities for research centers, universities, and companies. For the first time in the history of the industrial evolution, a revolution was announced before it happened [24]. This new paradigm provided various opportunities for companies and research institutes to shape the future of I4.0 actively.

Despite this difficulty for many people and companies, when it comes to identifying and implement I4.0 scenarios, over the years, the concept has become more evident. All the implemented I4.0 scenarios converge to very similar requirements and goals in order to achieve added value in different manufacturing applications. The added value is based on ICT¹⁰ advances, such as IoT, AI and Cloud/Edge. Oztemel and Gursev [25] present 12 goals to achieve I4.0, which were originally inspired in [17]:

- **Standardization** - Standardization of systems and creation of a reference architecture;
- **Optimized Management** - Performing optimized management of complex systems;

⁶Artificial Intelligence

⁷Smart Manufacturing Systems

⁸Smart Manufacturing Leadership Coalition

⁹Made In China 2025

¹⁰Information and Communication Technology

- **Communication Network Infrastructure** - Manage industrial network infrastructure for communication reliability;
- **Safety & Security** - On one hand, setting a safe and secure environment to human operator. On the other hand, enable system protection against misuse and unauthorised access;
- **Human Management** - Organizing and managing the workplace in a human-centered way, by considering processes, human, automation and environmental changes;
- **Upskilling** - Personnel training for upskilling the workforce and management (Education 4.0);
- **Regulatory Framework** - Creating an organizational framework for data organization, to ensure new technology is compliant with the law;
- **Resource Efficiency** - Increasing the efficiency of resource utilization (Green Manufacturing);
- **Autonomy** - Self-behaving systems (Self-*), where minimum human interaction is required;
- **Process Efficiency & Product Quality** - Product and Process interaction (due to autonomic behavior);
- **Data Analytics** - Big data analysis, for data-driven autonomous decision-making;
- **Adaptability** - Adaptability and flexibility, for responsiveness to changes in real time.

1.2 Motivation

The main goal for industrial entities to exchange information is to enhance task performance and increase the overall system productivity. This information may represent the physical world - tool position, machine conditions, and the virtual world - simulation models and electronic documentation. The primary sources of information regarding the physical world are sensors, which can measure the entities' state and environmental conditions. This context-aware information is indispensable for entities to perform appropriate decisions in many applications, such as machine parametrization, predict equipment wear out for schedule maintenance phases, and predict human error in real-time, based on the fatigue information of human operators. Raw sensor data must be aggregated and interpreted to higher-value context information to analyze and infer characteristics of such a complex system to perform autonomous decisions. This is only possible thanks to data-driven methods based on sensor fusion and data mining.

Negotiation and collaboration skills between autonomous entities allow decentralized and improved decision-making by utilizing local and global information, resulting in increased overall productivity by reducing waste, extending equipment life, increasing efficiency, reducing downtime, and performing smart maintenance. In such an intelligent system, where decentralized autonomous decisions are made without human control, the prominent role of humans shifts from a machine operator towards a strategic decision-maker and a flexible problem-solver in a human-machine collaboration environment. This level of system autonomy implies innovative assistance

systems. On a virtual level, information is adequately aggregated for visualization purposes to make informed decisions. On a physical level, human safety is guaranteed by an enhanced and intuitive interaction between humans and machines.

The security of smart CPSs and IIoT¹¹ compliant systems is crucial to the development and acceptance of the technology. The IoT is an immature technology and, one of the critical issues that affect the usage of IoT is the lack of mature and complete security models and standards. Unlike the Internet, the IoT may integrate several heterogeneous networks. As the IoT environment becomes complex, the resulting security issues present more challenges than any other existing network system. The challenges to this premise include at least the following three aspects [26]:

1. Cost – Mobile Edge devices in a CPPSs are resourced constrained, i.e., lack of memory, reduced computational capability, and limited power supply. The existing network security protocols do not consider these restriction factors.
2. Security – Compared to traditional networks, the IoT merges more networks, which perplexes the security issues.
3. Privacy – Edge devices in a CPPSs may not be able to defend from all forms (physical and cyber) of attacks.

Some efforts have been made to secure industrial control systems by using classical defense strategies and monitoring systems, such as firewalls, cryptography, and access control. Regarding CPSs and IoT security, more recent efforts focus on mapping solutions from existing domains, such as SCADA¹² and WSANs. Protection and security defenses are essential, but they are not enough. Defense mechanisms are generally effective, but breaches will eventually occur due to stolen credentials, misconfigurations, unpatched vulnerabilities, unknown attacks, and rogue insiders. It is a matter of time for a system to be hacked, which means tools and plans to deal with breaches are also needed. Detection plays a significant role in reducing the system's exposure to intrusions. In the security domain, this solution could be referred to as IDS¹³ [27].

Although detection and prevention techniques were studied to be applied to SCADA and WSANs, these solutions were usually not designed for CPPSs. They may not cover all the security requirements of IoT. Also, even if they were suitable, no system can be full-prove to all attack vectors, especially considering zero-day attacks. Zero-day attacks are novel approaches to invade existing systems. They are being developed considering the exploitation of different vulnerabilities of the system [28, 29]. In the literature, one can find several successful manufacturing-related cyber-physical attacks, driven by monetary and political interests [30]. Well-known examples are: i) the Maroochy Shire Water Services attack in 2000, which resulted in dumping around waterways in Queensland, Australia, large amounts of raw sewerage [31]; ii) the *Stuxnet* worm, which damaged an Iranian nuclear facility of Natanz back in 2010 [32]; and iii) the German Steel Mill attack at the end of 2014, which blast a furnace by preventing it from correctly shutting down [33].

¹¹Industrial Internet of Things

¹²Supervisory Control and Data Acquisition

¹³Intrusion Detection System

Thus, considering CPPS security, threats may target the physical part, the cyber part, or even the connectivity level between both. This makes CPPSs vulnerable in several different layers of their architecture. Nowadays, several academic and commercial solutions exist regarding computing security. However, because of the unique requirements and specifications of CPPSs, these solutions may not always be feasible to be implemented. IDSs used in traditional Internet and computer systems may not work in such complex systems. CPPSs require autonomic, adaptive, and self-managed approaches to decide with minimum human supervision what is the best action plan to be taken facing a specific detected attack.

Also, considering the increasing adoption of intelligent components in CPPSs, the implementation of security features can be very complex. Security features should be included in the CPPS already in the design stage. This requirement raises a need to support the design and verification of CPPSs by including prevention and detection components as default. Thus, *Security by Design* can be achieved using the MBE¹⁴ development paradigm. In industrial scenarios, a well-known MBE approach follows the IEC¹⁵ 61499, which is already being used as a general framework to develop distributed measurement and control systems.

1.3 Aim & Scope

This Thesis aims to investigate and make contributions regarding the development of an IDS architecture for industrial Edge devices, which follows a MBE approach for system design, such as the industrial standard IEC 61499. We employ an autonomic research perspective to this problem by studying the DCA¹⁶, which belongs to the family of AISs¹⁷. In this study, the following objectives are defined:

- Review important background theory in topics addressed in this work, such as CPPSs, MBE, IEC 61499 standard, industrial IDSs, Self-* properties, AISs and the DCA.
- Review relevant work regarding autonomic approaches used in SMSs, such as AISs applications, considering they TRL¹⁸ and LOA¹⁹.
- Analysis of the main strengths and weaknesses of the DCA, while characterizing the research strategies to take advantage of opportunities and threats in implementing an industrial IDS.
- Apply and assess the usage of a DCA-based IDS to detect network intrusions in SMSs' industrial communication and protocols, such as OPC UA²⁰.

¹⁴Model-based Engineering

¹⁵International Electrotechnical Commission

¹⁶Dendritic Cell Algorithm

¹⁷Artificial Immune Systems

¹⁸Technology Readiness Level

¹⁹Level of Autonomy

²⁰OPC Unified Architecture

- Study the suitability of **DCA** for online or real-time detection and in what conditions the **DCA** detection performance holds when incremental learning is used for data processing.
- Study the **IEC** 61499 standard and understand the feasibility of designing a **FB**²¹-based immune **IDS** for overall **CPPS** design and management.

1.3.1 Research Questions

Based on the main objectives described before, the research question addressed in this Thesis is:

In what ways can the Dendritic Cell Algorithm be used for anomaly intrusion detection in Cyber-Physical Production Systems?

We decompose the previous research question into nine specific and interrelated minor research questions. The first six research questions are qualitative, i.e., they enable the development of a theoretical framework for the research study. In this case, we have the application of **AIS** in **SMS**s and requirement analysis for research strategies identification. Answers to these questions are found in Chapter 4 and Chapter 5.

RQ T.1 *What is the research trend of Artificial Immune System applications in Smart Manufacturing Systems?*

RQ T.2 *What is the status of Artificial Immune System applications in Smart Manufacturing Systems?*

RQ T.3 *What are the main motivations for using Artificial Immune Systems in Smart Manufacturing Systems?*

RQ T.4 *What are the research gaps and future areas within Smart Manufacturing Systems based on Artificial Immune System solutions?*

RQ T.5 *What are the primary strengths and weaknesses when developing Dendritic Cell Algorithm-based solutions for classification problems?*

RQ T.6 *What research strategies can be adopted for addressing opportunities and threats in industrial anomaly detection?*

The following three research questions are quantitative, i.e., they are predictions about the expected relationships among different variables in this study, namely influence of incremental learning in **DCA** detection speed and accuracy and **FB** impact on the *Complexity* and *Maintainability* of an **IDS** security solution. Answers to these questions are found in Chapter 7.

RQ T.7 *What impact in the Dendritic Cell Algorithm's detection rate and accuracy has an incremental learning approach?*

²¹Function Block

RQ T.8 *What is the cause-effect relationship between algorithm variables, runtime properties and detection performance in Dendritic Cell Algorithm-based intrusion detection?*

RQ T.9 *How effective is IEC 61499 in supporting the implementation of low complexity and high maintainability security by design Cyber-Physical Production Systems?*

1.3.2 Main Contributions to Knowledge

Given the previous study objectives and the research questions to be pursued, this Thesis makes the following contributions to the field:

- Conceptualization of topics included in the areas of SMSs and Autonomic Computing, such as CPPSs, MBE, IEC 61499 standard, industrial IDSs, Self-* properties, AISs and the DCA.
- SLR²² on AISs applications used in SMSs.
- Conceptual framework about the implementation of AIS in SMS.
- SWOT²³ analysis to identify research gaps, and propose strategies about the development of a DCA-based approach for intrusion detection in industrial scenarios.
- Development of the *M2M using OPC UA* [34] dataset. This dataset was generated while capturing OPC UA network traffic in a laboratory CPPS testbed, including the injection of attacks in the network.
- Development of the iDCA²⁴, a variant of the dDCA²⁵, oriented for online network-based intrusion detection.
- Validation of both dDCA and iDCA using different intrusion detection related datasets, such as the recently proposed *M2M using OPC UA* [34], and existing datasets in the literature, such as the *KDD Cup99*, *NSL-KDD* and *UNSW-NB15*.
- Comprehensive analysis and further comparison of dDCA and iDCA application in intrusion detection problems. This analysis is based on the measurement of algorithm's performance and assessment of runtime properties when tuning algorithm variables.
- Development of a FB architecture, based on the IEC 61499 standard, for the iDCA deployment in a CPPS in early design stages.
- Validation of the iDCA- FB architecture using a CPPS testbed, by enabling real-time M2M²⁶ communication between Edge devices and network intrusion detection.

²²Systematic Literature Review

²³Strengths, Weaknesses, Opportunities and Threats

²⁴Incremental Dendritic Cell Algorithm

²⁵deterministic Dendritic Cell Algorithm

²⁶Machine-to-Machine

1.4 Structure of the Thesis

This Thesis is organized into four main parts.

Part I - **Background**, along with the introduction to the document (Chapter 1), provides to the reader detail about all the theoretical concepts, topics, methodologies, and technologies used in this work. The technological background presented was organized into two different chapters. Chapter 2 tackles concepts and technologies related with **SMSs**, such as **CPPSs**, **IEC 61499** and the **DINASORE**²⁷ for the **MBE** approach, and industrial cyber-security, focusing on **IDSs**. On the other hand, Chapter 3 presents concepts related to the *Autonomic Computing* paradigm, such as Self-* properties in computer/engineering systems and consequent **LOA**, **BIS**²⁸ inspiration and related **AISs** models. Finally, it is presented in more detail the **DCA**, since it is the immune-model inspiration that we used in this work.

Part II - **Research** includes all the research contributions in this Thesis to answer the research questions identified earlier. This part consists of four different chapters.

It starts by Chapter 4, where we provide a **SLR** on **AISs** applications used in **SMSs**. This **SLR** is based on a comprehensive evaluation of journal and conference articles in the fields of **AISs** and **SMSs**. Based on our findings, we propose a conceptual framework about the implementation of **AIS** in **SMS**, aiming to guide manufacturers to adopt **AIS** in their manufacturing applications.

Then, Chapter 5 describes in detail the industrial and immune contexts for the intrusion detection problem. We also provide a comprehensive **SWOT** analysis about **DCA**-based **IDS**, which is used to assess research strategies and design the proposed approach. This approach is described in detail in Chapter 6, focusing on the research and development of immune-based models and the deployment of such solutions in an **CPPS** following a **FB**-based approach. Notably, we start by proposing a new dataset for **IDS** validation, designated *M2M using OPC UA*. Then, we propose the **iDCA**, an incremental version of the **dDCA**. Finally, we propose a **FB** architecture, based on the **IEC 61499** standard, for the **iDCA** deployment in a **CPPS** in early design stages.

In Chapter 7, all the experiments and achieved results are described in detail. The first experiment (**E1. iDCA vs. dDCA**) tackles the validation of both **dDCA** and **iDCA** in different datasets used in intrusion detection problems, including the proposed *M2M using OPC UA*. The main performance metrics used for assessment are the detection accuracy, in terms of **AUC-ROC**²⁹, *Detection Rate* and *Classification Rate*. On the other hand, the second experiment (**E2. A-HIDS Pipeline**) uses the **iDCA**, now oriented to a **FB**-based solution, and validates the proposed architecture in a **CPPS** testbed. The main performance metrics used for assessment are the *Resource Consumption*, *System Overhead* and **FB**-oriented metrics for system *Maintainability* and *Design Complexity*.

Moreover, Chapter 8 summarizes the main findings of this Thesis by presenting the main achievements and outcomes. Also, it is provided a discussion on the limitations of the results and perspectives for future work.

²⁷Dynamic INtelligent Architecture for Software and MODular REconfiguration

²⁸Biological Immune System

²⁹Area Under the Receiver Operating Characteristic Curve

Finally, Part III - [Appendix & Bibliography](#) include the appendices and bibliography. The appendices refer to a collection of supplementary material, such as *Database Search Inputs* (Appendix [A](#)), *Industrial AIS Conceptual Framework* (Appendix [B](#)), *iDCA - System Modeling* (Appendix [C](#)), *A-HIDS Pipeline* (Appendix [D](#)), *SWOT Analysis* (Appendix [E](#)), *Cross-Correlation Results* (Appendix [F](#)), and *IEC 61499-based Security Solutions* (Appendix [G](#)). On the other hand, the bibliography refers to the list of all the sources used and cited in this Thesis.

The outline of the first two parts in this Thesis is represented in Figure 1.1.

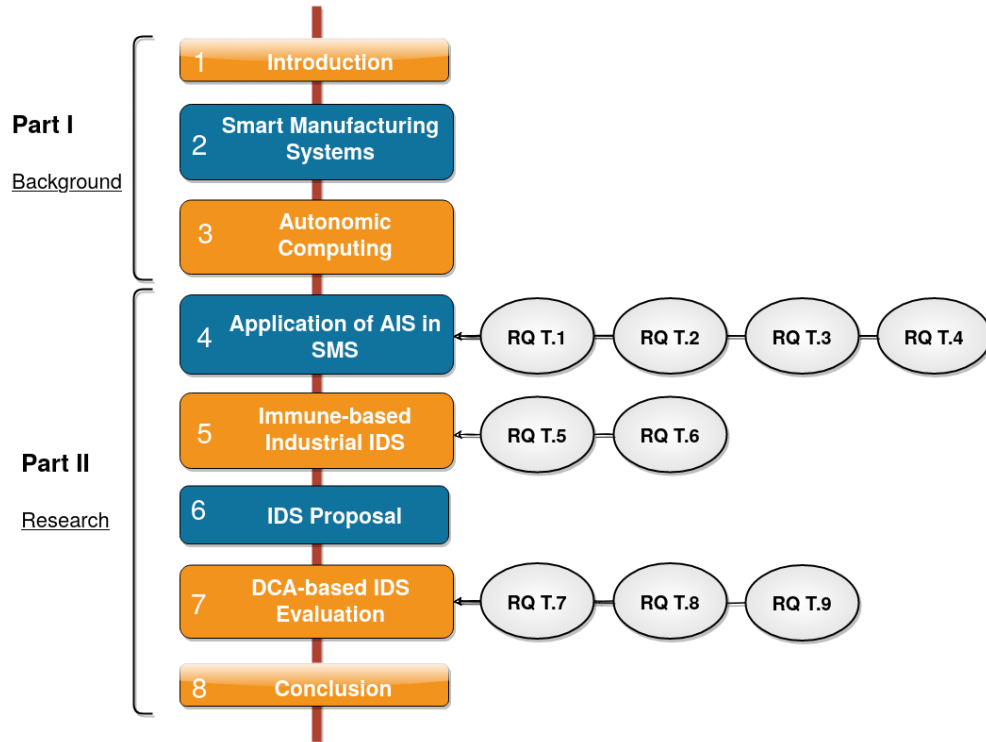


Figure 1.1: Thesis structure.

1.5 How to Read this Thesis

This Thesis can be seen as a reference text that provides knowledge about the merger of the [I4.0/SMS](#) disciplines and the [AIS](#) research field. Both these topics come together to tackle a massive problem in today's [SMS](#) scenarios, namely the security and intrusion detection in [CPPS](#). The proposed solution takes advantage of the Self-* and autonomic capabilities introduced by *Soft-computing* and bio-inspired algorithms. Moreover, this Thesis is a guidebook about the [MBE](#)-oriented development and deployment of bio-inspired security-based solutions in the Edge layer of the [CPPS](#), following the [IEC 61499](#) standard. Thus, the audience for this Thesis may be interested in the fields of *Computational Intelligence*, *Biological Inspired Computing*, *Artificial Immune Systems*, *Smart Manufacturing Systems*, *Intrusion Detection Systems*, and *Cyber-Physical Production Systems*. The most typical profiles of readers may be:

- Research Scientists, concerned with a theoretical and empirical investigation of AIS.
- Software and Informatics Engineers who seek guidance for implementing, deploying, and maintaining a DCA-based strategy using industrial standards.
- 14.0 Professionals, concerned with implementing in their manufacturing shop-floor intrusion detection by design, enabling and adapting the best features of ICT security to be used in an industrial-related context.
- Undergraduate and Graduate Students interested in learning about SMSs and/or *Autonomic Computing*

Chapter 2

Smart Manufacturing Systems

2.1 Introduction

In this chapter, the reader will learn about the technological background related to SMS¹. This theoretical chapter will be helpful in later chapters to understand the developments, experimental methodologies, and achieved results. We start by introducing CPPSs² in Section 2.2, by detailing the concept and describing main technology enablers, general architectures, and analyzing critical requirements. In each requirement, we propose a set of validation tests and implementation challenges, which can be found in the design, analysis, verification, and validation stages of implementing a CPPS.

Based on these implementation challenges in different CPPS development stages, we direct the focus to the MBE³ paradigm in Section 2.3. This development methodology provides advantages that document-based systems engineering cannot provide when implementing complex systems such as CPPS. Within this methodology, detail is provided regarding the IEC⁴ 61499, as a well-known industrial standard that allows designing and managing CPPSs. Also, the DINASORE⁵ framework is described since it follows the IEC 61499 standard to enable software and modular reconfiguration in CPPSs.

Next, the reader attention is directed in Section 2.4 to the industrial cyber-security topic, going into detail about IDSs⁶. Particularly, we address NIDS⁷ main characteristics while detailing network concepts and typical network attacks in IP⁸ networks and protocols, such as the OPC UA⁹. We wrap up by identifying the main datasets used to validate NIDSs. Finally, Section 2.5 concludes the chapter, stating the final remarks about the work presented.

Partial results of the presented work in this chapter have been published in:

¹Smart Manufacturing Systems

²Cyber-Physical Production Systems

³Model-based Engineering

⁴International Electrotechnical Commission

⁵Dynamic INtelligent Architecture for Software and MODular REconfiguration

⁶Intrusion Detection Systems

⁷Network-based Intrusion Detection System

⁸Internet Protocol

⁹OPC Unified Architecture

- Antao, L., Pinto, R., Reis, J., & Gonçalves, G. (2018, April). Requirements for testing and validating the industrial internet of things. In 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 110-115). IEEE. [6].¹⁰

2.2 Cyber-Physical Production Systems

Straightforwardly, **CPSs**¹¹ refers to the integration of computational and physical processes. The **CPPS** term is used when **CPSs** are employed in an industrial environment. However, there is no general definition of **CPS**, which often leads to ambiguity when describing these types of systems. It is usual to find loosely many classical terms to refer to **CPPS**, such as **SCADA**¹², **ICS**¹³, among others. In the **SMS** paradigm, this ambiguity is more evident when multiple other concepts and topics from different research areas overlap with **CPS**, such as embedded systems, **IoT**¹⁴, **MASS**¹⁵, industrial networks, **WSNs**¹⁶ or *Digital Twin*. This section provides a comprehensive description about **CPSs** in terms of technology origins, reference architectures, and main requirements. In the remainder of this Thesis, unless a specific reference is made necessary, we adopt the **CPPS** terminology to refer to any manufacturing infrastructure that enables the **SMS** paradigm within an industrial context.

2.2.1 Technology Origins

ICT¹⁷ is regarded as one of the most significant fields in recent decades and the next [35]. Because of the constant miniaturization of technology, information processing devices have shifted from giant computers to compact portable electronic devices integrated into larger items. The use of **ICT** devices embedded in bigger systems gave rise to the concept of *Embedded Systems*. Marwedel [36] defined these systems as information processing systems embedded within enclosing products that are distinguished by their real-time constraints, reliability, concurrency, and efficiency requirements.

Over the years, these embedded systems have been developed with sophisticated sensors and actuators, interfaces, processing units, complex control loops, software agents, and communication channels. Because the embedded computation on these devices employs closed models and algorithms to control the electronics, the connection to physical processes is usually overlooked. The advent of the concept **CPS** was lately emphasized by the tight relationship between computing and physics. Lee defines **CPS** as the integration of computing and physical processes [37, 38].

¹⁰©2018 IEEE. Reprinted, with permission, from Antao, L., Pinto, R., Reis, J., Requirements for testing and validating the industrial internet of things, April/2018.

¹¹*Cyber-Physical Systems*

¹²*Supervisory Control and Data Acquisition*

¹³*Industrial Control Systems*

¹⁴*Internet of Things*

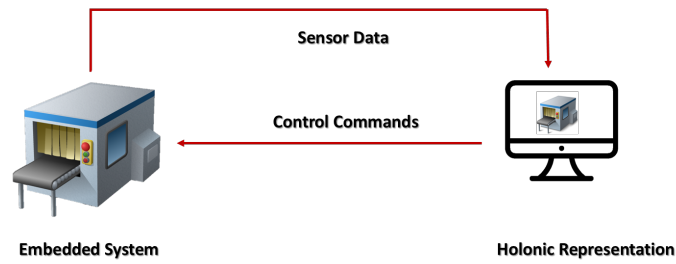
¹⁵*Multi-Agent Systems*

¹⁶*Wireless Sensor Networks*

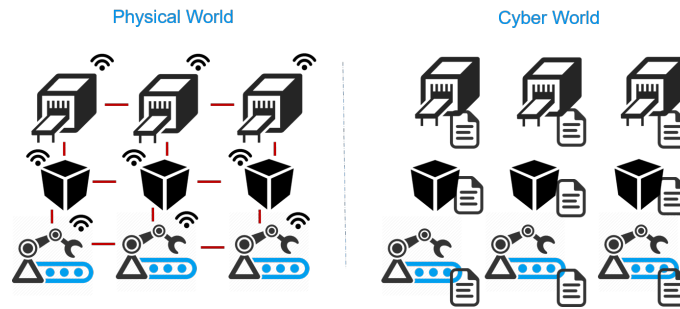
¹⁷*Information and Communication Technology*

Later, Hellinger and Seeger [39] expanded on the notion of CPS by emphasizing the role of networking, control-loops, and distribution. Indeed, CPSs vary from embedded systems in that CPSs function on a much broader scale. In contrast, an embedded system is a self-contained system restricted to a single device. Many embedded systems and the connectivity with other devices and systems are often included in CPPSs.

Usually, in a CPPS, numerous components of the system are often represented by several Digital Twins. Because centralized control is impractical for large-scale system integration, distributed control is enabled through collaboration across networked Digital Twins in the MAS architectural style. Networked Digital Twins enable collaborative information processing and distributed autonomous decision-making systems in conjunction with sensing and actuation capabilities. Figure 2.1 represents a CPPS basic scheme, where a simple control-loop is represented, which enables the autonomy property of CPPSs. On the other hand, it is represented a typical CPPS environment, followed by IIoT¹⁸ principles, where all Edge devices can engage and interact with each other.



(a) Cyber-Physical System control-loop.



(b) Cyber-Physical System environment.

Figure 2.1: Cyber-Physical System basic scheme.

Because of advancements in wireless technology, Edge devices may now be connected to the network for remote monitoring and control. These systems based on device communication, also known as M2M¹⁹ communication, were constructed on closed networks designed expressly for this purpose. Nowadays, this network connection is usually built on IP communication protocols,

¹⁸Industrial Internet of Things

¹⁹Machine-to-Machine

such as the **OPC UA**, which allows Edge devices to be monitored and managed through the Internet. The fundamental idea of **M2M** is to connect items between each other, while **IoT** is to connect them using the Internet.

In recent years, the benefits of using **IoT** principles in the industry have been strengthened by merging and integrating a variety of different technologies, including Cloud/Fog Computing and Big Data, **WSANs**²⁰, **SOA**²¹ and **MASs**. **IIoT** platforms are supported by **CPPSs** because they allow standard communication interfaces across Edge devices, as well as connection and interoperability for horizontal and vertical networking between devices and platforms [40].

2.2.2 Reference Architecture

There are several proposed architectures for the **IIoT** [41] given by initiatives and research projects, such as the **RAMI 4.0**²² [42], **IIRA**²³ [43], and **IIoT-A**²⁴ [44]. Other designs were proposed, e.g., the partition into five layers [8], as illustrated in Figure 2.2, which has been adopted by many authors [45]. Despite these efforts to provide a reference design for **IIoT**, there is yet no widely accepted architecture in the research community.

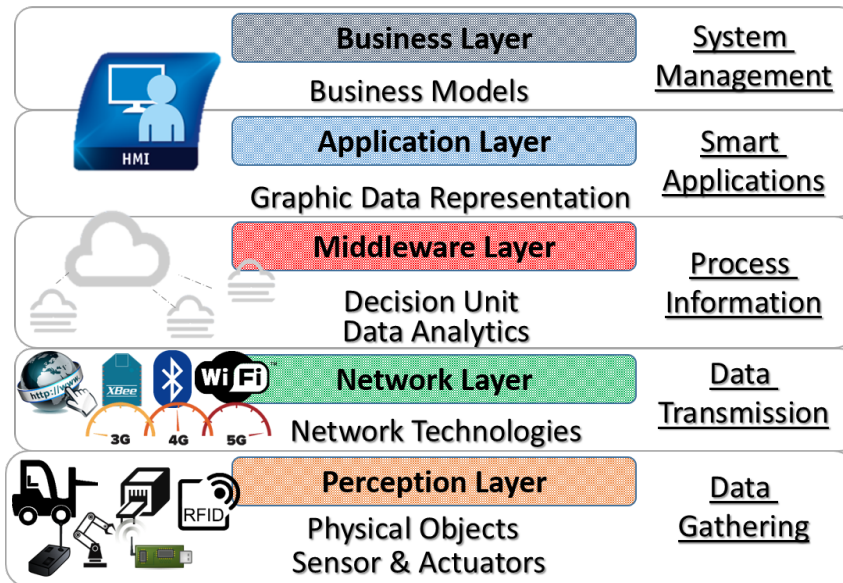


Figure 2.2: Five layer *Internet of Things* architecture [8].

This study adopts the **IIoT** architecture proposed by Wu *et al.* [8], which divides an **IoT** system into five layers, namely *Business*, *Application*, *Middleware*, *Network* and *Perception*.

The *Perception* layer is made up of physical objects that are monitored/controlled by sensor and actuator devices, with the primary goal of collecting sensor data and command actuation. The

²⁰Wireless Sensor & Actuator Networks

²¹Service-oriented Architecture

²²Reference Architectural Model Industrie 4.0

²³Industrial Internet Reference Architecture

²⁴Internet of Things Architecture

sensor data is subsequently sent to the *Middleware* layer via the *Network* layer, using a variety of network protocols. The information is processed at the *Middleware* layer, where it may be saved and analyzed using powerful data analytics approaches. The primary objective is to establish an autonomous decision-making process. This will lead to sending actuation orders back to physical objects to conduct activities that will alter the overall conditions of the physical environment. The information gathered/analyzed may be displayed to an end-user via the *Application* layer, which may also be used to control the overall system. Finally, the *Business* layer enables system administrators to monitor and strategically manage the overall functionality of the *IoT* platform.

Furthermore, according to Han *et al.* [46], the *CPS* architecture is characterized in five different levels, which can be mapped in the different *IIoT* layers, namely: 1) *Physical level*; 2) *Sensor & Actuator level*; 3) *Network level*; 4) *Control level*; and 5) *Information level*. Furthermore, according to Wang *et al.* [47], a general information flow of *CPS*s may be divided into four major phases: *Monitoring*, *Networking*, *Computing* and *Actuating*. The *Displaying* phase was included since it is an essential element in the information flow that is appropriate for the new generation of mobile *CPS*s. This *CPS* information flow is based on the *MAPEK*²⁵ architecture (for further details, see Chapter 3). Table 2.1 represents the relationships between all the *IoT* layers, *CPS* levels and information flow phases.

Table 2.1: *Cyber-Physical System* and *Internet of Things* generic structure, and information flow.

IoT Layer	CPS Level	Information Flow Phase
Business layer	Information level	Displaying
Application layer	Information level	Displaying
Middleware layer	Control level	Computing
Network layer	Network level	Networking
Perception layer	Physical level Sensor & Actuator level	Monitoring Actuating

The initial phase in the information flow is the *Monitoring*, which involves monitoring physical processes, ambient conditions, and the status of the physical item at the *IoT Perception* layer. This is made feasible by sensors placed in the physical items being watched. This process produces data generated in real-time by various sensors. The *Networking* stage follows, which consists of gathering and transferring sensor data. First, a gateway node (referred to as an *IoT gateway*) may aggregate data generated in real-time from several sensor nodes, which collects and pre-processes this information. Second, data is pulled from the gateway to an external device for analysis. Without the need for a gateway, sensor data may be transferred straight from sensor nodes to an external device for analysis. This stage results in physical data aggregation and transfer.

The *Computation* stage is responsible for analyzing and processing the data gathered during the *Monitoring* phase. Compared to low-power sensor nodes, external servers for data processing

²⁵ *Monitor-Analyze-Plan-Execute-Knowledge*

are typically associated with powerful machinery. These machines may also support and operate virtualizations (Digital Twins) of physical items. The Digital Twins are in charge of determining if the physical operations meet pre-defined criteria and the system's overall goals. Data storage, fusion, categorization, and regression are the foundations of information processing resources, commonly referred to as Cloud/Edge services. These strategies derive quality information from raw measured data and make suitable judgments when selecting the best action plan to implement in the system.

In the *Displaying* stage, a HMI²⁶ device will show the information processed and saved in the *Computation* step to an end-user. According to the various degrees of information access, both end-users in the *Application* and *Business* layers will have access to the desired information. They will also manage a few system activities by submitting control instructions. The *Actuation* step corresponds to the execution of the action plans selected in the *Computation* step or the end user's control instructions. The control directives supplied to the actuators are the consequence of this stage. The actuators will alter one or more physical processes, hence altering the state of the world. The environmental effect of these acts will be assessed in the *Monitoring* stage, and a new world state will be studied in the *Computation* step, where new action plans will be proposed.

2.2.3 CPPS Requirements

Some requirements have been developed over time, notably for generic IIoT platforms and CPPSs. In most situations, only specific domains of the IIoT platform are approached, providing requirements specific to that sector. For example, Kaestle *et al.* [48] provided a categorized and organized list of requirements for CPPS communication and data sharing in smart factories. Dillon *et al.* [49] examine QoS²⁷ requirements in CPPSs applications. Data security is one of the most often discussed aspects of the IIoT and CPPSs. Alqassem [50] proposes an engineering framework to ensure that the privacy and security needs for IIoT are correctly developed. Se-Ra Oh and Young-Gab Kim [51] proposed IIoT security requirements while evaluating various research work on IIoT security needs. Vermesan and Friess [52] reveal the progress of IIoT applications by referring to several requirements in the IIoT security, trust, and privacy domains. Amadeo *et al.* proposed a high-level design of a networking architecture that addresses IIoT problems and approaches a number of IIoT requirements.

In terms of industrial scenarios, requirements relevant to the industrial automation domain must be addressed in addition to those established for a generic IoT platform. There are several types of processes in industrial automation, and each one has particular functional and non-functional requirements. Fantana *et al.* [53] presented a concise set of requirements for the IIoT hardware and software. Breivold and Sandström [54] proposed several general IIoT challenges, including data and service security, data integrity and information privacy, scalability and interoperability, as well as some industrial challenges, including fault tolerance, functional safety, and scalable collaboration, among others.

²⁶Human-Machine Interaction

²⁷Quality of Service

Based on the work of Breivold and Sandström [54], we establish eight essential CPPS's requirements in this section, which describes the CPPS's requirements that should be checked while developing, maintaining and testing the system. Furthermore, for each requirement, various tests and implementation challenges are described. The critical CPPS requirements are summarized in Table 2.2, namely *Scalability*, *Reliability*, *Security & Privacy*, *Timing & Determinism*, *Safety*, *Recovery*, *Interoperability* and *Reconfigurability*.

2.2.3.1 Scalability

Scalability is described as the system's capacity to grow and accommodate more resources required for successful functioning. In terms of CPPS, a scalable system should be able to interact with an increasing number of nodes while also allowing for a growing quantity of data to be shared between them. Under these situations, the system must maintain responsiveness by avoiding increased network latency and a fall in throughput. One of the primary benefits of implementing CPPSs is their flexibility in incorporating new equipment into the system. Hence the *Scalability* criterion must be addressed.

To test and demonstrate *Scalability* in CPPSs, one needs to increase the number of nodes in the system, as well as the amount of data they generate and exchange. When it comes to expanding the number of nodes, both real devices and simulation tools that imitate the same behavior can be used. The amount of data produced may be tested from two perspectives: 1) Increase the sampling rate of a sensor, confirming that no data is lost; 2) Increase the quantity of data exchanged by employing node-level processing methods. Furthermore, and assuming that Big Data is part of a company's business plan, expanding the quantity of data analytics services accessible on a server to process sensor data is a test that should be undertaken. In this case, the system's infrastructure needs to scale accordingly as more services and interdependencies are added.

The issues with these tests are mostly related to resources and operating expenses. Increased node count indicates more equipment and resources allocated for installation. However, one may opt to run a simulation program to offset these expenditures, as the costs in this scenario are mostly on specialist individuals. When it comes to boosting the throughput of each node, the primary issue is that it must be reconfigured to adjust the sampling rate or execute processing modules for statistical analysis. Finally, due to the design and orchestration of such services, *Scalability* testing by expanding the number of services accessible in a server may be problematic.

2.2.3.2 Reliability

The capacity of a system to perform as predicted under predetermined conditions over a pre-defined time is measured by its *Reliability*. The primary industrial *Reliability* metrics, such as MTTF²⁸ or MTTR²⁹, are defined by the concepts of faults/anomalies and failures. A fault, sometimes known as an anomaly, is an error in a program's step, process, or data specification.

²⁸Mean Time to Failure

²⁹Mean Time to Repair

Table 2.2: *Cyber-Physical Production System* testing requirements.

CPPS Requirement	Testing	IoT Layer	Challenges
<i>Scalability</i>	1. Increase number of network nodes. 2. Increase available data. 3. Increase service availability.	Perception Perception Middleware	- Latency; - Cost of acquiring or upgrade devices; - Constrained data processing methods.
<i>Reliability</i>	1. Long term execution. 2. Anomaly injection to generate failures. 3. Extreme environment conditions. 4. Overall counting of received/sent packages.	End-to-end End-to-end Perception Network	- Relationship between anomaly and generated failures; - Code verification methods.
<i>Security & Privacy</i>	1. Cyber attack injection. 2. Stealing sensitive data. 3. Security resources are up and running.	End-to-end End-to-end End-to-end	- Unavailability to inject zero-day attacks; - Simulate known attacks; - Lack of expertise in cyber security methods.
<i>Timing & Determinism</i>	1. Guarantee cycle time. 2. Test equipment process with varying parameters.	End-to-end Perception	- Identify which component introduces delay; - Evaluate environmental impact over the process.
<i>Safety</i>	1. Simulation of safety process parameters (both in controlled and relevant environment). 2. Counting number of accidents in the shop-floor.	Perception Perception	- Knowing the accident's cause; - Reliable parameter simulation; - Availability of relevant environment to test.
<i>Recovery</i>	1. Evaluate continuous operation of the system when some of its parts are shut-down. 2. Maintain previous state after rebooting (both individual node or global system). 3. Analyze time of reboot.	End-to-end End-to-end End-to-end	- Identify the damage level that precludes system's recovery; - Identify what is the previous state of the system; - Identify the acceptable time of rebooting.
<i>Interoperability</i>	1. Send messages with non matching semantics or undefined ontology (between different nodes or modules in the same node). 2. Integration with 3rd party platforms (legacy entities).	End-to-end End-to-end	- Communication API does not exists; - Non compatibility between existing APIs.
<i>Reconfigurability</i>	1. Analyze time of reconfiguration. 2. Verify system reconfiguration on new topology. 3. Verify system reconfiguration when a node is added.	End-to-end End-to-end End-to-end	- Verify reconfiguration in complex systems; - Identify acceptable time of reconfiguration.

An error is the expression of a flaw in a system. Failure is defined as a system's failure to fulfill its needed function within the set performance requirements. Because of its nature and structure, analyzing a [CPPS Reliability](#) is rugged. While failure models can represent performance deterioration in hardware and physical aspects, these models do not apply to software and computational elements. Methods for identifying defects, abnormalities, or software bugs in code can be exceedingly challenging to implement in such complex systems.

The system may be deployed in an atypical environment and validated to see if it continues to function appropriately, to evaluate the dependability of the [CPPS](#)'s physical aspects. However, to do accurate analysis, one must consider the integrated nature of [CPPSs](#) (both cyber and physical world). Considering the overall performance of each component of the [CPPS](#), evaluating system operation for lengthy periods to find problems in different components may be a viable test to execute. Furthermore, the most critical components of a [CPPS](#) are related to its networking capabilities. Another test that may be conducted is counting received/sent packages in each communication to analyze message loss. Finally, another test may be the injection of faults into the [CPPS](#) to cause failures and analyze overall system behavior. One of the significant problems with this approach is knowing upfront the relationship between an anomaly and the corresponding failure generation. It can be challenging to identify the faults that should be injected in the first place to provoke a [CPPS](#) failure.

2.2.3.3 Security & Privacy

While security and dependability are distinct concepts, they are highly related. A non-secure system is not dependable, and a non-reliable system cannot be secure. In reality, in the cyber security area, faults are recognized as intrusions because both faults and intrusions emerge as errors, which result in failures. In this context, these failures are referred to as attacks. Unlike errors and failures, the distinction is that someone with hostile intent deliberately initiates intrusions and attacks. The rising complexity of [CPPSs](#) and the underlying computer equipment and communication connections to the outside world have enlarged the attack surface. This complexity enables a wide range of cyber assaults regarding privacy, security, or both. Privacy attacks are typically passive and may need getting access to private data. In contrast, security attacks compromise the integrity of the information or devices in the system by directly assaulting the physical or cyber parts of the [CPPS](#).

The first test should ensure that security resources like anti-virus, firewalls, and cryptographic systems are functioning correctly when checking if the [CPPS](#) is secure. However, a lack of knowledge of security mechanisms and how they should be applied in [CPPSs](#) may make this test challenging. After successfully hacking the system, one may target system components to analyze its behavior in the face of various attacks. In terms of privacy, one may attempt to steal sensitive information. In terms of security, one may try to insert multiple threats into the system. The biggest problem with attack injection is determining how to imitate the assault in the first place. In addition, various attacks are being developed mainly to target and discover new vulnerabilities in [CPPSs](#). These attacks were never seen before, and the implemented security resources may not

be prepared to detect them (zero-day attack). So, one is unavailable to test the system for these attacks since it is impossible to simulate and inject unknown attacks.

2.2.3.4 Timing & Determinism

Timing & Determinism may be one of the essential requirements for CPPSs in industrial applications since it focuses on the direct consequences of product quality and process cycle time. *Timing & Determinism*, as the name implies, is concerned with ensuring that an installation of a CPPS does not jeopardize the stated cycle time for all digital equipment or modify the process parameters, hence maintaining product quality. In the event of virtualizing a whole manufacturing system, various types of equipment are represented as *Digital Twins* that communicate with one another over the CPPS. This means that any errors when executing such a system result in unwanted changes in product quality.

To test this requirement, formal registration of machine parameters vs. product quality, as well as the cycle time per existing line, must be undertaken before and after the deployment of the CPPS. Product quality decline may be identified quickly by recording the product quality, indicating that more CPPS's changes are necessary. Furthermore, any communication or processing delay is easily identifiable with cycle time registration. The difficulties related with such testing, in the event of discovering any delay or quality deterioration, determine if they are a direct result of the CPPS or if they were previously present in the system before the CPPS adoption. Furthermore, it is difficult to determine if these changes are the result of any uncontrolled external element generated by the CPPS itself, such as ambient circumstances.

2.2.3.5 Safety

The goal of *Safety* is to safeguard the system from unintended physical failures and to avoid risks such as explosions, fires, floods, chemical spills, and possible equipment crashes. Furthermore, there is currently great interest in investigating the *Safety* of systems that contain industrial collaborative robotics. This is justifiable by the necessity to safeguard human operators from mishaps when executing jobs jointly with industrial robotic manipulators. The gap between *Safety* and security is blurring as systems get more sophisticated and interconnected because both share the same purpose of preventing the CPPS from failing. Measuring the frequency of incidents on the shop floor and human operator casualties may indicate whether or not the system is safe. The main issue is determining whether a human or machine fault caused the mishap. In the context of test execution, defects in the system may be introduced to cause accidents. However, there is a considerable risk associated with this practice (both human and material). For that reason, safety tests should be performed in a simulated environment and, when available, in an appropriately controlled environment. The main problem regarding simulation and controlled environments is the lack of reliability in parameters and trustworthy external conditions to the process.

2.2.3.6 Recovery

Recovery refers to CPPS's capacity to restore nominal functioning via resetting as a mechanism for completely recovering from transitory errors. Restart-based techniques are broadly classified into two types: revival, which is used to restart a failing component reactively, and rejuvenation, which is used to restart functional components to prevent state deterioration. In the case of CPPSs, *Recovery* can relate to specific elements in the system or the complete system recovery. When a tiny portion of the system is compromised, the remaining part should operate and compensate for the functionality of the compromised components. The initial test may be to shut off system components and see if the system continues to run partially. The critical issue with this strategy is determining the system's level of *Reconfigurability* in advance, i.e., the limit of damage extent in the system that permits it to continue running. In terms of restart-based techniques, one may check whether the restarted component retains its state after restarting and examine the time it takes to do so. The primary issues revolve around determining the system's initial state and, based on system functionality, the allowed time of reboot.

2.2.3.7 Interoperability

The M2M communication attribute, along with standardized communication, is a cornerstone for CPPS deployments, primarily through developments in IoT platforms and architectures. This implies that standardized communications protocols, semantics and ontology definition are required for harmonized and intelligible communication. Furthermore, the utilized networking infrastructure is crucial since it can severely limit its capacity and throughput.

To see if the system can manage *Interoperability* issues, one should consider communicating with network nodes using messages with unspecified semantics. Also, messages with valid semantics but incorrect content should be sent in the application context. In the latter situation, a node may anticipate getting a sensor data message of one type but instead receives a message of another type, which it should discard rather than perceive as accurate. These tests determine whether or not the system stops and whether or not the proper behavior is used. Furthermore, the connectivity to external systems should be tested utilizing several current services with legacy systems (e.g., MESs³⁰ and ERPs³¹). The issues with these tests are primarily at the system level, where a third-party application connection (e.g., MES) may not be established. Thus, a dedicated implementation needs to be developed to test system communications, which usually is out of scope according to the system design. Also, one may find incompatibility issues when the communication protocols are not the same.

2.2.3.8 Reconfigurability

Reconfigurability refers to the ability to switch between CPPS configurations. A CPPS configuration is a well-defined specification of how the system's nodes should interact and M2M com-

³⁰Manufacturing Execution Systems

³¹Enterprise Resource Plannings

munication is carried out. Network nodes should optimize their communication architecture, such as peer-to-peer, tree-like, or star-like, to meet production needs such as guaranteeing positioning, product quality, safety, and cycle time. As a result, system reconfiguration improves system communications by changing the network topology.

CPPS reconfiguration testing distinguishes itself by establishing a new configuration for the system in which, for example, two nodes that rely on each other cease communication. In such a circumstance, the system should terminate the process because there is not enough information to continue running while the other nodes keep their settings. Another test is to add a new node to the system and see whether a reconfiguration to accommodate that node is successful while maintaining overall functioning. Finally, the duration of reconfiguration should be checked. If the system takes too long to reconfigure, its purpose for a given production need is no longer acceptable. The issues with these tests are linked mainly to the complexity of determining whether or not the reconfiguration occurred as described and determining a fair period for reconfiguration. In terms of reconfiguration time, assuming that the best-case scenario is as minimum as possible, the difficulty associated is to indicate when a reconfiguration time is no longer accepted since there might be different thresholds according to different production periods.

2.3 Model-based Engineering

Considering the described requirements and the necessity of performing multiple tests to validate and test different system properties, the implementation and execution of these tests may be even more challenging and more expensive than the development of the **CPPS** itself. Thus, an alternative system development methodology may be an excellent approach to support the requirements, design, analysis, verification, and validation associated with the development of complex systems. For this, the **MBE** is considered.

MBE is a system development paradigm that employs visual modeling techniques in system design and management at all phases of the system's life cycle. This paradigm helps engineers and software developers to migrate from traditional system development techniques (document-based and code-centric) to more effective requirements-driven and architecture-centric approaches. Traditional development methodologies are fast becoming inappropriate owing to the amount of complexity in these systems when considering the manner **CPPSs** are built, implemented, and managed [55]. All system components are designed individually and then combined to create the final system using traditional procedures. The **MBE** technique is excellent for the creation of **CPPSs** since it consists of utilizing models to design software and perform component testing, readily adapting the complexity and dynamism of **CPPSs**.

Given this context, **CPPS** development needs flexible automation that allows for changes with minimal engineering work. The **MBE** method enables component-oriented design, in which components of the **CPPS** are modeled independently, with their roles, functionality, and purpose explicitly defined. These components can be assembled and reused on-demand in various cost-effective ways. This method should automate the design's complex and error-prone operations.

No defects are introduced, and the final implementation works according to the models' vision.

2.3.1 IEC 61499

CPPSs are commonly used to control manufacturing and process control systems. Because the environment is often heterogeneous, i.e., it comprises hardware from several manufacturers, developing distributed manufacturing and process control systems is time-consuming and error-prone. The **IEC** adopted the **IEC 61499** standard for distributed **FB**³² in 2005, which may be utilized to ease the creation of **CPPS**s. This standard specifies a component-oriented design process, allowing reusable components that encapsulate state and behavior, and supporting many object-oriented programming methods. The encapsulation of software components is enhanced with **IEC 61499**, offering a vendor-independent standard and simplifying support for **M2M** communication. Its distributed capabilities and built-in support for dynamic reconfiguration provide the required infrastructure for **I4.0**³³, enabling the design and control of massive automation systems or **IIoT** applications [56] in a **FB**-oriented way.

FBs are functional software units that are compatible with the **IEC 61499** standard, with the **BFB**³⁴ type being the fundamental entity. Other **FB** definitions exist, such as the **CFB**³⁵ and **SIFB**³⁶. When prompted by events, a **BFB** executes software code, which after execution can produce and pass on additional events. The software code executes algorithms that process incoming data to change internal variables and output outcomes in most cases. When the algorithm completes its execution, these results are coupled with output events, which will trigger the execution of another **FB**. **CFB**s, on the other hand, are containers of **FB** Pipelines, i.e., **FB**s organized in networks. Finally, **SIFB**s are **FB**s with no internal behavior since their interface is modeled, but the behavior is determined later in the implementation [57]. In the remainder of this Thesis, unless a specific reference is made necessary regarding **CFB** and **SIFB**, we adopt the **FB** terminology to refer to the **BFB** type. Figure 2.3 depicts a generic interface of an **IEC 61499 FB**.

FBs are linked together by event and data connections to form **FB** Pipelines, which may be executed using a **RTE**³⁷. The standard's execution paradigm is implemented by the **RTE**. Along with **IEC 61499** standard, several **IDE**s³⁸ arose for orchestration, mapping, and deployment of **IEC 61499** enabled applications based on various **RTE**s, which focus on real-time execution of **FB** applications according to its execution model. The *4DIAC-IDE* [58], *NxtStudio* [59] and **FBDK**³⁹ [60] are some relevant **IEC 61499 IDE**s. *4DIAC-IDE* is an open-source Eclipse-based program that is arguably the most widely used **IDE** that allows for the creation and deployment of pipelines based on **FB**s. The Eclipse Project is used as a basic development foundation by the

³²Function Block

³³Industry 4.0

³⁴Basic Function Block

³⁵Composite Function Block

³⁶Service Interface Function Block

³⁷Runtime Environment

³⁸Integrated Development Environmentss

³⁹Function Block Development Kit

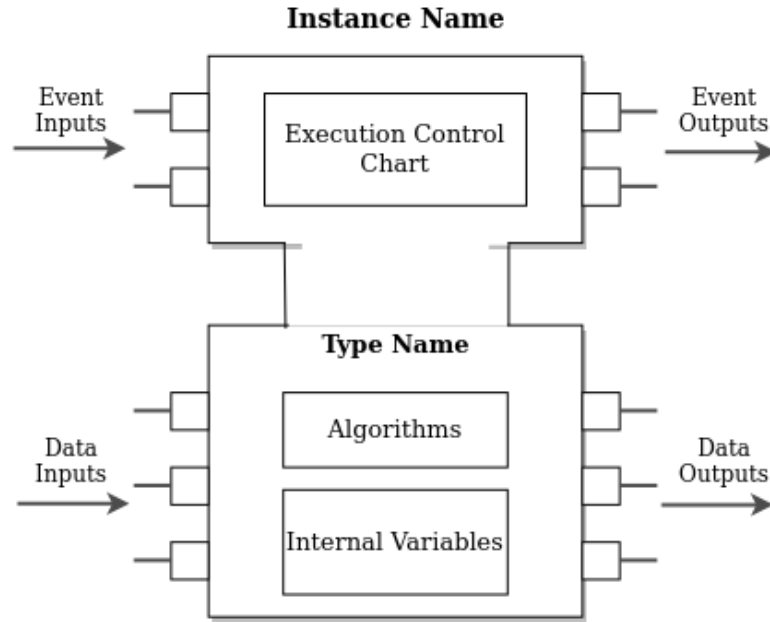


Figure 2.3: Generic interface of an *International Electrotechnical Commission 61499 Function Block*.

4DIAC-IDE, which provides a GUI⁴⁰ based in a desktop program with the traditional Eclipse IDE look. It allows for the most thorough drawing and deployment of FB Pipelines and provides an interface with the FORTE RTE. FBDK, on the other hand, is a nice alternative that is free to use for research purposes. It includes the FBRT⁴¹. Finally, *NxtStudio* is provided by *NxtControl*, a Schneider Electric company, and is the most used commercial tool.

Regarding RTEs, they enable the design and development of distributed architectures by utilizing the IEC 61499 standard, which is heavily influenced by Edge/Fog technology. FORTE, FBRT and FBDK RTE [61] are a few examples. FORTE is likely the most used since it is a portable implementation written in an object-oriented language (C++) that is ideal for both online reconfiguration, execution, and monitoring of FBs supported by the IEC 61499 standard. However, there are significant drawbacks, such as local language support and third-party interface with other systems. These are some of the features tackled by the DINASORE [62].

2.3.2 DINASORE

The primary purpose of the DINASORE framework [62, 9] is to act as a doorway to the incorporation of advanced analytics and ML⁴² into CPPSs based on the IEC 61499 standard. By utilizing the 4DIAC as IDE and FORTE as RTE, DINASORE enables simple construction of FBs and FB Pipelines using the *Python* coding language. Recent trends reveal that *Python* is one of the most

⁴⁰Graphical User Interface

⁴¹Function Block Run-Time Environments

⁴²Machine Learning

commonly used programming languages for constructing **ML** applications since it supports multiple frameworks for data science. The aim behind adopting *Python* in **DINASORE** is to allow the most recent advancements in **ML** to be integrated at the Edge layer with current shop-floor equipment. Because **DINASORE** executes at the Edge level, field communication methods for **FB** and **M2M** communication are required. So, while utilizing **FBs**, **DINASORE** also supports **OPC UA**, allowing any external data to be automatically given in an **OPC UA** server for further system integration. Figure 2.4 depicts the general **DINASORE** architecture. This includes Edge devices that run **DINASORE** instances and the 4DIAC, which is used by a user to create **FB** Pipelines and deploy the new configuration to all devices in the **CPPS**.

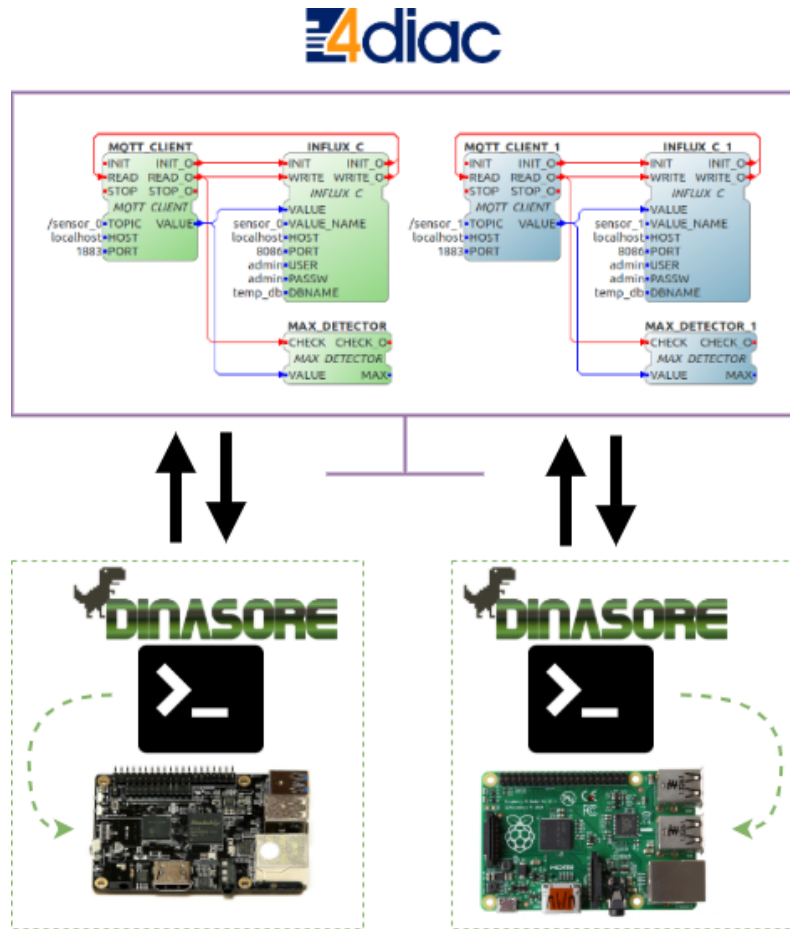


Figure 2.4: General *Dynamic Intelligent Architecture for Software and Modular REconfiguration* architecture [9].

In this regard, **DINASORE** uses the **4DIAC-IDE** as **IDE** for the construction of **FB** Pipelines and widespread deployment in Edge devices. Based on **FORTE**, **DINASORE** runs as a **RTE** in each Edge device. The primary benefits are the support for **FB** development in *Python*, allowing the latest developments in **ML** to integrate at the Edge level, and third-party integration utilizing the **OPC UA** communication protocol, enabling **M2M** communication at the Edge level. The basic development process using **DINASORE** consists of four distinct steps, as shown in Figure 2.5:

1. **FB Creation:** Build a new **FB**, to be used in **DINASORE**, involves the creation of two different files. On the one hand, a **XML**⁴³ file, which encodes the **FB** metadata in the form of **XML** tags. On the other hand, a *Python* file is used to implement the functionalities. Both these files can be created with popular text editors, meaning that the *4DIAC-IDE* is not required at this stage.
2. **FB Pipeline Creation:** When all the **FBs** to be used are available, the user can design the system's configuration in the *4DIAC-IDE*, by drag&drop **FBs** and linking them. Linking different **FBs** consists in connecting *Event Outputs* and *Data Outputs* of the origin **FBs** to *Event Inputs* and *Data Inputs* of the destination **FBs**. The resulting **FB** network is known also as **FB Pipeline**.
3. **FB to Device Mapping:** Now that the **FB Pipeline** is created in the *4DIAC-IDE*, the user must map each **FB** of the Pipeline to the Edge device that will execute it. All networked Edge devices should also be identified in *4DIAC-IDE*, along with their **IP** addresses.
4. **Solution Deployment:** The final step consists in deploying the solution achieved (Pipeline & mapping to Edge devices) to the **CPPS**.

2.3.3 Function Block Pipeline Metrics

In this section it is presented adapted measures from software, proposed earlier by Zhabelova and Vyatkin [63], to assess overall **FB** Pipelines. The measures proposed are: i) High-level design complexity metrics, such as $S(i)$ ⁴⁴, $DC(i)$ ⁴⁵, and $C(i)$ ⁴⁶; ii) M_H ⁴⁷; iii) M_m ⁴⁸; and iv) MI ⁴⁹.

(i) High level design complexity metrics:

- $S(i)$: Reflects the coupling of the **FB** i to the rest of the system. The metric considers the number of f_{out} ⁵⁰ in a **FB** i , and is given by Eq. 2.1.

$$S(i) = f_{out}^2(i) \quad (2.1)$$

- $DC(i)$: Shows efficiency of data utilization and information flow. The metric considers the ratio between the number of NI ⁵¹ and NO ⁵² in a **FB** i with the number of f_{out} , and is given by Eq. 2.2.

⁴³Extensible Markup Language

⁴⁴Function Block Structural Complexity

⁴⁵Function Block Data Complexity

⁴⁶Function Block System Complexity

⁴⁷Halstead's Metrics

⁴⁸M McCabe's Metric

⁴⁹Maintainability Index

⁵⁰Output Events

⁵¹Data Inputs

⁵²Data Outputs

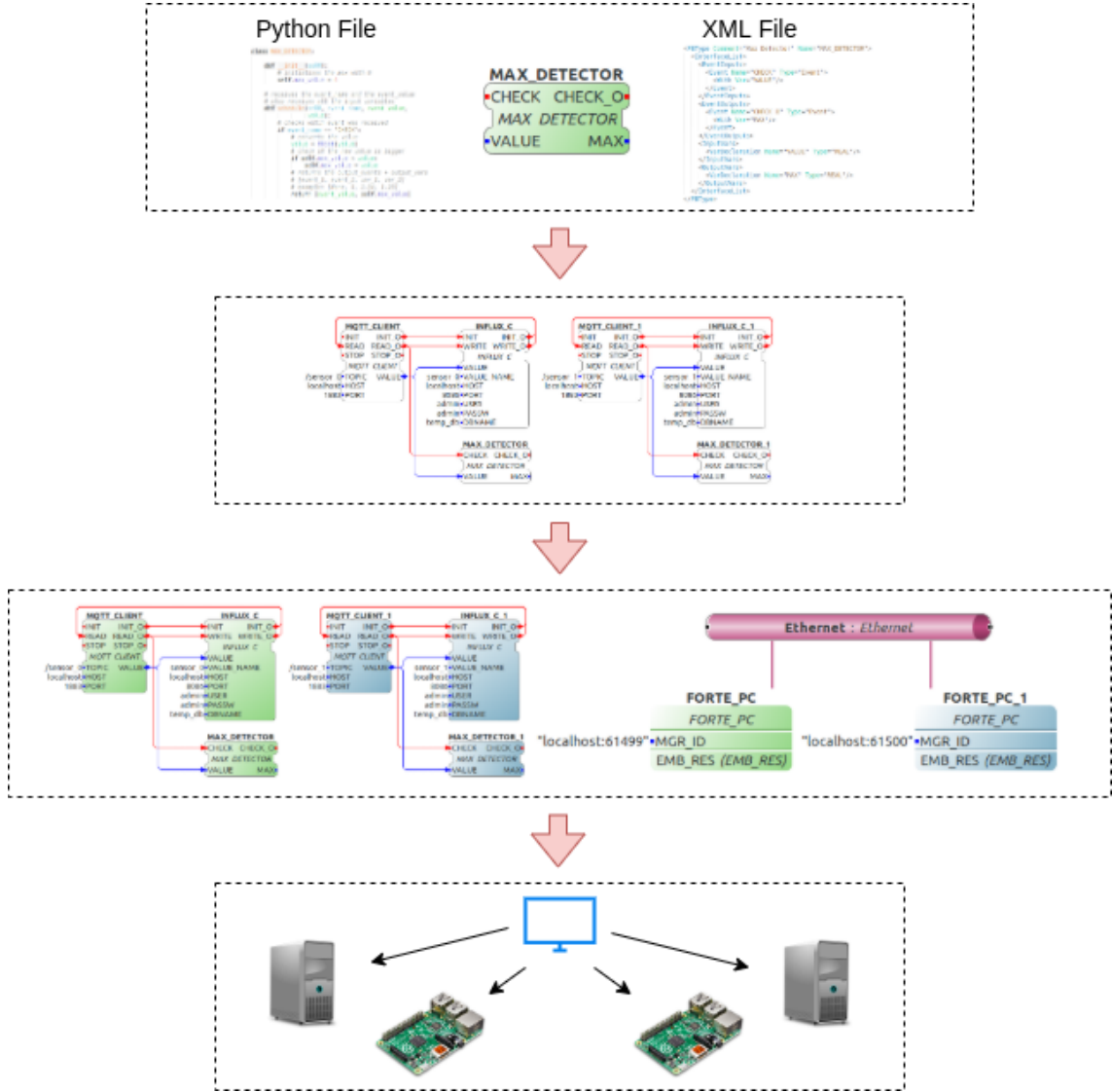


Figure 2.5: General Dynamic INtelligent Architecture for Software and MOdular REconfiguration development process [9].

$$DC(i) = \frac{[NI + NO]}{f_{out}^2(i) + 1} \quad (2.2)$$

- $C(i)$: As data and structural complexity increase, system complexity grows, and modularity decreases. This metric considers both the structural and data complexity in a unique measure and is given by Eq. 2.3.

$$C(i) = S(i) + DC(i) \quad (2.3)$$

Note that S^* ⁵³, DC^* ⁵⁴ and C^* ⁵⁵ represent the aggregated values for the entire FB Pipeline, defined respectively as the sums of all $S(i)$, $DC(i)$, and $C(i)$ for each FB instance in the Pipeline, as represented in Eq. 2.4.

$$\begin{aligned} S^* &= \sum_{i=1}^n S(i) \\ DC^* &= \sum_{i=1}^n DC(i) \\ C^* &= \sum_{i=1}^n C(i) \end{aligned} \quad (2.4)$$

- (ii) M_H : is an indicator of volume and entropy measure, by taking into account the number of operators and operands of both FBs's ECCs⁵⁶ and respective source code. *Halstead's Metrics* are: N ⁵⁷, n ⁵⁸, $\hat{N}$ ⁵⁹, PR ⁶⁰, V ⁶¹, D ⁶² and E ⁶³. They are given by Eq. 2.5.

$$\begin{aligned} N &= N_1 + N_2 \\ n &= n_1 + n_2 \\ \hat{N} &= n_1 \log_2(n_1) + n_2 \log_2(n_2) \\ PR &= \frac{\hat{N}}{N} \\ V &= N \log_2(n) \\ D &= \frac{n_1}{2} \times \frac{N_2}{n_2} \\ E &= D \times V \end{aligned} \quad (2.5)$$

Note that N_1 and N_2 correspond to the total number of operators and operands, respectively. On the other hand, n_1 and n_2 correspond to the total number of distinct operators and operands.

- (iii) M_m : McCabe's cyclomatic complexity indicates the number of paths in a control flow graph. It is a different way of representing control and data flow complexity using graph-based metrics. It is given by Eq. 2.6.

⁵³Structural Complexity

⁵⁴Data Complexity

⁵⁵System Complexity

⁵⁶Execution Control Charts

⁵⁷Program Length

⁵⁸Program Vocabulary

⁵⁹Estimated Length

⁶⁰Purity Ratio

⁶¹Program Volume

⁶²Difficulty

⁶³Program Effort

$$M_m = \sum_{i=1}^n M_m(Alg_i) + M_m(ECC) \quad (2.6)$$

$M_m(Alg_i)$ ⁶⁴ represents the cyclomatic complexity of the FB i (algorithm source code), by counting the number of *if/else* statements and loops, such as *for* and *while*. On the other hand, $M_m(ECC)$ ⁶⁵ represents the cyclomatic complexity of the FB i (internal ECC), by counting both the number of edges (transitions in the ECC) and nodes (states in the ECC). They are given by Eq. 2.7.

$$\begin{aligned} M_m(Alg_i) &= N_{ifs} + N_{loops} + 1 \\ M_m(ECC) &= N_{edges} - N_{nodes} + 2 \end{aligned} \quad (2.7)$$

- (iv) **MI**: Represents the degree to which a FB Pipeline is open to change after being deployed in the Edge layer. One of the most popular metrics to measure maintainability takes into account LOC ⁶⁶, as represented in Eq. 2.8, Halstead's measures of V and M_m . It is given by Eq. 2.9. All these metrics consist in the sum of both analyses for the FB internal ECC and the FB algorithms source code. Note that $LOC(Alg_i)$ ⁶⁷ is simply the number of lines in the source code. On the other hand, $LOC(ECC)$ ⁶⁸ is the sum of all states and transitions in the FB ECC.

$$\begin{aligned} LOC &= \sum_{i=1}^n LOC(Alg_i) + LOC(ECC) \\ LOC(ECC) &= N_{states} + N_{transitions} \end{aligned} \quad (2.8)$$

$$MI = 171 - 5.2\ln(V) - 0.23M_m - 16.2\ln(LOC) \quad (2.9)$$

2.4 Industrial Cyber-Security

Over time, CPPSs have become increasingly vulnerable to security flaws, especially with the recent inclusion of IoT principles. Similar to other unique technologies that have evolved in the past, developers consider the inherent security qualities of the technology after it has been produced and is ready for deployment. Security in CPPSs has been highlighted as a critical research problem to accelerate IIoT development towards a solid and trustworthy technology, culminating in community acceptance. Until recently, CPPSs were of a dedicated nature, which means they were primarily concerned with interactions in a relatively confined environment. The paradigm, however, is shifting toward a more dispersed, complicated, and open system. Because of this

⁶⁴Function Block Source Code Cyclomatic Complexity

⁶⁵Function Block Execution Control Chart Cyclomatic Complexity

⁶⁶Lines of Code

⁶⁷Lines of Source Code

⁶⁸Execution Control Chart States and Transitions

openness, hackers and other cybercriminals have a larger potential attack surface. Unauthorized individuals can easily acquire remote access to the system without raising suspicion.

In terms of business operations, adequate security measures will enable reliability in IIoT applications. According to Roman *et al.* [64], IoT must have solid security foundations implemented in all aspects at all phases. Following that, various IIoT security aspects are presented:

- **Protocol and Network Security** – *Protocol and Network Security* pertain to the protection of communication channels between limited devices as well as those devices and external systems. Cryptographic techniques, key-management systems, and secure protocols are typically used to protect the network infrastructure.
- **Data Privacy** – It is expected that an IoT application can create massive volumes of data. Because different levels of information are becoming available, attackers are compelled to monitor and track this data without users' knowledge. Data security may be done on two levels. First, because some devices may lack the capabilities to manage cryptographic algorithms, privacy techniques should be included in the architecture of the device (built-in by design). Second, the identity of entities that provide data services and management of users' information should be known transparently.
- **Identity Management** – *Identity Management* entails having a platform that maintains not just a device's identity, which must be able to prove it, but also anonymization in situations where the device's identity should not be exposed. Authentication and authorization procedures are often used to identify Edge devices and verify their identity.
- **Trust and Governance** – *Trust* has two different meanings in IoT. Trust is expressed through methods that, first, diminish the uncertainty of Edge devices when they interact with one another and, second, indicate how users feel while dealing with an IoT application. *Governance* is the most effective technique to build confidence since it symbolizes the implementation of measures that assure security continuity.
- **Fault Tolerance** – *Fault Tolerance* is required in IoT setups to ensure service dependability. *Fault Tolerance* may be achieved with IDSs, which monitor the condition of the system by analyzing data and, in the event of an anomaly, can protect the system against it and restore properties from any harm. The emphasis of the following sections is in IDSs.

Industrial cyber-security has become an active research topic since businesses must protect their systems containing personal and sensitive information from cyber-attacks at all costs to continue offering revenue-generating services to clients uninterrupted. A network attack is a hostile attempt to exploit a computer or network's vulnerability to gain access to or damage the system's security. An attacker or invader is someone who launches or attempts to launch an assault or attack into a system. Intruders are classified into two types: external and internal. External intruders are unauthorized users of the systems or machines they attack. In contrast, internal attackers have access but do not have root or superuser privileges. Internal intruders can also be classified as masquerade intruders, i.e., they enter as another person with legitimate access to critical data.

Traditional network security solutions, such as user authentication, encryption, and firewalls, have been used as the first line of defense to ensure network security. Still, they are insufficient to prevent all attacks in today's highly networked world. Intrusion detection is commonly used as a second perimeter wall to increase network security and has become crucial in today's manufacturing systems.

2.4.1 Intrusion Detection Systems

An **IDS** is a software program that monitors harmful behavior. An **IDS**'s purpose is to identify intrusions by gathering and analyzing data. These systems detect evidence about the intrusion's existence rather than the intrusion itself, either after it has occurred or, preferably, while it is in progress. An **IDS** accuracy is dependent on the data that depicts the target system operations, which must be reliable, comprehensive, and trustworthy. The quantity of data acquired by the **IDS** from the target activity is a trade-off between overhead and efficiency because gathering information is costly and gathering the correct data is critical.

IDSs are regarded as the second line of defense to identify known and undiscovered cyber attacks utilizing specific methodologies. On the other hand, intrusion prevention and baseline security measures form the initial line of protection to stop and prevent intrusions. If the attack has already happened, it signifies that the initial line of defense has failed. Rossi [65] argues that organizations should change their attention from prevention to detection. Data breaches are unavoidable as attackers get more devious in their strategies, growing increasingly sophisticated. As a result, businesses should expect to be targeted and should be prepared for it. It is preferable to detect an intrusion while occurring and launch an immediate response than attempting to prevent it from happening. On the other hand, Brennan [66] argues that, while prevention is difficult, it should not be overlooked. Both prevention and detection are essential in a complete security solution, and they must work together.

An **IDS** is an essential component of any industrial system's network security. The following factors contribute to the development of an **IDS**:

- The majority of existing systems have security flaws that expose them to intrusions. It is difficult to find and fix all of these flaws [67], and extensive systems, in particular, are prone to issues that hostile users might exploit.
- Prevention methods are insufficient. It is virtually impossible to design a perfectly secure system [67]. When an intrusion has occurred that cannot be stopped by current security procedures, an **IDS** is employed.
- Because insider attacks are usually carried out by authorized personnel, even the most secure systems may be penetrated. Furthermore, many businesses consider that internal threats are significantly more harmful than external ones.
- New intrusions are found regularly. As a result, security solutions must be updated or built to defend systems from new threats. This is why intrusion detection is such an active research topic.

A generic IDS has three primary functions: collecting data, analyzing data, and responding to the analysis. Data is collected on the host level (HIDS⁶⁹) by retaining log files containing the system's actions or on the network level (NIDS) by recording network traffic. The acquired data is subsequently evaluated using statistical methods, pattern matching, or data mining techniques. Suppose an intrusion is suspected as a result of the data analysis phase. In that case, a response is triggered according to the level of impact needed to mitigate the intrusion [68]. This reaction might be passive, such as spreading an alert, or more active, such as launching counterattacks. Figure 2.6 depicts the critical components of a generic IDS.

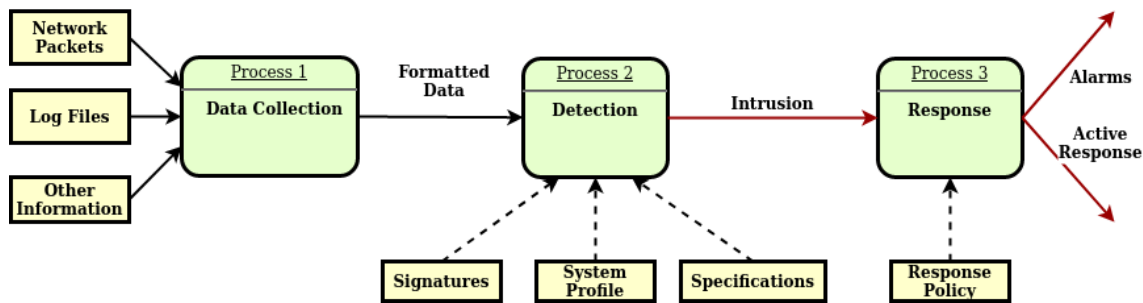


Figure 2.6: Main Intrusion Detection System components [10].

2.4.1.1 IDS Data Collection

An IDS can be categorized by the source of the audit data, namely:

- *Network-based Intrusion Detection System:* The NIDS [69] listens for and evaluates network data transmitted within the system. In general, intrusions appear as abnormal patterns. Attacks undertaken by outside adversaries seeking unauthorized network access to steal information or damage the network are the primary source of these irregularities. Most networks are connected to the rest of the world through the Internet. All incoming packets or flows are examined by the NIDS for suspicious patterns. An NIDS inspects not only incoming/outgoing data but also offers vital information regarding intrusions from local traffic. Some attacks may even originate within a monitored network or network segment and may not be considered incoming traffic. Data for NIDSs can be of various granularities, such as network packets or network flows. Furthermore, the data is frequently multidimensional, comprising categorical and continuous quantitative features.
- *Host-based Intrusion Detection Systems:* The HIDS [70] is concerned with host data by analyzing system files, unsuccessful access attempts, and unusual memory allocations, as well as CPU⁷⁰ and I/O activities. A HIDS monitors and explores the internals of the Edge device rather than its external communication interfaces. It keeps track of all or parts of the Edge device's dynamic activity and status. HIDSs may detect internal activity such as which

⁶⁹Host-based Intrusion Detection Systems

⁷⁰Central Processing Unit

programs access which resources and attempts to get illegal access. Similarly, a **HIDS** may investigate a system's state and stored data, whether it's in **RAM**⁷¹, the file system, log files, or anywhere else. As a result, a **HIDS** is a system that checks to see whether anything or anybody, internal or external, has circumvented the security policies that the operating system seeks to impose.

- **Hybrid Intrusion Detection System:** This solution takes into account both **NIDS** and **HIDS** features by performing system log file checks on each host and centralized overall network traffic assessments. This way, it is possible to bring the best of both approaches into a unique solution. The other way around is also valid, i.e., analyzing income/outcome network data at the host level can be seen as a hybrid solution. The nature of the data analyzed makes the solution network-based. Still, the deployment location is typical in host-based detection.

2.4.1.2 IDS Detection

Despite being characterized based on the audit data, the detection analysis technique used in the **IDS** is also a criterion for further categorizing. The primary **IDS** detection techniques are signature-based, anomaly-based, specification-based, and reputation-based. In this Thesis we are interested in anomaly-based techniques.

Signature-based Intrusion Detection [71, 72] refers to systems capable of identifying known attacks by matching the data stream with pre-existing signatures. Signatures can be bytes conveyed through a network, a program's system call, or an application information flow based on predetermined attack descriptions. Signature detection, also known as misuse, supervised, or pattern-based detection, is distinguished by its high detection efficiency and low **FPR**⁷². For this reason, it is commonly used in the majority of commercial tools. However, it is confined to detecting only known attacks for which a signature has been specified. This is a limitation since invasions are often polymorphic, evolving into new threats regularly. As a result, new attackers' signatures must be regularly updated in the signature database, which requires a large amount of system memory.

On the other hand, *Anomaly-based Intrusion Detection* [73] is used to detect harmful behavior by comparing it to a predefined desired behavior (normal or baseline behavior). The premise is that attacks are outliers in the data collection, deviating from the expected behavior of a typical dataset. The system's standard behavior is defined by training it using live data or collecting truth data. As a result, anomaly-based intrusion detection creates models based only on typical behavior (system profiles). It uses these models to detect abnormalities in observed data (deviations from the models). The primary benefit is that previously undiscovered attacks may now be identified. However, these models are characterized by a significant **FPR**. It is difficult in dynamic systems to appropriately recognize atypical activity as an attack, i.e., they have difficulties adapting to continually changing normal behavior. Also, discovering boundaries between normal and abnormal behavior is an issue due to the deficiency of anomalous samples in the training phase. Additionally,

⁷¹Random Access Memory

⁷²False Positive Rate

the system becomes vulnerable in the training phase (in the case of non-live training). According to Butun *et al.* [74], and taking into account the nature of the processing involved in the behavioral model, an *A-IDS*⁷³ may be further characterized as:

- *Statistical-based A-IDS*: This technique (both parametric and non-parametric) evaluates events or network traffic using statistical algorithms to assess whether or not a piece of data corresponds to a given statistical model. When an event occurs during the anomaly detection process, an anomaly score is assigned to compare the currently observed profile and the previously trained statistical profile. This baseline profile is produced using historical data, i.e., while the system operates normally (without any attack/intrusion). Furthermore, although parametric procedures use the underlying distribution assumption, non-parametric approaches discover the underlying distribution independently. The anomaly score represents the degree of irregularity for the given occurrence. If it exceeds a particular threshold, the *IDS* will generate an alert.

Statistical processes, in general, are more resistant to incomplete and inaccurate training data and are better suited for dispersed and online deployment. Parametric techniques are unsuitable for datasets that do not fit a previously determined distribution. Non-parametric methods, on the other hand, do not have that constraint. However, they might be computationally resourced intensive to provide the flexibility for online deployment.

- *Knowledge-based A-IDS*: Also known as rule-based *A-IDS*, it relies on prior knowledge of system parameters under both normal and attacks operating situations. It is based on the construction of computer environment rules set by professionals to specify legitimate system activities. *Knowledge-based A-IDSs* identify and classify any anomalous data that violates these standards as an attack.

This strategy is often straightforward and quick, with no learning component. As a downside, the strategy will be useless if not applied to the precise circumstance stated by the rules, i.e., application situations that do not correlate to previously utilized information. Furthermore, the technique does not work well with incomplete or inaccurate data.

- *ML-based A-IDS*: This is a sub-field of *Hard Computing* that is centered on the creation of mathematical models that allow the occurrences under investigation to be categorized. The strategy needs accurate input data, employs a precise methodology, and provides precise output, resulting in a crisp system. Overall, depending on the learning methods utilized, *ML* approaches can be supervised or unsupervised. In supervised learning, the training data is labeled. Each input data point in the training set is labeled accordingly. On the other hand, unsupervised learning does not label the input, and the system learns by assessing data attributes to create the classifier.

⁷³Anomaly Intrusion Detection System

These algorithms often provide solid classification/detection performance and speed. However, they take considerable time to develop the relevant model, which is prohibited in real-time systems. These strategies, however, may be used in dispersed and online contexts, provided they have been modified appropriately. Furthermore, some techniques may work with incomplete data. However, because the learning mechanism for such a technique is sophisticated, its implementation in online applications may be complex. Furthermore, relying on an initial training stage may not be appropriate for every application situation in terms of gathering training data and leaving the system unsecured for training purposes.

- *Soft Computing-based A-IDS*: *Soft Computing* is a new discipline of computer science that tolerates imprecise and uncertain problems with partial truth to find an approximate, resilient, and low-cost optimal solution. In an imprecise situation, *Soft Computing* techniques address problems by imitating unique human capacity to reason and learn from mistakes. Multiple disciplines of *Soft Computing* examine various biological processes like animal swarming, genetic evolution, creature survival, and the human nervous system. Fuzzy Logic, [RST](#)⁷⁴, Genetic Algorithms, [ANNs](#)⁷⁵, [ACO](#)⁷⁶, [AIS](#)⁷⁷, and other topics related to *Evolutionary Computation* are examples.

These systems frequently recognize or categorize persistent features without feedback from the environment, perform well with partial data, and consume fewer resources. Furthermore, the learning process may be simplified because it employs an evolutionary approach (entities follow simple rules that, over time, evolve into complex behaviors). In contrast, traditional [AI](#)⁷⁸, such as [ML](#), employs a 'creationist' approach (the algorithm is created from the top and is already embedded with 'intelligence'). However, if a sufficient amount of real-world traffic data is not available, training the algorithms becomes highly challenging. Several techniques also have scalability difficulties.

Specification-based Intrusion Detection [[75](#), [76](#)] operates on the same idea as anomaly-based techniques. While anomaly-based approaches utilize data mining to extract elements of normal behavior from gathered data, specification-based techniques use formal models of a system's typical behavior that a human explicitly describes. These systems benefit from having a low [FNR](#)⁷⁹ and, unlike anomaly-based strategies, the system is immediately effective because no training phase is required. Finally, *Reputation-based Intrusion Detection* [[77](#)] is applicable in ad-hoc applications as well as massive networks. The purpose is to discover selfish or collaborating nodes to increase their reputation.

⁷⁴*Rough Set Theory*

⁷⁵*Artificial Neural Networks*

⁷⁶*Ant Colony Optimization*

⁷⁷*Artificial Immune Systems*

⁷⁸*Artificial Intelligence*

⁷⁹*False Negative Rate*

2.4.1.3 Detection Performance Evaluation

Detection performance evaluation primarily addresses difficulties linked to benchmarking and collecting datasets for objectively evaluating **IDS**s while utilizing crucial metrics for quality assessment. In **IDS**s, numerous evaluation metrics are employed. The following sections describe several critical evaluation metrics.

Accuracy Detection Acc^{80} is used to evaluate the performance of an **IDS** in terms of accuracy. It monitors detection and failure rates and the number of false alarms produced by the system. The accuracy metric is used to assess an **IDS** 's ability to detect an attack appropriately. Accuracy of an **IDS** may be measured using a variety of criteria [78, 79].

- *Sensitivity* and *Specificity*: If the **IDS** is employed in a binary classification problem, i.e., anomalous data is labeled as *True* while normal data is labeled as *False*, metrics like TPs^{81} , TNs^{82} , FPs^{83} and FNs^{84} can be used. In a 2-class classification scenario, they serve as the foundation for the *Confusion Matrix*. A detection, or TP , happens when the **IDS** successfully recognizes an anomalous instance. In contrast, a false alarm, or FP , occurs when the **IDS** incorrectly classify a normal instance as anomalous. A TN occurs when a normal instance is correctly classified as a legitimate action, while an FN , or failure to report, is when an anomaly is not detected.

Sensitivity in this scenario refers to the sensitivity of identifying anomalies correctly by computing the ratio of well-categorized anomalies (detections) to all existing anomalies, as indicated by Eq. (2.10). This is also referred to as TPR^{85} . The *Specificity*, also known as TNR^{86} , is the ratio of well-categorized normal behavior to all normal data, as expressed in Eq. (2.11).

$$TPR = \frac{TP}{TP + FN} \quad (2.10)$$

$$TNR = \frac{TN}{TN + FP} \quad (2.11)$$

On the other hand, there are also metrics related to both FPR and FNR , as represented in Eq. (2.12) and Eq. (2.13) respectively.

$$FPR = \frac{FP}{TN + FP} \quad (2.12)$$

⁸⁰Accuracy

⁸¹True Positives

⁸²True Negatives

⁸³False Positives

⁸⁴False Negatives

⁸⁵True Positive Rate

⁸⁶True Negative Rate

$$FNR = \frac{FN}{TN + TP} \quad (2.13)$$

- *Misclassification Rate*: When an IDS predicts a class, either normal or abnormal, the expected class varies from the actual class. This measure, which is defined in Eq. (2.14), helps estimate the possibility of a difference between real and anticipated categorization rates.

$$\frac{FN + FP}{TP + FN + FP + TN} \quad (2.14)$$

On the other hand, there is also a metric related to *Acc*, as represented in Eq. (2.15).

$$Acc = \frac{TN + TP}{TP + FN + FP + TN} \quad (2.15)$$

- *Precision, Recall and F-measure*: For an n-class intrusion classification problem, *F1*⁸⁷ is the most preferred metric for accuracy. It integrates *Precision* and *Recall* into a single statistic, i.e., it is the harmonic mean of accuracy and *Recall*. *Precision* in the context of network anomaly detection assesses an IDS efficacy in recognizing abnormal as well as typical instances, and is defined in Eq. (2.16). *Precision* and *Recall* are frequently inversely related, and there is usually a trade-off between these two ratios. Note that *Recall* is the same as *Sensitivity* in binary class problems. *F1* is defined in Eq. (2.17).

$$Precision = \frac{TP}{TP + FP} \quad (2.16)$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.17)$$

$$F1 = \frac{2TP}{2TP + FP + FN}$$

- *ROC*⁸⁸: A *ROC* curve is a plot of the *TPR* in function of the *FPR*. Because of this, the *ROC* curve is a useful XY-plane representation of the sensitivity of detection versus false positive probability. In these situations, the *AUC-ROC*⁸⁹ is calculated. The *AUC-ROC* is a numerical representation of the performance of the binary classifier, in terms of how well the IDS can distinguish between two classes. It ranges in value from 0 to 1. A model whose predictions are 100% wrong has an *AUC-ROC* of 0.0, while one whose predictions are 100% correct has an *AUC-ROC* of 1.0. An *AUC-ROC* of 0.5 means that the classification is similar to randomly choosing one of the two classes.

⁸⁷*F-measure*

⁸⁸*Receiver Operating Characteristic*

⁸⁹*Area Under the Receiver Operating Characteristic Curve*

Performance A comprehensive review of an **IDS** performance is a crucial responsibility. It implies a host of difficulties that go beyond the **IDS**. The hardware platform, the operating system, and even the **IDS** deployment [80] are examples of such challenges. The ability of an **IDS** to manage traffic on a high-speed network with minimum packet loss while operating in real-time is the most critical performance criterion. Particularly considering **NIDS** problems, packet sizes might vary in real-world network traffic. As a result, the ability of an **NIDS** to handle packets of any size determines its efficacy. In addition to processing speed, **CPU** and memory consumption may be utilized to evaluate the performance of **IDSs**. These are frequently used as indirect metrics to account for an intrusion detection technique's temporal and geographical complexities.

Completeness This evaluation criterion aims to evaluate an **IDS** against a complete set of known attacks. It checks the whole space of all vulnerabilities and attacks to see if a specific **IDS** covers them all. However, complete information concerning attacks or abuses of privileges is required to implement this method successfully, which may not be feasible. A **IDS** capacity is total if it can manage all known vulnerabilities or attacks efficiently. Such a comprehensive system must have some way of coping with unforeseen threats.

Timeliness An **IDS** should finish its analysis, detection, and alert generation as soon as feasible to notify the system administrator or security manager. This person may then take necessary action before the system suffers significant harm. It may prevent the attacker from gaining access to the audit source or the **IDS** itself. In the **NIDS** problem, for example, there will be a notable lag between the occurrence of the assault and the development of the detection system's reaction. This lag is crucial in the assessment of the **NIDS** (the smaller the value, the better the **NIDS** timeliness performance [81]).

2.4.2 Anomaly Network Detection

Algorithms that examine data to discover abnormalities or outliers are called anomaly detection algorithms. A divergence from typical system behavior attributed to several causes is an anomaly. It is defined as an unallowable deviation from the approved or standard operating condition of at least one system characteristic attribute (feature). Suppose no recovery decisions are taken when confronted with an anomaly. In that case, it may result in malfunctions or failures, which are long-term disturbances in a system's ability or performance to complete an essential function within defined operational conditions. An anomaly detection mechanism aims to investigate, grasp, and explain system behavior and recognize or differentiate anomalous event occurrences from routine occurrences, such as malicious attempts. As a result, the anomaly detection problem may be seen as a classification challenge from the perspective of **ML**.

However, it is important to understand the differences between (binary) classification and anomaly detection. Although related, they are two distinct **ML** problems. Classification is a type of supervised learning used to classify observations into two (binary) or more classes. Its target is always categorical, and the class labels are expected to be balanced so that all classes have almost

equal importance. On the other hand, anomaly detection aims at distinguishing between normal and abnormal observations. Anomalous observations are often rare ones that do not follow the expected pattern of other observations. This also implies that the dataset is imbalanced. This means that supervised anomaly detection can be seen as a binary classification problem. The only feature that keeps them apart is whether the dataset is imbalanced or not. The other anomaly detection approaches (unsupervised or even semi-supervised) cannot be related to classification problems since they do not consider labeled classes.

Anomalies are classified into two types: performance-related abnormalities and security-related anomalies. Performance-related abnormalities are produced by flaws in the system's design, implementation, and administration. Malicious behavior, on the other hand, causes security-related irregularities. Considering the IDS domain as previously seen, security-related anomalies are the focus of Anomaly-based IDSs. Anomalies, according to Chandola *et al.* [82], can manifest themselves in three ways:

- Point anomaly: This is the most fundamental circumstance in which a single data instance differs from the rest of the data. For example, if a data instance's value is bigger than the sum of all other data instances' values.
- Contextual anomaly: A data object that is inappropriate in its context. Depending on the scenario, the concept of context changes. A reported average yearly temperature of 9°C in the Amazon rain forest, for example, might be termed a contextual anomaly since it is abnormal in the context in which it is recorded (here, the context is longitude and latitude). However, it would not be anomalous in other contexts (e.g., Europe).
- Collective anomaly: happens when a linked data group deviates from the whole dataset. For example, if a periodic behavior that repeats for every constant time interval abruptly changes for a single period, neither that period nor any point in the context deviating from the sequence is anomalous. Still, the entire period behaves inconsistently compared to surrounding data points, despite possibly not having any significantly high values or high rate of change.

According to Bhattacharyya and Kalita [11], an A-NIDS⁹⁰ must identify and report unusual behavioral patterns in a network in real-time or near real-time. The following are the primary requirements for a network intrusion detection approach or system based on anomalies: i) It should detect network abnormalities with high accuracy and little or no prior knowledge of the target system's typical operations. It should also be able to learn the predicted behavior of the system via observations; ii) A technique should reliably detect harmful activities in various network contexts while generating the fewest false alarms; iii) The number of input parameters should be kept to a minimum, and their effect should be modest; iv) A technique or system should be capable of recognizing not just timed or bursty strikes, but also any unusual class or carefully launched attacks.; v) It should be capable of identifying novel abnormal patterns in addition to detecting all known threats.

⁹⁰Anomaly Network Intrusion Detection System

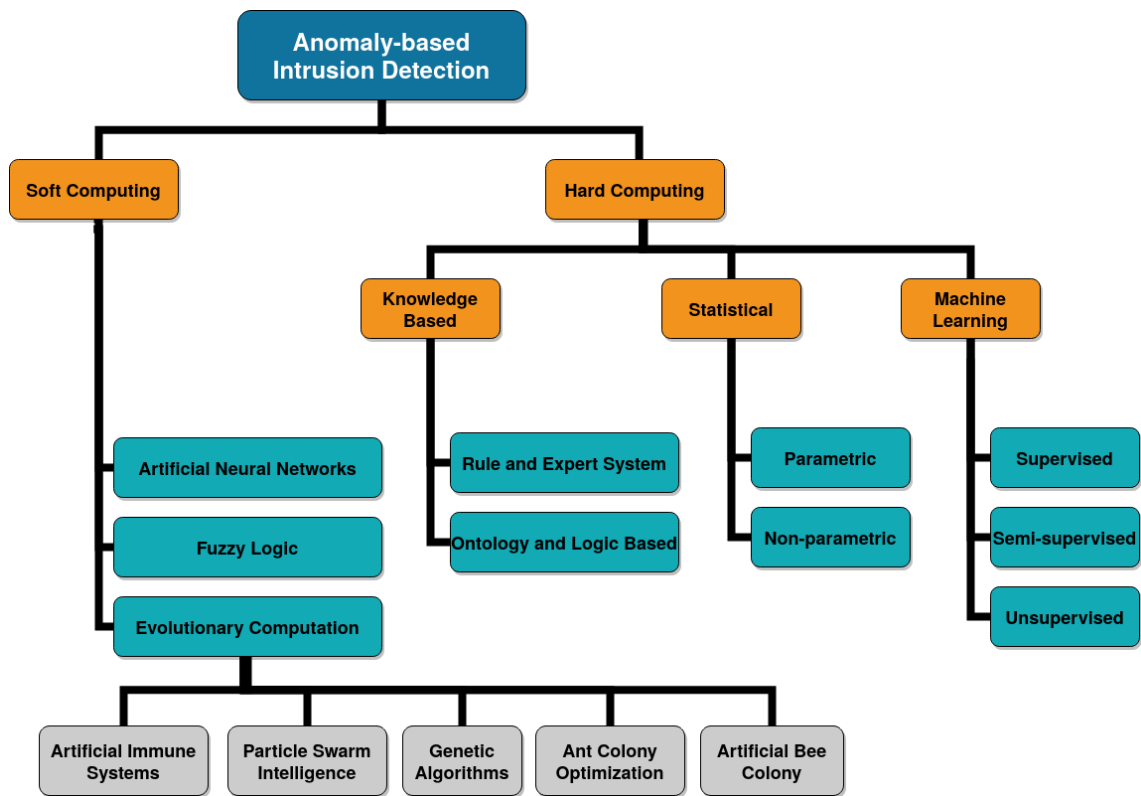


Figure 2.7: *Anomaly Intrusion Detection System* taxonomy [11].

A large number of methods have been introduced for A-NIDSs. Figure 2.7 represents a taxonomy for the types of anomaly detection methods [11], especially considering network-based approaches.

2.4.3 Networks and Anomalies

A solid grasp of core network ideas and how anomalies might occur in networks is essential for approaching anomaly network detection. A network is a complex interacting system made up of several separate components. Two or more computer systems are connected if they can transfer or receive data across a shared and accessible medium. Individual entity activity contributes to the ensemble behavior of the broader network. Computers, devices, machines, servers, and other network nodes are wired or wirelessly linked to exchange data.

Most networks are organized in layers to reduce design complexity. The top layers develop dependencies of the layer below them. Each layer is intended to deliver a defined set of services to the layer above it while concealing implementation details. A protocol is a set of rules that control how layers and devices interact. A service is a series of operations provided by a layer to its immediate above layer. A service can also interface between two successive or adjacent layers. The bottom layer serves as the service provider, and the upper layer serves as the service consumer. A protocol is a set of rules that defines the meaning and format of packets or messages transmitted between entities in a layer. Peer entities are entities that are on the same tier. The entities use

protocols to implement services. Entities may change the protocol as long as they comply with the rules.

So, based on the layered architecture of networks, there are well-known network architectures or reference models, such as the [OSI](#)⁹¹ and [TCP/IP](#). [ISO](#)⁹² introduced the [OSI](#) model as a reference model for computer protocol architecture and a framework for developing protocol standards. This model has seven layers:

- *Application Layer*: This layer allows users to connect to the [OSI](#) environment and distributed information services.
- *Presentation Layer*: This layer shields application processes from changes in data representation (syntax).
- *Session Layer*: It offers the control mechanism for application communication. This layer's primary duties are establishing, administering, and closing connections (sessions) between collaborating applications.
- *Transport Layer*: This layer enables the secure and transparent flow of data between two endpoints. End-to-end error recovery and flow control are also supported.
- *Network Layer*: This layer isolates the top levels from the data transmission and switching technologies needed to connect systems. It is also in charge of connection installation, maintenance, and termination.
- *Data link Layer*: This layer is in charge of sending information across the physical link in a reliable manner. It transports blocks (frames) while providing the required synchronization, error correction, and flow control.
- *Physical Layer*: This layer is in charge of sending a stream of unstructured bits via the physical channel. To access the physical media, it must cope with mechanical, electrical, functional, and procedural difficulties.

On the other hand, the [TCP/IP](#) is a cost-effective version of the [OSI](#), since it was introduced with only four layers. The network community has widely accepted the model, and it is represented in Figure 2.8.

The *Physical* layer transfers frame bits from one node to the next. It employs protocols associated with the physical transmission channel of the *Data Link*, such as Ethernet, Bluetooth, Wi-Fi, and so on. The *Network* layer transports datagrams across gateways or routers to forward packets in the network. The [IP](#) protocol, which is used by routers and end systems, is supported by this layer. The *Transport* layer transmits communications between the *Application* layer and the *Transport* layer (between client and server). It employed protocols like [TCP](#) and [UDP](#) to do this. Finally, the *Application* layer is in charge of supporting network applications that use relevant protocols, such as [HTTP](#).

⁹¹Open Systems Interconnection

⁹²The International Organization for Standardization

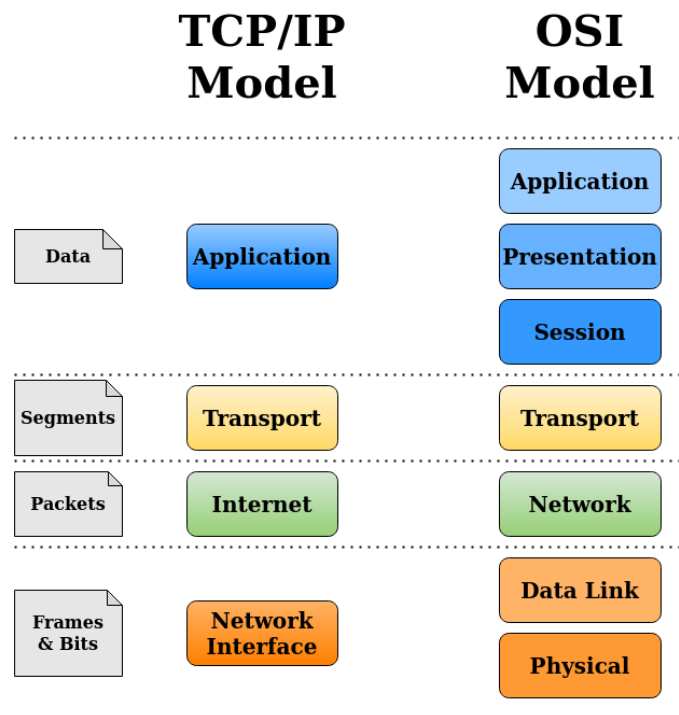


Figure 2.8: Network reference models.

Communication and networking are crucial elements for system integration in industrial scenarios. It essentially consists of transferring structured information over a network via a communication protocol. This protocol is generally defined in the *Application* layer. When it comes to industrial applications, the communication technology, and protocols employed must consider important characteristics considering timing and synchronization constraints. These characteristics are resilience and speed of transmission, bandwidth, and signal strength. Furthermore, because the network requirements and constraints at the process and control levels differ from those at the operations and monitoring levels, both field networks (lower level) and IP networks (high level) are employed. Finally, because there is both vertical and horizontal data integration, communication is more than sensor data transfer between Edge devices and a gateway. It is also a bi-directional communication between devices, which is commonly referred to as M2M communication. While industrial protocols are directed towards control and monitoring systems, IP protocols applied to industry focus on system integration and interoperability, according to the IoT principles (connect every device by integrating them with the Internet). A well-known IP communication protocol developed for industrial contexts is the OPC UA.

2.4.3.1 OPC UA

OPC UA is an industrial automation protocol that combines the capability of the *OPC Classic* specification into a single modular framework, whose functionalities are being standardized as the IEC 62541 norm. It is extensively used in industrial communications because it provides

safe data transmission and multi-platform connectivity. This implies that various devices might have an [OPC UA](#) client and/or server to communicate data [83]. The [OPC UA](#) specification is a multi-part specification divided into three sections, namely the Core, Access Type, and Utility specification components, as shown in Figure 2.9.

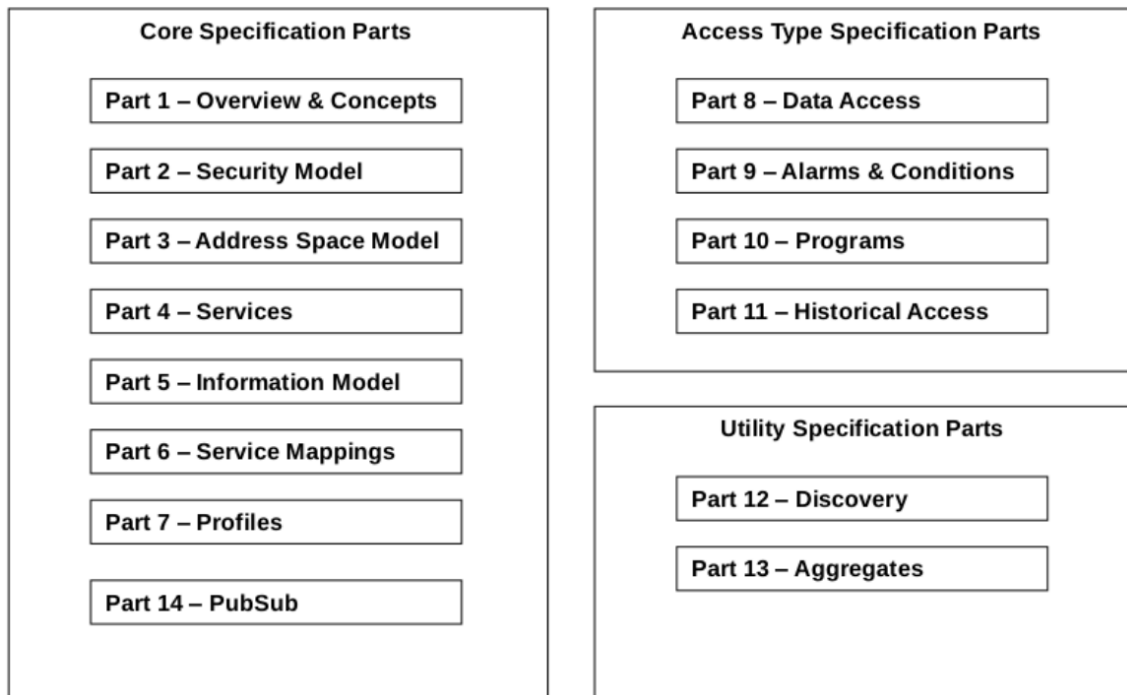


Figure 2.9: *OPC Unified Architecture* specifications [12].

Part 1 through Part 7, plus Part 14, consists of the core capabilities that describe how data is structured and exposed and the services used to access and manage it. Access type capabilities range from Part 8 to Part 11, specifying how fundamental capabilities and data models access various data types. Finally, utility capabilities, including Parts 12 and 13, explain discovery processes and data aggregation methods.

Regarding *Part 4 - Services*, [OPC UA](#) follows the design paradigm of [SOA](#), by defining generic services sets, such as: i) discovery between Clients and Server's endpoints; ii) secure communication channels; iii) application-layer connection in the context of a session; iv) node management in the address space; v) explore and filter the structure of the address space; vi) readable/writable node's attributes; vii) interfaces for invoking methods attached to an object; viii) monitored attributes or events and subscription of messages for monitored items. Figure 2.9 illustrates the [OPC UA](#) service discovery mechanism.

In this form of architecture, service discovery is accomplished by using a local server. All other servers register the services that they support on this server. When the local server for service discovery receives a request from a client, it examines whether servers on the network support that service and gives the client the information to connect to the same server. The server has two components: one that returns to consumers the information needed to establish a connection and

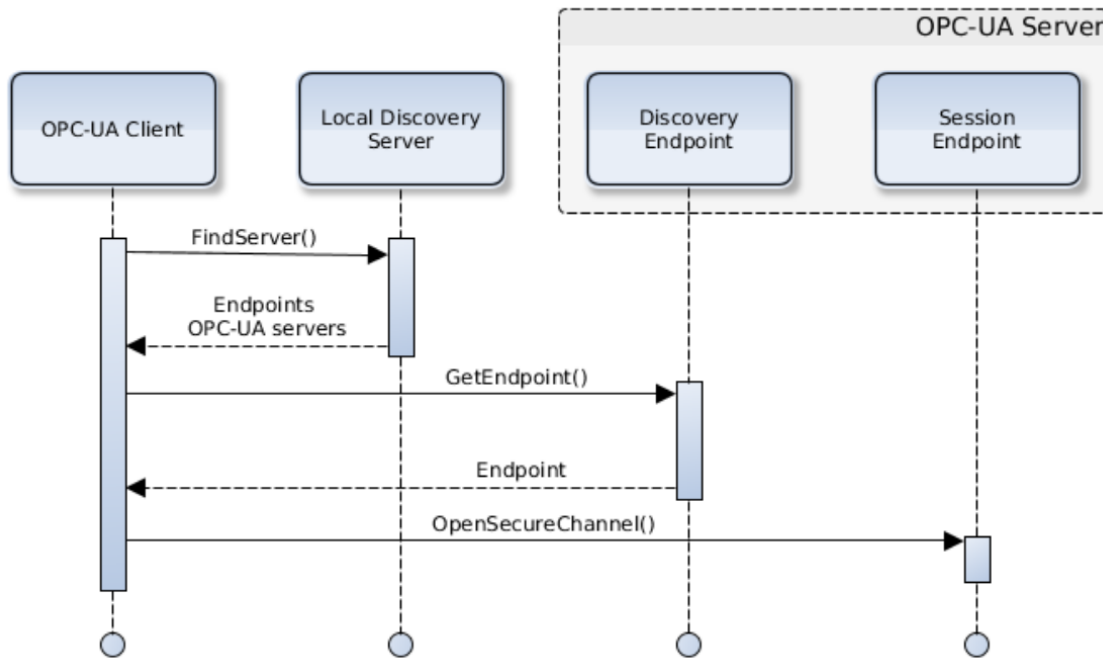


Figure 2.10: Example of the *OPC Unified Architecture* service discovery.

initiate a communication, and another that sends commands (read/write). After establishing a secure communication channel, both the client and the server can utilize various services such as item monitoring and subscription, method invocation, read/write node characteristics, etc.

Everything in **OPC UA** is represented as a *Node*, which is made up of *Attributes* (properties with data values that characterize the *Nodes*) and *References* (relationships with other *Nodes*). Each *Attribute* contains an integer id, a name, a description, a data type, and an optional/mandatory indication. Clients can get *Attribute* values by utilizing the **OPC UA** services. A communication stack, which supports binary and **XML** formats, is utilized on both the client and server sides to encode and decode message requests and answers. Regarding the transport layer, the **OPC UA** protocol may employ both **TCP** and **HTTP**, with the **TCP** format being the most often used since it achieves great performance and has lower computational costs when it comes to coding/decoding.

2.4.4 Network Attacks

As stated before, security-related anomalies are data instances divergent from the system's expected behavior caused by attacks or intrusions. An attack or intrusion consists of a set of actions that jeopardizes the security of a network or computer system. According to Bhattacharyya and Kalita [11], attacks can be classified into seven categories based on their implementation process:

- *Infection*: This type of attack attempts to infect the target system by interfering with it or by installing malicious files. Known attacks belonging to this category are *Viruses*, *Worms* and *Trojans*.

- *Exploding*: This type of attack attempts to overload the target system with flaws, e.g., *Buffer Overflow* or *Message Spoofing*.
- *Probe*: This category's goal is to collect information on the target system using scanning tools. The most significant attacks in this category include coordinated probes or scans, such as *Sniffing*, *Port Mapping*, and *Security Scanning*.
- *Cheat*: Attempts to use a false identity, e.g., *IP Spoofing*, *MAC Spoofing*, *DNS Spoofing* and *Session Hijacking*.
- *Traverse*: Attempts to breach a victim system using a simple match against all potential keys are common instances of this category. Examples are *Brute Force*, *Dictionary Attacks* and *Doorknob Attacks*.
- *Concurrency*: This type of attack attempts to victimize a system or service by sending a large number of similar requests, exceeding the capacity of the system or service, such as *Flooding* and *DoS*⁹³.
- *Others*: These attacks aim to infect the target system directly by exploiting system vulnerabilities or weaknesses.

Furthermore, network attacks may be characterized as passive or active [84]. Attackers use passive assaults to watch and capture crucial information shared within network traffic to conduct an active attack later. Examples include traffic monitoring and analysis, as well as packet sniffing. Active assaults, on the other hand, have a more severe impact on the network since they target system destruction. They are classified into four categories: i) *DoS*; ii) *U2R*⁹⁴; iii) *R2L*⁹⁵; iv) *Probing Attack*. These are actual the attacks used to create the *KDD Cup99* dataset (more detail in Section 2.4.5).

DoS is defined in this context as attacks that aim to obstruct access to system resources by consuming them exhaustively, to the point that the service can no longer be delivered. This is a hybrid of the previous *Exploding* and *Concurrency* categories. *Buffer overflow*, *TCP 3-Way Handshake*, and *PoD*⁹⁶ are common variations of *DoS* attacks. *U2R* refers to attacks that attempt to achieve superuser privileges on a system by exploiting flaws in the system. Typically, these assaults are carried out after the intruder has gained access to the system via a normal user account and has stolen the password. *R2L* attacks, on the other hand, include an intruder attempting to obtain unauthorized access to remote devices or hosts across a network by sniffing passwords. Finally, a *Probing Attack* involves scanning the network for information such as *IP* addresses, services supplied, operating systems utilized, and so on. This information can be used later to explore potential system vulnerabilities and launch a wider-scale attack against the system.

⁹³*Denial of Service*

⁹⁴*User to Root*

⁹⁵*Remote to Local*

⁹⁶*Ping of Death*

Oriented to CPPS scenarios using OPC UA, Polge, Robert & Le Traon [85] and Cavalieri *et al.* [86] highlighted the major OPC UA risks and their effect on OPC UA applications. As previously stated, this standard contains a security specification, so it implements adequate security features. However, the authors argue that it cannot compensate for legacy technologies' lack of 'security-by-design'. In addition, Neu, Schiering & Zorzo [87] offered a thorough investigation of DoS attacks that untrusted clients may conduct against OPC UA servers. The following are the key recognized threats detected in the OPC UA protocol and, more broadly, in the CPPS:

- *Eavesdropping* - *Eavesdropping* or MITM⁹⁷ attacks target system privacy by monitoring and acquiring information via remote access technologies. An attacker can either passively monitor the information by listening to the transmission medium and comprehending unprotected communications, or actively analyze the traffic by initiating certain processes by providing control data as queries to locate sensitive information.
- *Denial of Service* - The primary purpose of DoS attacks is to disable the device's ability to execute its intended function. On the one hand, if it is implemented at the physical level, DoS occurs as a result of jamming, manipulation, and collision in packet transmission. On the other hand, if deployed at the network layer, it is accomplished by interfering with the network protocol, preventing measured data from reaching its target, or causing battery fatigue due to packet re-transmissions. DoS attacks are characterized by link-layer collisions and fatigue, network layer black holes and misdirection, malicious floods, and time synchronization disruption at the transport layer.
- *Server Profiling* - *Server Profiling* is a sort of *Node Replication* attack in which hostile nodes imitate legal nodes and are propagated in a network. Because hostile nodes inject bogus packets to be routed, network performance would be degraded.
- *Message Spoofing* - A *Message Spoofing* attack involves impersonating a network node. Unlike the *Node Replication* attack, in which a malicious node clones and takes the identity of a genuine node, the *Message Spoofing* attack does not involve an actual node, but rather a larger and more powerful machine used as a server or gateway. The attacker may steal the received information and send new messages or change existing ones to transmit information that differs from what was received in the first place.
- *Message Alteration* - *Message Alteration* (or *Corruption*) attacks aim to compromise the integrity of a packet by remotely altering its contents.
- *Message Flooding* - *Message Flooding* is the practice of sending a large number of connection setup requests to the server or host that cannot be completed. This attack fills the host's memory buffer. Once this buffer is filled, no further connections can be established,

⁹⁷Man-in-the-Middle

resulting in an a DoS attack. The *HELLO Flood* attack is similar. It is used to drain energy from Edge devices by sending a *HELLO* message that is comparable to the messages already broadcasted within the system.

- *Message Replay* - In systems with real-time or near real-time needs, purposely delaying packets within a forwarding component might decrease the quality of service by delaying the delivery of important events. In addition, the attacker may cache previously sent data and resend it later to deceive the receiving node.

2.4.5 Datasets

Various research groups created datasets to aid in the performance evaluation of different A-NIDSs when trying to detect network threats. These datasets were created by selecting network nodes, gathering and pre-processing network traffic data, conducting various forms of network attacks, and monitoring traffic trends. It is also critical to recognize that some of these datasets are public and others are private. Furthermore, network simulating datasets are built by simulating normal and attack traffic rather than monitoring a network from a real scenario or testbed. We only consider public datasets created using a real scenario or testbed, since private ones are inaccessible and network simulation does not represent critical network dynamics for intrusion detection. In this case, the most important public datasets for network intrusion detection are the *KDD Cup99*⁹⁸, the *NSL-KDD*⁹⁹ and *UNSW-NB15*¹⁰⁰.

2.4.5.1 KDD Cup99

The *KDD Cup99* [88, 89] dataset is the most known and used benchmark for network intrusion detection. It was created in 1998 by the *MIT Lincoln Labs* as part of the *DARPA*¹⁰¹ Intrusion Detection Evaluation Program. For nine weeks, a network architecture was built to capture raw TCP dump data for a standard US Air Force LAN¹⁰². They ran the LAN as if it were a genuine Air Force network, but they introduced various attacks. The basic training dataset was around four terabytes of compressed binary TCP dump data obtained from seven weeks of network activity. This yielded almost five million network records. This dataset was then filtered for use in the International Knowledge Discovery and Data Mining Tools Competition [89], yielding what is currently known as the *KDD Cup99* dataset.

Each record represents a connection between two network hosts and is distinguished by 41 features, 38 of which are numerical continuous and discrete and three of which are categorical [88]. The dataset characteristics may be classified into three types: i) Basic features: attributes taken from a TCP/IP connection; ii) Traffic features: features computed concerning a time window interval; and iii) Content features: features including suspicious behavior in the data component of

⁹⁸<https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

⁹⁹<https://www.unb.ca/cic/datasets/nsf.html>

¹⁰⁰<https://research.unsw.edu.au/projects/unsw-nb15-dataset>

¹⁰¹Defense Advanced Research Projects Agency

¹⁰²Local-Area Network

packets. Each record is classified as normal or an abnormality, which can be one of four categories, including [DoS](#), [U2R](#), [R2L](#) and *Probing Attack*. The dataset is widely used and is regarded as a gold standard for intrusion detection. The dataset can also be accessed in two other versions: *Corrected KDD* and the *10 percent KDD* datasets. The *Corrected KDD* is the *KDD Cup99* dataset used for testing, i.e., that includes the corrected attack labels. It contains 311 029 records. On the other hand, the *10 percent KDD* is a 10% subset, with 494 021 records.

2.4.5.2 NSL-KDD

The *NSL-KDD* [90] is an attempt to correct and overcome the *KDD Cup99*'s limitations. The enhancements are linked to: i) Size and redundancy: *NSL-KDD* has fewer data points than *KDD Cup99*, because only one copy of duplicated records is kept; ii) *NSL-KDD* presents a reduced set of 20 features in comparison to *KDD Cup99*; iii) Skewedness: *NSL-KDD* contains an undersampling of some classes; iv) Non-Stationarity: Improved data distribution because the target spread of the train and test sets are significantly more alike. The *NSL-KDD* is made up of two datasets: the *KDDTrain⁺*, which has 125 973 records, and the *KDDTest⁺*, which has 22 544 records. Also, a 20% subset of the *KDDTrain⁺* and a 20% subset of the *KDDTest⁺* are available.

2.4.5.3 UNSW-NB15

The most recent dataset is *UNSW-NB15* [91], which may be viewed as a contemporary *KDD Cup99* alternative. It uses the same generating process and features as *KDD Cup99*. Still, it addresses many difficulties that make *KDD Cup99* less adequate for usage in contemporary [A-NIDS](#) methods.

The *UNSW-NB15* dataset's network packets were generated in the Cyber Range Lab of [UNSW](#) Canberra to provide a hybrid of genuine modern normal activities and synthetic contemporary assault behaviors. The collection contains nine types of attacks: *Fuzzers*, *Analysis*, *Backdoors*, *DoS*, *Exploits*, *Generic*, *Reconnaissance*, *Shellcode* and *Worms*. There are a total of 257 673 records. There are both training and testing sets, with the training set including 175 341 records and the testing set being 82 332 records from the different types, attack and normal.

2.5 Conclusion

[SMS](#)s can be seen as intelligent environments, where different kinds of [CPS](#)s are continuously working to enable one or more functionalities. A [CPS](#) exhibits properties of ubiquitous computing and artificial intelligence, which can operate autonomously and is usually connected to other [CPS](#)s. Moreover, [CPS](#)s are very important since they enable the application of [IoT](#) principles in a given context. If one considers the industrial context, [CPPS](#)s and [IIoT](#) play a significant role in the emergence of some prominent application areas, such as:

- Improve reliability and efficiency of processes

- Real-time parametrization of industrial equipment based on the product quality
- Tracking of materials in a warehouse
- Management of the equipment life cycle
- Reconfiguration of industrial processes according to the demand

However, several research challenges can be seen as essential requirements for technology acceptance in the industry, such as reliability, security & privacy, recovery, interoperability, and reconfigurability. This chapter addressed the industrial cyber-security topic as background, supporting the reliability, security & privacy, and recovery properties. On the other hand, the chapter also discussed the [MBE](#) and the [IEC 61499](#) standard topics regarding the interoperability and reconfigurability requirements.

This chapter's main contribution is the presentation of multiple technologies and topics related to [SMSs](#). It was addressed in detail [CPPSs](#), [MBE](#), [IEC 61499](#), the [DINASORE](#) framework, anomaly network detection, main network attacks deployed in the [OPC UA](#) protocol, and the most important datasets used to validate [NIDSs](#).

Chapter 3

Autonomic Computing

3.1 Introduction

In this chapter, the reader will learn about the theoretical concepts and technologies related to the *Autonomic Computing* paradigm. For this, in Section 3.2, we introduce the paradigm by detailing the Self-x properties, i.e., properties that characterize the maintenance, improvement, and restoration of the system functionality without external actions on the system. The primary objective with Self-x properties is to enable the system's Self-managing characteristics by adapting to unpredictable changes and overcoming the rapidly growing complexity of computing systems. Thus, based on the Self-x properties, we present a taxonomy of manufacturing systems autonomy, i.e., [LOA](#)¹, defining a six-level model of automated decision-making based on industrial processes.

Then, the reader's attention is redirected to the concept of *Biologic Inspired Computation* in Section 3.3. This field of study seeks to solve computer science problems using biology models, such as social and emergent behavior. Thus, we use the [BIS](#)² as inspiration and describe the main immune-inspired techniques comprehensively, in terms of both biological process inspiration and engineering immune algorithms, also known as [AISs](#)³. Within the [AIS](#) research field, in Section 3.4, the reader will learn in detail the [DCA](#)⁴, which is currently one of the most important immune models and the one used in the research work described in later chapters. Finally, Section 3.5 concludes the chapter, stating final remarks about the work presented.

Partial results of the presented work in this chapter have been submitted at:

- Pinto, C., Pinto, R., & Gonçalves, G. (2021). Towards Bio-inspired Anomaly Detection using the Cursory Dendritic Cell Algorithm. In Algorithms. MDPI. (Published) [3].

¹Level of Autonomy

²Biological Immune System

³Artificial Immune Systems

⁴Dendritic Cell Algorithm

3.2 System Autonomy

Computer systems have grown to the point where human interaction for operation and control is becoming increasingly challenging. The idea of ubiquitous and pervasive computing will become a reality in the [IoT](#)⁵ paradigm, which will surround us with embedded technology and convert every physical thing into a [CPS](#)⁶. These [CPS](#)s are linked to sensors and actuators to maintain the connection between the real and virtual worlds, and they are linked to other [CPS](#)s via communication networks such as the Internet. Thus, how can such a complicated system be controlled, given all of the active interacting diverse devices and uncertainty? The [IBM](#)⁷ proposed the idea of *Autonomic Computing* in 2001, alluding to the need for new techniques to manage and regulate the behavior of complex systems. These systems should be able to adapt themselves, removing the need for human intervention [92].

According to [IBM](#) [92], a system is autonomic if it: 1) understands what resources it has access to, what its capabilities and constraints are, and knows how and why it is connected to other systems; 2) can autonomously configure and reconfigure itself in response to changing computing environments; 3) can optimize its performance to ensure the most efficient computing process; 4) can work around difficulties by healing itself or redirecting functions away from the damaged region; 5) can detect, identify and protect itself against various types of attacks to maintain overall system security and integrity; 6) can adapt to its changing environment by interacting with neighboring systems and establishing communication protocols; 7) relies on open standards and cannot exist in a proprietary environment; and 8) anticipates demand for its resources while remaining transparent to users.

The *Autonomic Computing* concept, which was later extended to *Organic Computing* [93], was inspired by the human autonomic nervous system [94]. This system removes from consciousness the tasks of coordinating all of our bodily functions (for regular maintenance and optimization) by taking care of the majority of them. According to Kephart and Chess [13], system administrators are unable to predict, develop, and maintain the complexity of interconnections between devices and computer systems, particularly in [CPS](#). As a result, computer systems should continually adapt to changing environmental conditions, using a logic loop that monitors the execution context. Such dynamics may be implemented using a [MAPE](#)⁸ loop or a [MAPEK](#)⁹ loop architecture [92, 13], as shown in Figure 3.1.

According to Ganek and Corbi [95], the essence of *Autonomic Computing* is system Self-management, which includes the following Self-x, Self-* (Self-star), or Self-CHOP properties: Self-configuration, Self-healing, Self-optimization, and Self-protecting. Later, the autonomic paradigm was expanded to include properties such as Self-organizing, Self-stabilization, Self-awareness, Self-scaling, Self-containment, Self-adaptive, Self-immunity, among others.

⁵Internet of Things

⁶Cyber-Physical System

⁷International Business Machines Corporation

⁸Monitor-Analyze-Plan-Execute

⁹Monitor-Analyze-Plan-Execute-Knowledge

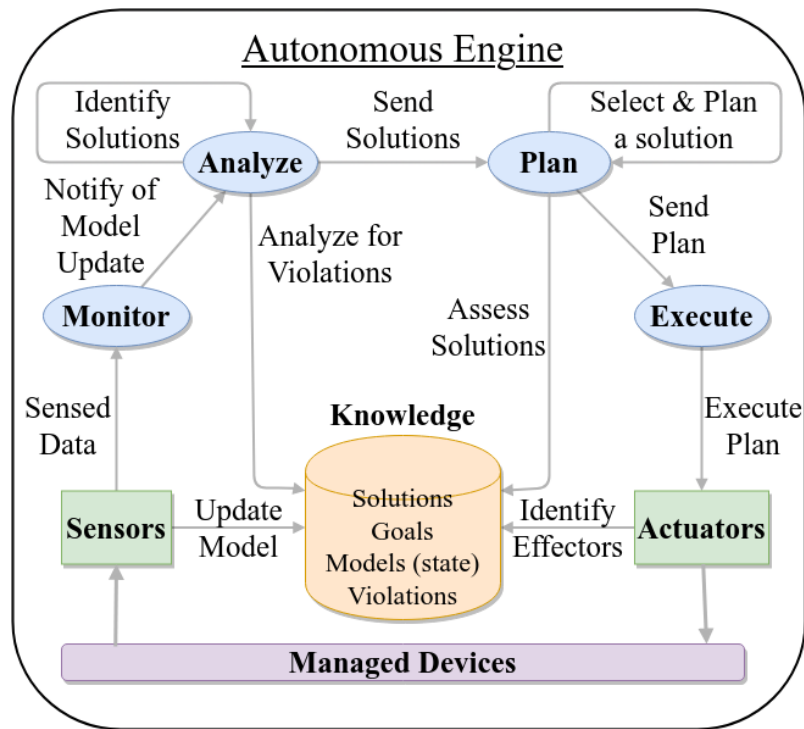


Figure 3.1: *Monitor-Analyze-Plan-Execute-Knowledge* model for autonomic systems [13].

3.2.1 Self-x Properties

According to Berns and Ghosh [96], Self-managing systems are those that maintain, improve, and restore their functioning or attributes without the need for external intervention. The general quality that incorporates the many Self-x qualities is Self-management, which is based on Zadeh's idea of adaptivity [97]. However, there is ambiguity concerning the meaning of these Self-x qualities because a clear difference between concepts is a matter of perspective. Many Self-x terms are synonyms for previously established concepts, such as Self-adjusting, which is a synonym for Self-management, and Self-repairing, which is a synonym for Self-healing. Also, specific Self-x terms, such as Self-awareness and Self-monitoring, do not fit into the Self-management category and do not describe the system's behavior. These are features that the system may have and employ but do not indicate anything about the system's behavior. The following section discusses important Self-x features and a possible related definition.

- Self-management [13] or Self-adjusting [98]: Systems that maintain, improve, or restore their functioning or attributes without requiring external intervention.
- Self-stabilization [99, 100]: Self-stabilization refers to a system's capacity to recover spontaneously from defects induced by external events that do not impact the source code of the various processes.
- Self-adaptive [97, 101]: A Self-adaptive system modifies its safety features in response to changes in the environment.

- Self-healing [102, 103, 104] or Self-repairing [105]: Self-healing is concerned with preserving or recovering the state of the system in a finite length of time after it has been influenced by a subset of external activities, resulting in a deteriorated level of performance. This is in contrast to Self-stabilizing, which ensures recovery from any actions that disrupt the system's global state.
- Self-optimization [106]: The goal of self-optimization is to maximize (or minimize) the value of an objective function as a system attribute. The objective is to recognize sub-optimal behaviors and adjust itself to maintain or increase system performance. The objective function represents the system's overall state.
- Self-configuration [107, 108]: Given that configuration is defined as a collection of links between system components, a Self-configuration system can modify its configurations in response to a set of external events to restore or enhance its global state. As long as the reaction affects the configuration, Self-configuration can be a reaction to any collection of external activities.
- Self-organization [109, 110, 111, 112]: When the system's safety qualities are restored, Self-organizing systems exhibit the same characteristics as Self-healing systems, except for changing the system topology by adding or removing nodes. Self-organization focuses on efforts that address component join and departure. Every Self-organizing system is Self-configuring, but not the other way around.
- Self-scaling [113, 114]: Self-scaling is concerned with changes in the system scale. It refers to the system's capacity to work well over a wide range of system sizes. Self-scaling caters to the functional features of the system rather than the topological properties attended to by Self-organization. In contrast, Self-scaling systems may need to adapt topology to accommodate a change in size.
- Self-awareness [115, 116] or Self-monitoring [117]: The system must be Self-aware of its state and surrounding environmental circumstances to modify and adapt to changes and external activities. Thus, it can achieve the Self-management objectives. A system component must be Self-aware of the passage of time while carrying out its corresponding operation, the progress of its execution, and the resources it consumes.
- Self-immunity [118, 119, 120]: A system is Self-immune when it recovers safety properties in the face of an external action (malicious or not). Ultimately, safety properties are no longer compromised when the same action occurs, acquiring tolerance over time. Every Self-immune system is also a Self-healing system, but not the other way around. This is due to the learning component of a Self-immune system, which allows a tolerance mechanism to be activated at run-time just when needed, rather than statically overloading the system regardless of whether the errors occur or not.
- Self-containment [121]: If external hostile acts compromise part of the system, and a regular operation is finally restored with the undisturbed portion of the system, then the system is

Self-contained. The goal is to reduce the impact of the harmful conduct to control the damage and eventually recover.

- Self-protection [122, 123]: A system is Self-protecting if it continually preserves safety attributes such as privacy and integrity in the face of a subset of hostile external. It acts by employing trust mechanisms or privacy policies to safeguard the system and its data. Self-protection implies Self-containment. Furthermore, every Self-protecting mechanism is Self-immune, but the opposite is not valid.

3.2.2 Levels of Autonomy

One of the critical aims of SMSs¹⁰, as previously stated, is to target the adoption of autonomous and collaborative manufacturing assets with advanced self-capabilities [124]. With this in mind, Peres *et al.* [125] provide a taxonomy of manufacturing system autonomy, outlining a six-level model of automated decision-making based on industrial processes. For each LOA, the model was contextualized with industrial scenarios:

- Level 0: There is no autonomy. Without the support of an AI¹¹ system, human operators have complete control.
- Level 1: Assistance with certain functions. Human operators are fully responsible for all choices.
- Level 2: Partially autonomous. Human operators have complete authority and specify (some) goals in clearly defined areas.
- Level 3: Autonomy is delimited. In bigger sub-areas, the AI system alerts if problems arise, and humans confirm the system's proposed remedies.
- Level 4: Adaptability and self-sufficiency. Human operators can oversee or act in emergency scenarios within a set of system parameters.
- Level 5: Complete independence. The AI system acts autonomously in all domains, including collaboration and shifting system boundaries. There is no requirement for human operators to be present.

3.3 Biologic Inspired Computation

In this Thesis, we seek to study and enable Self-immunity properties in CPPSs¹². This is possible due to the development of an A-IDS¹³. Also, since Self-immunity properties belong to the autonomic paradigm, the goal is to implement techniques that follow *Autonomic Computing* principles, such as AISs. AISs belong to the broader paradigm of *Biologic Inspired Computation* [126].

¹⁰Smart Manufacturing Systems

¹¹Artificial Intelligence

¹²Cyber-Physical Production Systems

¹³Anomaly Intrusion Detection System

Biologic Inspired Computation refers to the study of biologically motivated computation, which belongs to the field of *Natural Computing* and a sub-field of *Computational Intelligence*. *Biologic Inspired Computation* main goal is to solve engineering and computation problems by generating solutions inspired by different biological processes from the natural world. There are several sub-disciplines that focus on adaptive and intelligence systems, such as *Fuzzy Systems*, [ANNs](#)¹⁴, and *Evolutionary Computation*, e.g., *Swarm Intelligence* and [AISs](#).

3.3.1 Biologic Immune System

The [BIS](#) is a Self-organizing and robust system with extraordinary skills such as recognition/discrimination, maintenance, inference from danger/context, and memory. To maintain a healthy state, the [BIS](#) processes the status of the body and takes action appropriately through healing processes [127, 128]. The primary role of the [BIS](#) is to defend an organism from illness by recognizing and differentiating a wide range of external intruders from the organism's healthy tissue. *Pathogens*, in the broadest sense, are any infectious agent capable of causing disease, including viruses, parasitic worms, and cancer cells. *Antigens* are chemicals found on the surface of *Pathogens* that are capable of triggering an immunological response in the host organism. In contrast, *Antigens* might occasionally be part of the host itself.

The [BIS](#) is a layered system with multiple sorts of protection mechanisms active in each layer. The three primary protection lines are the anatomic barrier, innate immunity, and adaptive immunity. The anatomic barrier consists of physical obstacles such as skin, mucous membranes, and skin secretions that prevent *Pathogens*, such as bacteria and viruses, from entering the body. Suppose a *Pathogen* gets past the initial line of defense. In that case, innate immunity kicks in with a quick yet non-specific reaction, such as an inflammatory response and antimicrobial proteins. If *Pathogens* successfully dodge the innate response, adaptive immunity is activated, which adapts its response throughout an infection to increase its detection abilities and destroy those specific *Pathogens*.

To provide immunity to the organism, the [BIS](#) consists of a set of organs, as represented in Figure 3.2, cells and molecules [129, 130]. The central and peripheral lymphoid organs are the primary [BIS](#) organs. The primary lymphoid organs, such as bone marrow and the thymus, are in charge of producing and assisting white blood cells, such as *Lymphocytes* and *Granulocytes*. The bone marrow produces all of our immune cells, whereas the thymus trains *T-cells* (a kind of *Lymphocyte* cell) to identify foreign *Antigens* instead of proteins created by our own body. The function of peripheral lymphoid organs such as lymph nodes, the spleen, and mucosal tissues is to enable the interaction between *Lymphocytes* with *Antigens*. The spleen is a crucial organ for presenting *Antigens* to the immune system because it accumulates and recycles immune cells. On the other hand, the lymphatic system, which consists of lymph nodes connected throughout the body by lymphatic vessels, is a primary location for *Lymphocytes* and serves as a filter of *Antigens*.

¹⁴Artificial Neural Networks

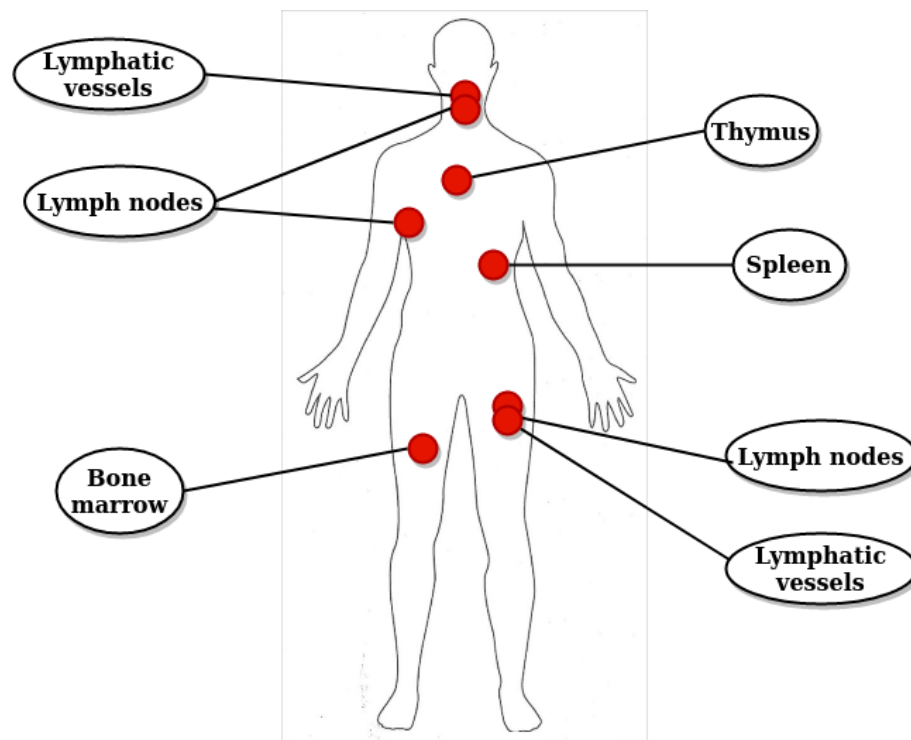


Figure 3.2: *Biological Immune System* main organs.

There are *Lymphocytes* and *Granulocytes* among the **BIS** cells. *Granulocytes* are cells in the innate immune system that include *Eosinophils*, *Basophils*, *Neutrophils*, *Mast Cells* and *Monocytes*. *Monocytes* are also known as *Macrophages* or *Dendritic Cells*. The majority of these cells are *Phagocytes*, or body patrolling cells that look for *Pathogens*. When *Pathogens* are discovered, they are destroyed by *Phagocytes*, either by direct contact or by engulfing and then destroying microbes. *Dendritic Cells* are *Phagocytes* in tissues that come into contact with the outside world. They act as a bridge between the body's tissues and the innate and adaptive immune systems by delivering *Antigens* to *T-cells*, one of the adaptive immune system's essential cell types.

Lymphocytes, on the other hand, are the most common kind of cell identified in lymph nodes. *Natural Killer Cells*, *T-cells*, and *B-cells* are among them. *Natural Killer Cells* are part of the innate immune system, whereas *T-cells*, and *B-cells* are part of the adaptive immune system. *T-cells* are cell-mediated, which means they destroy malignant and virus-infected cells while also directing other immune cells. *B-cells* produce *Antibodies*, also known as *Immunoglobulins*, which are specialized molecules that recognize *Antigens* and neutralize the corresponding *Pathogens*, designating them for destruction. Figure 3.3 depicts a simplified taxonomy of the main **BIS** cells.

The most essential cells in adaptive immune systems are *T-cells* and *B-cells*. *B-cells* are in charge of recognizing certain *Antigens* and creating *Antibodies* that bind to *Antigens*. *B-cells* are triggered and the maturation process evolves in the presence of *Antigens*. *B-cells* maturation consists on becoming *Plasma Cells* or *Memory Cells*. *Plasma Cells* actively secrete *Antibodies*, whereas *Memory Cells* are *Plasma Cells* that can produce *Antibodies* for an extended period of

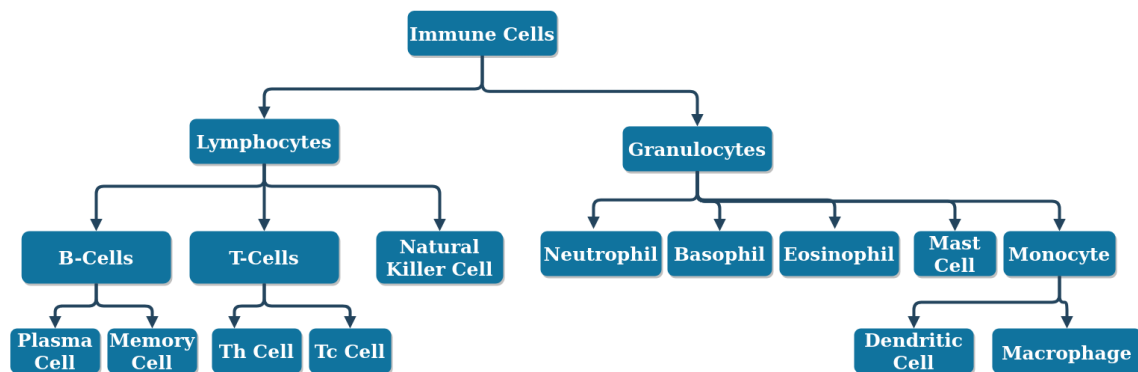


Figure 3.3: Simplified taxonomy of cells in the *Biological Immune System*.

time, i.e., they remain on the organism in order to quickly and specifically recognize an antigen that the body has previously encountered and initiate a faster corresponding immune response. *T-cell* maturation in the thymus can result in *T-cell* sub-populations. *Helper T-cells* (Th Cells), which assist *B-cells* in *Antigen* identification, and *Cytotoxic T-cells* (Tc Cells), which destroy infected self-cells, tumor cells, and foreign cells, are the most significant.

Although layered, innate and adaptive immunities are inextricably linked since numerous components in innate and adaptive immunity interact to produce an immune response in the face of an intruder. Several steps can be used to define a simplified primary immune response:

1. *Pathogens* invade the body by successfully breaching the first line of defense and attack body cells;
2. *Macrophages* encounter *Pathogens* (and also infected cells) and begin the *Phagocytosis* process;
3. During *Phagocytosis*, *Pathogens* are broken down into components, which are then presented to *T-cells*;
4. Facing the presented *Pathogens* components (such as *Antigens*), *T-cells* mature to sub-populations such as *Cytotoxic T-cells*, which will destroy infected body cells. *Macrophages* will later phagocytize and digest those dying cells;
5. On the other hand, *T-cells* activate specific *B-cells* (according to the *Antigens* presented), which will proliferate as *Plasma Cells* while creating the first specific *Antibodies* against the *Antigens*;
6. *Antibodies* bind with *Antigens* and neutralize remaining *Pathogens*, which will later be engulfed and digested by *Macrophages*;
7. Some *Plasma Cells* mature as *Memory Cells*, which can later induce a secondary immune response upon renewed contact with the same *Pathogens*.

3.3.2 Artificial Immune Systems

AISs are a class of biology-inspired computational approaches that arose in the 1990s, connecting several fields, including immunology, computer science, and engineering. **AISs** were created to computationally simulate immunologic processes in biology, abstracting the concepts into algorithms and implementing them in engineering settings. As previously said, the **BIS** is a very complex system, and many immunological mechanisms are still poorly understood, resulting in debate and little consensus among immunologists. This results in a lack of understanding of how immunological mechanisms work and, for **AIS** practitioners, a problem in accurately modeling a computer system. The great majority of **AISs** have been influenced by four critical immunological pathways, according to Dasgupta *et al.* [131]:

1. *Negative Selection Algorithm* [132]
2. *Clonal Selection Algorithm* [133]
3. *Artificial Immune Network* [134, 135]
4. *Danger Theory* [136, 137]

Given the nature of **AIS** approaches, they have already been shown to be useful for solving real-world issues in computer science and engineering. They are used in a variety of applications, including **ML**¹⁵-related, pattern recognition and classification, computer virus detection, anomaly detection, optimization, robotics, and so on. The reader can refer to Chapter 4 for more information about the application of **AISs** in **SMSs**. Aldhaheri *et al.* [138] suggests six distinct characteristics of the **AIS** approaches:

1. **Adaptive**: can learn over time and develop adaptive behaviors.
2. **Robust**: can process under imprecise and uncertain data circumstances.
3. **Resiliency**: the immune system is created with a bottom-up approach, following an Agent-based paradigm to realize dynamic, heterogeneous, and distributed environments.
4. **Self-tolerance**: can prevent an immune response against a particular self *Antigen*, i.e., representing benign *Pathogens*.
5. **Lightweight**: can process with low computational complexity.
6. **Distributed**: can process in a distributed manner without any centralized control or management.

There are various world-wide **AISs** applications. Most of them are found in computer security, namely virus detection, network security, and **A-IDSs**, because identifying anomalies in

¹⁵*Machine Learning*

computer systems closely mirrors the functioning of the BIS. Greensmith *et al.* [139] developed an IDS¹⁶ using the DT¹⁷ concept, namely the DCA. In the port scan detection problem, the authors evaluated a DCA-based IDS. Later, the same DCA approach was used to solve the bot detection problem [140], online break-in fraud for an Online Video on Demand System [141], and the DoS¹⁸ Detection Problem in IEEE 802.11 Networks [142]. More recently, Alsulami and Zein-Sabatto [143, 144] applied AISs methods, specifically the NSA¹⁹, to CPSs in the Aviation industry to identify false data injection and sensor spoofing threats.

The second major area of AISs applicability is in optimization problems. Cayzer and Aickelin [145] provide an AIS method to the task of film recommendation. Zand *et al.* [146] shows how the CSA²⁰ may be used for optimization in the File Transfer Scheduling problem. Based on AIS, a traffic signal control system was created to oversee an isolated junction and construct a regulation plan if a disruption (such as congestion or an accident) is detected [147]. In addition, Chen and Zhang [148] developed an AIS strategy for wireless access network scheduling in a 5G messaging delivery system.

Other suggestions are more focused on ML-related problems, such as pattern recognition and classification problems. Takeda *et al.* [149] proposed a biometric personal authentication technique that uses the CSA as a classifier training method. Xu *et al.* [150] utilized AIS for GPS²¹ data processing to solve GPS nonlinear models and prevent ill-conditions (single-frequency precise point positioning). Finally, AIS was applied in wireless multimedia sensor networks for energy-efficient, distributed, and collaborative picture pattern recognition on camera sensors [151].

3.3.2.1 Negative Selection Algorithm

Inspiration The NSA is based on the *Self/Nonself theory*, which was initially suggested by Forrest *et al.* [132]. The capacity of the BIS to discriminate between self and non-self cells constitutes the self-nonself discriminating behavior. According to the adaptive immunity functions, non-self cells are further classified to elicit a suitable response while giving self cells tolerance. This is done via *T-cell* maturation in the thymus, which is based on a negative selection process. The negative selection process consists in the development of a repertoire of *T-cells* that are not auto-immune, i.e., they will not react to self-cells. First, *T-cells* are created using a pseudo-random genomic rearrangement method. Then, created *T-cells* are exposed to self-proteins to select only well-formed *T-cells*. A well-formed *T-cell* does not attach to self-proteins and can thus be discharged from the thymus into the body to conduct immunological activities. *T-cells* that respond against self-proteins, on the other hand, are destroyed.

The NSA was one of the first and most well-accepted immunity theories since it explained the BIS functioning of recognizing foreign and possibly hazardous *Pathogens* and responding

¹⁶*Intrusion Detection System*

¹⁷*Danger Theory*

¹⁸*Denial of Service*

¹⁹*Negative Selection Algorithm*

²⁰*Clonal Selection Algorithm*

²¹*Global Positioning System*

appropriately while ignoring harmless substances and own bodily cells. A simplified form of the negative selection process is depicted in Figure 3.4.

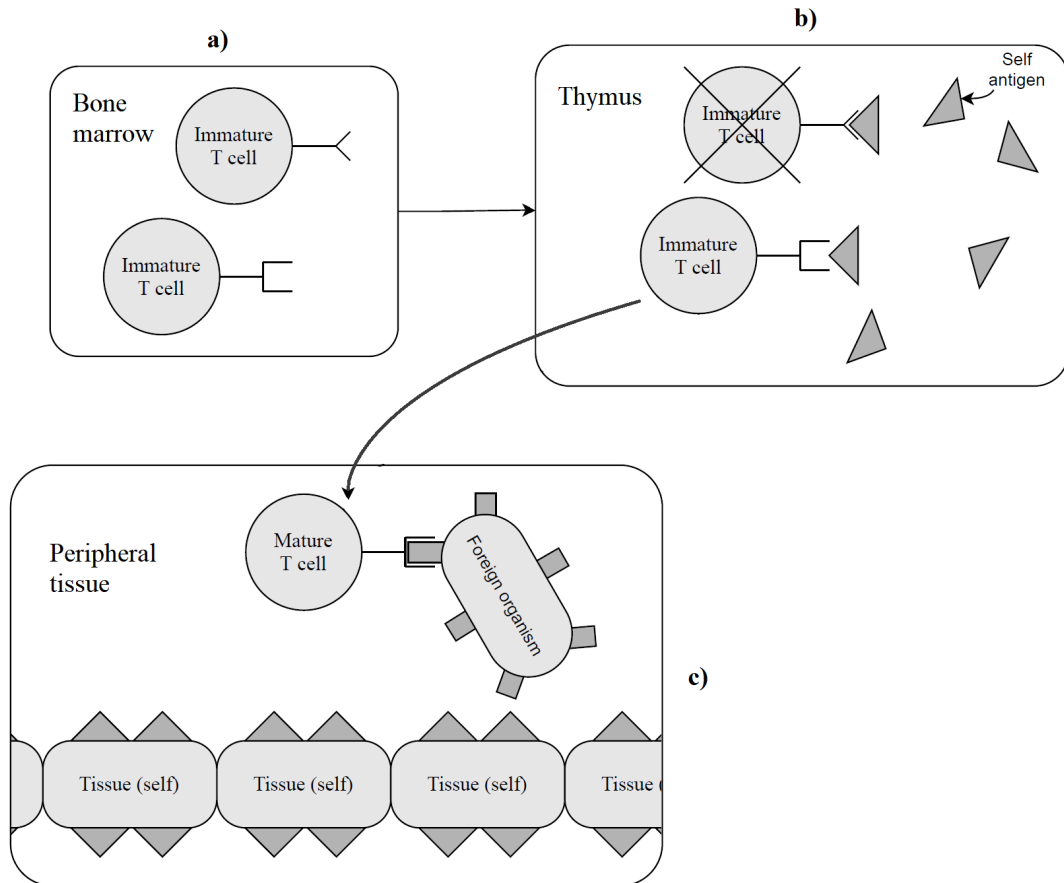


Figure 3.4: Simplified diagram of *Self/Nonself theory* [3]. It consists of 3 main steps: a) Immature *T-cells* with various *Antigen* receptors are produced in the bone marrow and migrate to the thymus; b) In the thymus, *T-cells* that recognize self *Antigens* undergo *Apoptosis* (programmed cell death). On the other hand, *T-cells* that don't recognize any self *Antigen* are allowed to live; c) Mature *T-cells* that don't recognize self *Antigens* are now used to recognize non self *Antigen* invading the body, such as virus and bacteria.

Algorithm Overview Despite the advancement of this approach, the basic characteristics of the *NSA* defined by Forrest *et al.* [132] in 1994 remain, including negative information representation, distributed creation of the detected set, and one-class classification [152]. Alternative variants of the *NSA* have been presented more recently. They rely heavily on different methods for data encoding, matching criteria, and detector generation mechanism [131]. Data and *T-cell* representation, generation and removal process, and matching rule are all important features of *NSA* [153]. A well-known *NSA*-based approach for network intrusion detection is *ARTIS*²² [118, 154]. Authors

²²*ARTificial Immune System*

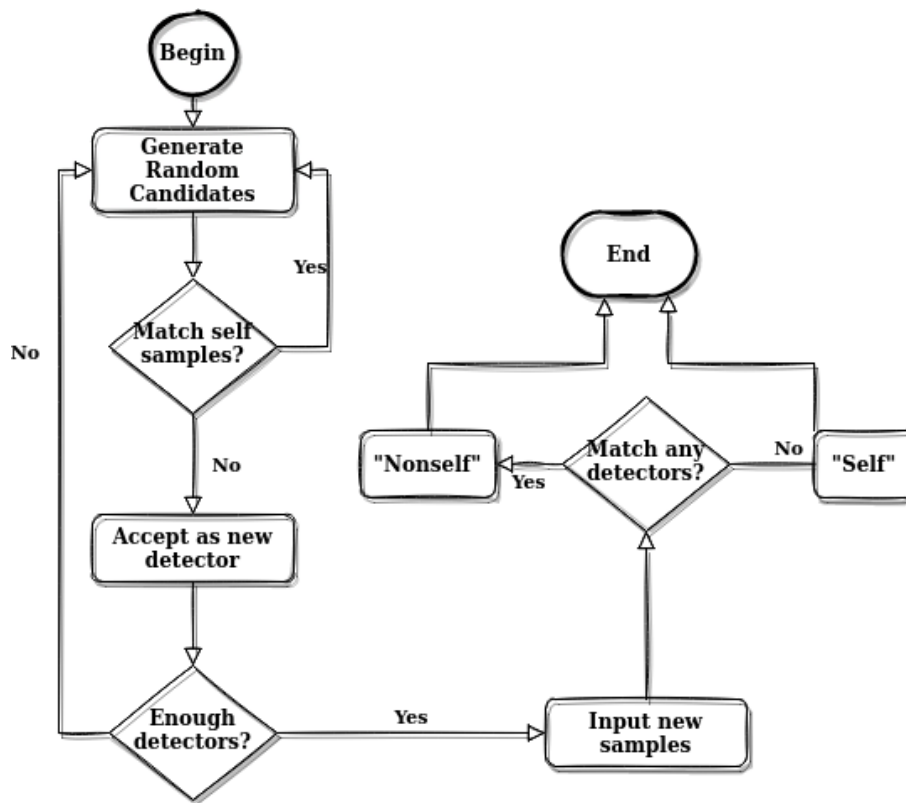


Figure 3.5: Flowchart of the *Negative Selection Algorithm*.

designate this solution as [LISYS](#)²³. Figure 3.5 depicts a flow chart of the [NSA](#), demonstrating the processes of detector set formation and detection of new occurrences.

The method generates a population of 'detectors,' which are similar to *T-cells*. Detectors can be created and expressed in many ways depending on the type of data represented in the problem domain (binary, real-valued, strings, and others.). Then, a metric (most likely distance-wise) is required to compare detectors to regular behavior data points. These regular behavior data points might be compared to self-cells. This allows for implementing the negative selection process, in which detectors that react to self-cells are destroyed immediately. In contrast, non-reactive detectors are kept for later (*T-cell* maturation). In practice, this reaction is caused by a matching rule utilized to compare detectors and self-cells. The remaining detectors are then used to identify and categorize non-normal data. If the detectors did not react to normal behavior, then when they do react, this will be termed abnormal behavior.

3.3.2.2 Clonal Selection Algorithm

Inspiration The [CSA](#) [155] is based on clonal selection theory, which seeks to explain the adaptive immune system's fundamental response to an *Antigen* stimulation, i.e., how *T-cells* and *B-cells* increase their response in the presence of *Antigens*. On the one hand, the presence of an *Antigen*

²³Lightweight Intrusion detection SYStem

activates *B-cells*, which mature later into short-lived *Plasma Cells* that secrete *Antibodies* and long-lived *Memory Cells*. *Antibodies* with greater affinities are cloned, and the clones are distinguished by a mutation process. *T-cells*, on the other hand, can control the previously described *B-cell* activity in response to the *Antigen* stimulation.

In these cases, following numerous *Antigens* exposures, the concentration of *B-cells* with increased affinity for those *Antigens* increases. *Memory Cells* will allow for rapid immune responses to future instances of *Antigens* that have previously been encountered. It is also crucial to note that when a *B-cell* is picked and proliferates, it is subjected to tiny hypermutations, which shape the recognition capacities of both receptors linked to the surface of *Lymphocytes* and *Antibodies* produced by *Plasma Cells*. A simplified version of the clonal selection theory is seen in Figure 3.6.

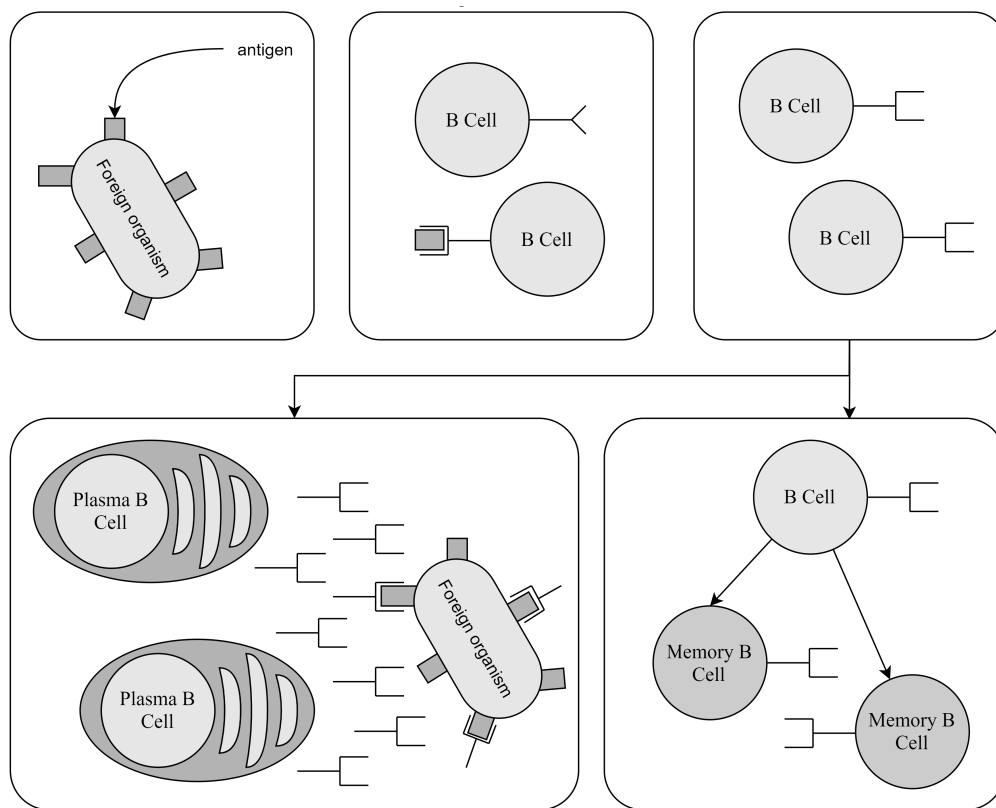


Figure 3.6: Simplified diagram of the clonal selection theory [3]. A foreign organism contains specific *Antigen* that can be recognized by *B-cells'* receptors with high affinity. *B-cells* that recognize those *Antigen* are cloned, resulting in *Plasma B-cells* and *Memory Cells*. Short-lived *Plasma B-cells* release *Antibodies* that bond to the considered *Antigen* and inhibit the foreign organism. Long-lived *Memory Cells* will react quickly in future occurrences of the same *Antigens*.

Algorithm Overview The *CSA* is based on the notion that only cells capable of detecting *Antigens* would replicate. The *CLONALG*²⁴ [133, 156] was the first algorithm presented to handle pattern recognition issues by taking clonal selection and affinity maturation processes into account. It

²⁴The *CLONal selection ALGORITHM*

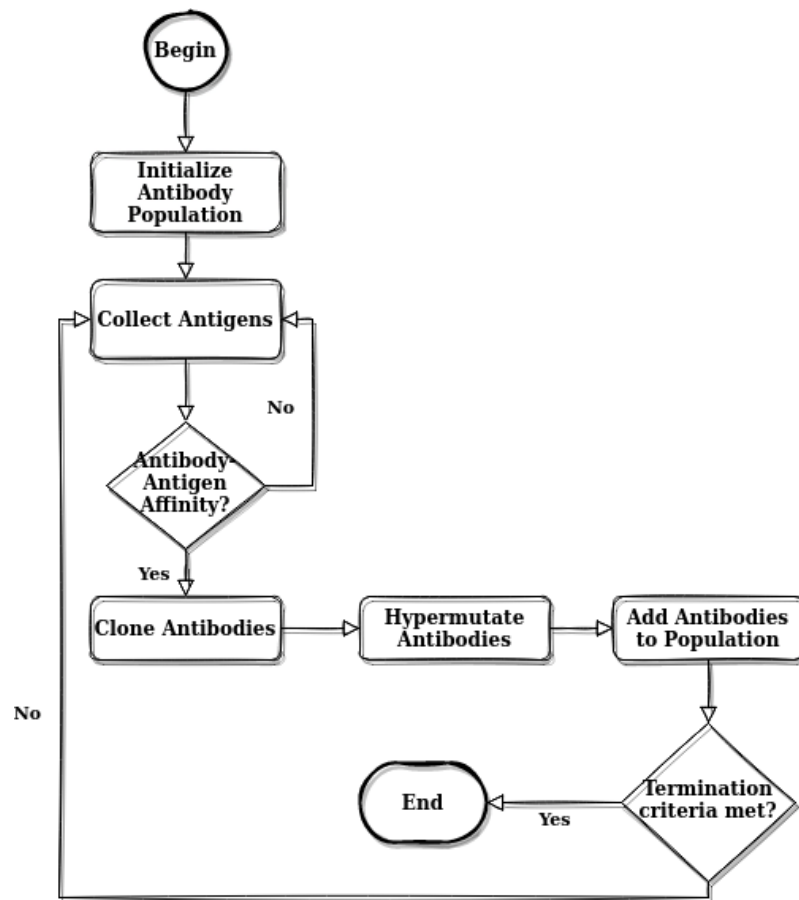


Figure 3.7: Flowchart of the *Clonal Selection Algorithm*.

includes various characteristics, including changeable *Antibody* population size, exploitation and exploration in the search space, finding of several ideal *Antibodies* (based on affinity evaluation), and halting criteria determination. It begins with random initialization of the *Antibody* population, followed by an evaluation of the affinity between each *Antibody* and a pattern. A pattern is equivalent to a *Antigen* in this context. The best highest affinity *Antibodies* are then selected, and numerous copies of these *Antibodies* are created. Following cloning, the duplicates will undergo hypermutation proportionate to their attraction for the cloned pattern. The altered copies will be introduced to the *Antibody* population, and many matured *Antibodies* will be reselected for memory. Finally, the procedure is repeated until a termination criterion is reached. Figure 3.7 displays an *CSA* flow chart. Some well-known clonal selection theory-inspired strategies were suggested, including *AIRS*²⁵ [157], *BCA*²⁶ [158] and the Adaptive Clonal Selection [159].

²⁵The Artificial Immune Recognition System

²⁶The B-Cell Algorithm

3.3.2.3 Immune Network Model

Inspiration The AIN²⁷ is also influenced by the clonal selection theory of acquired immunity, with a focus on highly dynamic interactions between immune cells. In 1974, Jerne [160] developed the immune network theory, also known as idiotypic-network theory, which argues that the immune system is more dynamic than previously thought. Instead of waiting for invading *Pathogens* to begin and promote an immune response, this dynamism refers to the immune system continually stimulating itself. According to the theory, immune cells identify both intruder *Pathogens* and other immune cells, resulting in a dynamic network of immune cells.

The immune system, according to the immune network hypothesis, maintains an idiotypic network of linked *B-cells* for *Antigen* detection and immunological memory. An *Idiotype* is a set of *Antibodies* that have a similar feature, indicated by their common variable region, that makes the *Antibodies* a complimentary match and *Antigen* specific. This *Antigen* specificity of *Antibodies* is commonly referred to as *Idiotype*. Complementary match refers to the formation of an opposing shape or structure that corresponds to the *Antigen Epitope* (antigenic determinant, i.e., the part of an *Antigen* that is recognized by the immune system).

Antibodies stimulate and repress each other in order to keep the network and memory stable. *Idiotypes* are triggered when a group of *Antibodies* connect to *Idiotypes* from another group of *Antibodies* or a specific *Antigen Epitope*. As a result, the concentration of *Antibodies* from the *Idiotype* rises, whereas the concentration of *Antibodies* from the recognized group falls. According to the notion, even in the absence of *Antigens*, contact and ongoing communication between *Antibodies* remains, therefore any cell can indirectly activate or repress immune responses.

Immunologists do not generally agree with the immunological network theory. Unlike other models, which abstract a simpler version of biological processes, immunological network theory seeks to represent a complicated system in an even more complex manner. In reality, other models may represent the previously described network dynamics. Figure 3.8 illustrates the immune network theory based on the Richter model [14], i.e., how *B-cells* and *Antibodies* react to *Antigen*, even though the majority of those cells do not directly react to those specific *Antigens*.

Algorithm Overview As previously said, the AIN draws influence from clonal selection theory, as seen in Figure 3.7. This algorithm attempts to investigate the interconnectivity of different immune cells, i.e., cell populations linked over successive generations, to stimulate or suppress immune responses. As a result, the *Antibody* population does not evolve in the way that the CSA predicts. There is also a considerable interaction between all *Antibodies* in the population. aiNet²⁸ [161] is a well-known approach inspired by the AIN.

²⁷Artificial Immune Network

²⁸The Artificial Immune Network algorithm

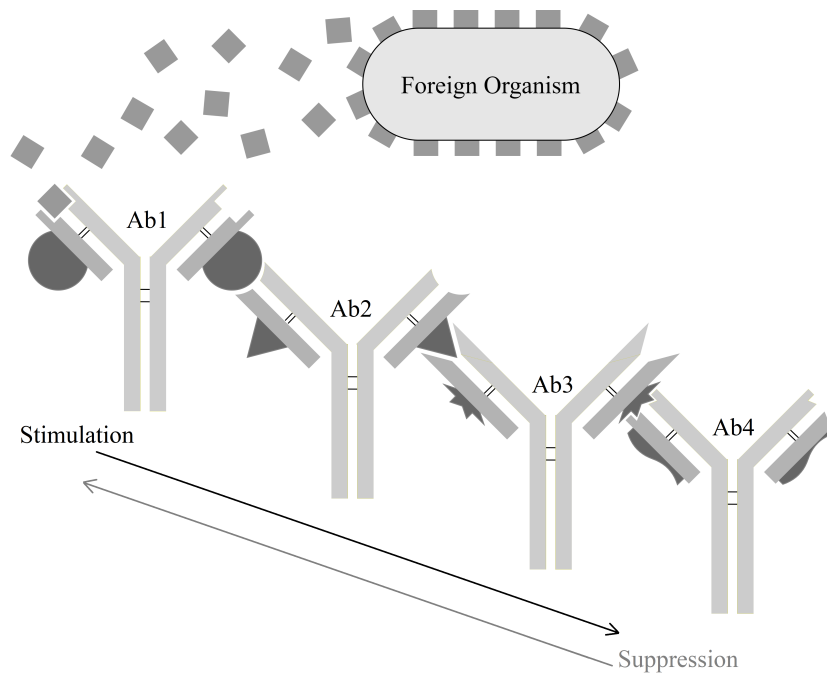


Figure 3.8: Simplified diagram of the immune network theory (Richter model [14]) [3]. *B-cells* with receptors of a given *Idiotype* (specific type of *Antibody* are capable of mutual interaction if their receptors have complementary specificity. Hence, the entirety of the *B-cell* system forms a functional network, which can stimulate or suppress a given immune response.

3.3.2.4 Danger Theory

Inspiration Matzinger [162] first suggested the **DT**, which explains how an immune response is generated while overcoming fundamental flaws noted in previous theories such as the **NSA**. Classical ideas, she claims, cannot explain why there are auto-immune illnesses, i.e., the **BIS** reacts to self-cells. Also, it does not explain why the **BIS** avoids immune responses to external organisms (non-self) that are harmless or even beneficial to the human body, such as gut microbiota.

In contrast to prior immune models, the **DT** introduces novel insights about innate immunity, whereas others exclusively include adaptive immunity. According to Matzinger, contrary to other ideas, the 'foreignness' of a *Pathogen* is not the essential element that causes an immune reaction, and 'selfness' is no guarantee of tolerance. A co-stimulatory signal from specific *Dcs*²⁹, which are a form of *Granulocytes* as previously described, also known as *APCs*³⁰, initiates the immune response. Injured cells, such as those exposed to *Pathogens*, send out danger/alarm signals, which activate *APCs*. These *Danger signals* are activated by *Necrosis* processes, which occur when cell death is not planned, such as in the case of illness, trauma, or infection. Healthy cells, on the other hand, or cells undergoing natural physiological death (*Apoptosis*), should not send warning signals. When triggered, *Dcs* provide a co-stimulatory signal to the immune system, causing it to

²⁹Dendritic Cells

³⁰Antigen Presenting Cells

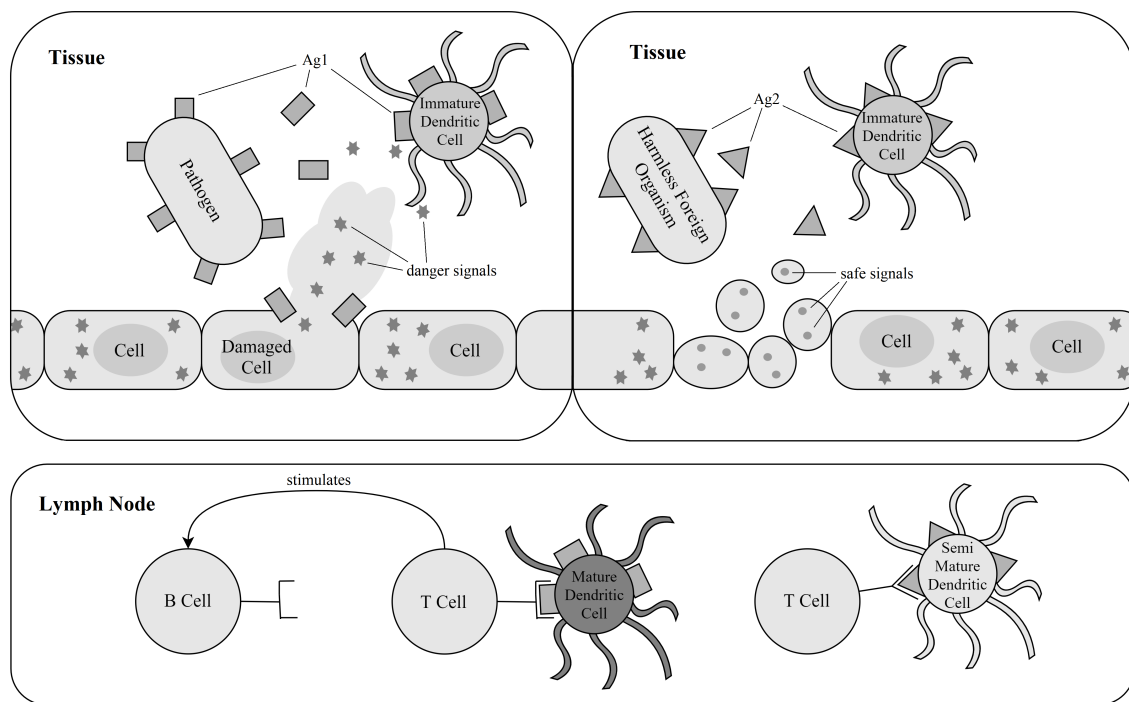


Figure 3.9: Simplified diagram of the *Danger Theory* [3]. Initially, *immature Dendritic Cells* navigate throughout the body, collecting encountered *Antigens* without discriminating between self or non-self. While navigating, *immature Dendritic Cells* also collect signals emitted by dying tissue cells. Damaged cells release chaotically their contents, which are identified as *Danger signals* by neighboring *immature Dendritic Cells*. On the other hand, when cell death is programmed, *immature Dendritic Cells* still collect *Antigens* while recognizing *Safe signals*. After collecting a sufficient amount of signals, *immature Dendritic Cells* evolve into either mature or semi-mature *Dendritic Cells*, depending if the overall context during *Antigens* collection was dangerous or safe. In both cases, *Dendritic Cells* migrate and present their collected *Antigens* to *T-Cells*. *T-Cells* that recognize the *Antigens* presented by *mature Dendritic Cells* and proceed to initiate an immune response, or they can tolerate *Antigens* if presented by *semi-mature Dendritic Cells*.

respond in the danger zone around the wounded cell. A *Dc* [163] is the most significant immune cell considered in the *DT*. Figure 3.9 depicts a simplified model of the *DT*.

In these types of algorithms, the role of *Dcs*, initially recognized by Steinman & Cohn [164], is inspired by innate immunity. *Dcs* are generated in the bone marrow and circulate through the blood, where they are dispersed in many tissues. When *Dcs* are activated by *Antigens* in the tissue, they mature and differentiate to the particular tissue. The identification of signals inside the tissue, especially *Danger signals* (produced by tissue cell damage), *PAMP*³¹, i.e., pre-defined bacterial signatures, *Safe signals* (induced by controlled cell deaths), and *Inflammatory signals* (general tissue distress), facilitates the level of maturation of *Dcs*.

So, there are three main types of *Dcs*: i) *immature Dendritic Cells* - collect parts of the *Antigen* and the signals; ii) *semi-mature Dendritic Cells* - *Dcs* that migrated in a tolerance context, i.e., their

³¹ *Pathogen-Associated Molecular Pattern*

local signals represent safe and *Antigen* presentation to *T-Cells* results in tolerance; iii) *mature Dendritic Cells* - *Dcs* that have local signals that represent danger, so *Antigen* presentation to *T-Cells* results in a reactive response. After capturing the *Antigens*, mature *Dcs* or semi-mature *Dcs* move to local lymphoid nodes and offer the collected *Antigens* to *T-cells* in order to trigger or tolerate an immunological response. The immature *Dcs* are not capable of stimulate *Lymphocytes*, which are influenced by the level of maturation of *Dcs*.

Algorithm Overview The algorithm's main purpose is to produce a collection of mature and semi-mature *Dcs* that give context specific information about how to identify input patterns (*Antigens*) as normal or abnormal. Some algorithms, such as the *DCA* and *dDCA*³² (optimized version of the *DCA*), focus on the functionality of *Dcs*, from signal and *Antigens* exposure through cell maturation and evaluation of cell migration context based on the gathered *Antigens*. Other algorithms, such as the *TLR*³³ [165], address the cell migration process thereafter, i.e., the *Dc* contact with *T-cell*, in which *Dcs* present *Antigens* to *T-cells* by establishing a *T-cell* matching metric. Furthermore, additional algorithms, such as the Artificial *NKC*³⁴ [166], are based on the recently published *Natural Killer cell* immunological theory. Figure 3.10 depicts a flowchart of the primary functions of a generic *DCA* form. More information about *DCA* is available in Section 3.4.

3.4 The Dendritic Cell Algorithm

The *DCA* [137, 163] is an abstraction of the *DT* that is based on the known functions of the biologic *Dc* of the innate immune system and is mainly employed as a binary classification algorithm. The *DCA* classifier has been successfully deployed to a broad range of real-world applications, resulting in a number of *DCA* versions aimed at modifying and enhancing the original proposal. One of these variants is the *dDCA* [167], which removes most of the stochastic parameters and random-based aspects while keeping all of the theoretical underpinnings underlying the method intact. The basic terminology used in this work for the *DCA* and its variations is represented in Table 3.1.

As previously stated, the biologic *Dc* is a sort of *APC* in the *BIS* that is responsible for capturing, processing, and disclosing *Ags* to *T-cells*. An *Ag* is an outside molecule that represents a *Pathogen*, such as bacteria or viruses, entering the body. Its presence usually elicits an immunological response, which begins with the reconnaissance duty of *Dcs*. *T-cells* are in charge of suppressing or activating immune responses, as well as directing the creation of *Antibodies*, which are employed to neutralize certain *Pathogens*. The kind of immune response is determined by the *Ag* revealing process from *Dcs* to *T-cells*. For example, if the *Dc* is supplied in a mature condition, *T-cells* will react to the *Ags* processed by that *Dc*. If the *Dc* is supplied in a semi-mature condition, the *T-cell* will tolerate the respective *Ags*.

³²deterministic Dendritic Cell Algorithm

³³Toll-Like Receptor

³⁴Natural Killer Cell

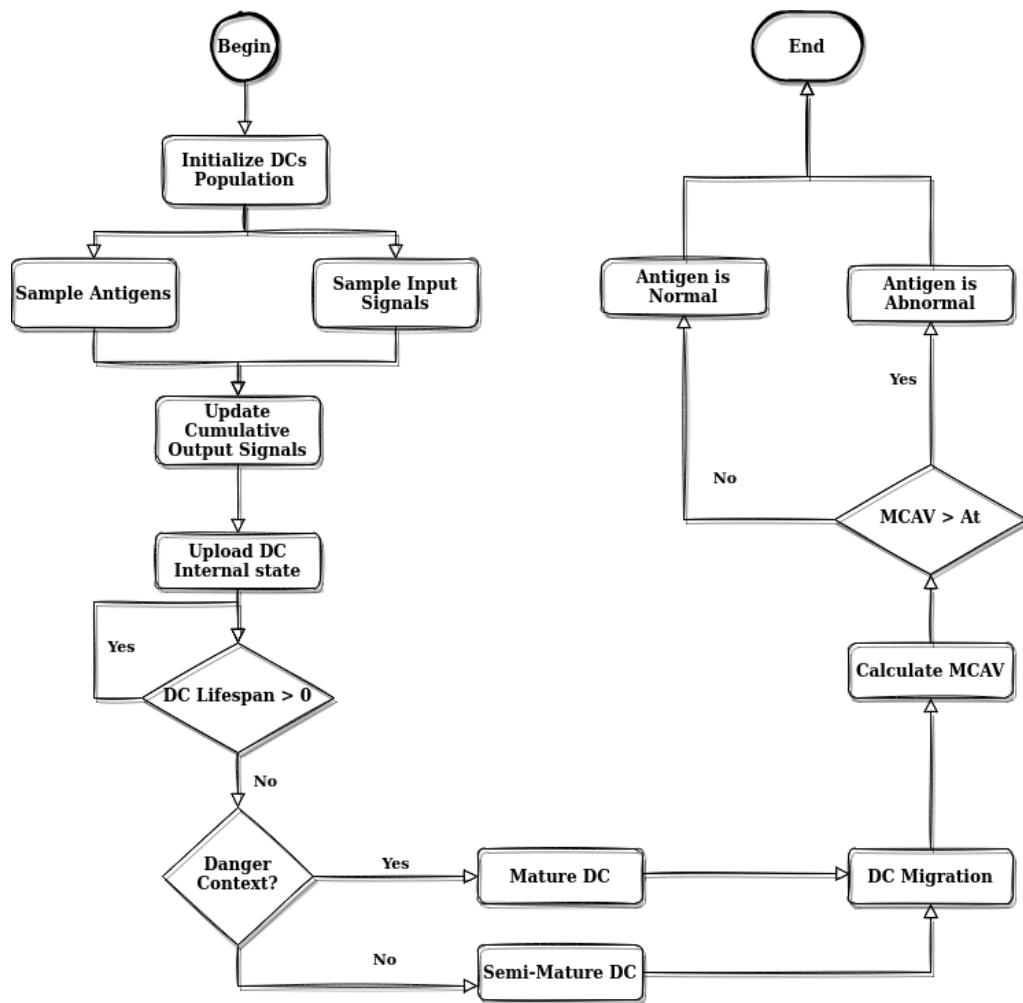


Figure 3.10: Flow Chart of the main functionality of the *Dendritic Cell Algorithm*.

The status of a *Dc* is heavily influenced by the signals detected in the surrounding environment. Each *Dc* has receptors on its surface to receive messages from its surroundings. These signals include: i) *PAMP* - indicates the presence of danger with high confidence; ii) *Ss* - indicates the absence of danger with moderate confidence; iii) *Ds* - indicates the presence of danger with low confidence; and iv) *Ck*³⁵ - increases the magnitude of all previous signals, thereby increasing their effect. As previously stated, the creation of input signals is heavily reliant on both the *Apoptosis* and *Necrosis* processes. In this context, *Sss* are formed in greater quantities during an *Apoptosis* process, indicating that the *Ags* gathered by *Dcs* were discovered in a normal context. On the other hand, in a *Necrosis* process, more *Dss* and *PAMPs* are created, suggesting irregularity and the need for immune activation.

Chelly and Elouedi [168] provided a formalization of *DCA* based on the abstract model of the biologic *Dcs*, which is separated into four key phases:

³⁵ *Inflammatory Cytokines*

Table 3.1: *Dendritic Cell Algorithm* main nomenclature.

Immune Component	Representation
<i>Dendritic Cell</i>	<i>Dc</i>
<i>Antigen</i>	<i>Ag</i>
<i>Pathogen-Associated Molecular Pattern</i>	<i>PAMP</i>
<i>Safe Signal</i>	<i>Ss</i>
<i>Danger Signal</i>	<i>Ds</i>
<i>Co-stimulatory signal</i>	<i>csm</i>
<i>Semi-mature Dendritic Cell Cytokine Cytokines</i>	<i>semi</i>
<i>Mature Dendritic Cell Cytokine Cytokines</i>	<i>mat</i>
<i>Context Output Value</i>	<i>k</i>
<i>Mature Context Antigen Value</i>	<i>mcav</i>
<i>Anomaly Threshold</i>	<i>At</i>

- **Pre-processing & Initialization:** This stage focuses on data pre-processing, specifically feature selection from training data, normalization, and input signal categorization. The signal categorization consists on mapping selected features to *Ss*, *Ds*, and *PAMP*s signals, and generation of *Ag*s or patterns to be classified, based on the available training data. If a typical tabular shape ML dataset is considered, each row corresponds to a different *Ag*. The process of obtaining various input signals from data points in a dataset is confusing and empirical. To determine what and how engineering elements are suitable to describe signals such as *Danger* and *Safe*, prior knowledge of the problem/context is required.
- **Detection:** This step focuses on the fusion of collected input signals (*Ss*, *Ds*, and/or *PAMP*) and *Ag*s. The detection is made feasible by utilizing an immature *Dc* population that samples distinct *Ag*s many times and combines them with the input signals.
- **Context Assessment:** This step focuses on signal output computation and the distinction of migrating *Dcs* into mature or semi-mature status. This resulted in evaluating the *Ag* sampling context after the detection stage. Cumulative output signals are updated in this case using Eq. (3.1), notably the *csm*, *semi* and *mat*, which will be utilized to assess the context of the *Dcs* and the related sampling context (mature or semi-mature). As a result, Eq. (3.1) is used to combine the input signals to obtain values for each of the three output concentrations, where C_x is the input concentration and W_x is the weight. The weight values are empirically pre-defined based on several problem/context circumstances. These are utilized to provide distinct input signals with more or less impact in the computation of the output signals, based on the confidence level of such input signals. Also, an example of suggested weighting values for signal processing is represented in Table 3.2.

$$C_{[csm,semi,mat]} = \frac{(W_{PAMP} \times C_{PAMP}) + (W_{Ss} \times C_{Ss}) + (W_{Ds} \times C_{Ds}) \times (1 + IC)}{W_{PAMP} + W_{Ss} + W_{Ds}} \quad (3.1)$$

Table 3.2: Example of suggested weighting values in *Dendritic Cell Algorithm*.

W_x	<i>csm</i>	<i>semi</i>	<i>mat</i>
<i>PAMP</i>	2	0	2
<i>Ds</i>	1	0	1
<i>Ss</i>	2	3	-3

- **Classification:** The final stage uses the *Dc* context resulting from the previous step to derive the nature of the response, by measuring the proportion of *Ag*s presented by *Dc*s that are fully matured. The *mcav* is used to determine the degree of abnormality of a given *Ag* (in conjunction with an *At* for binary classification). The *mcav* may be computed using Eq. (3.2). $\#mAg$ ³⁶ represents the number of mature antigens (a specific *Ag* sampled by fully matured *Dc*), and $\#Ag$ ³⁷ represents the total quantity of antigen presented (overall sampling of the *Ag*s by any *Dc* that migrated).

$$MCAV(Ag) = \frac{\#mAg}{\#Ag} \quad (3.2)$$

3.4.1 The deterministic Dendritic Cell Algorithm

As seen before, the *DCA* is a population-based algorithm inspired by the interaction between *Dcs* (system agents) and their environment. Thus, the *DCA* can integrate numerous input signals to assess the present context of the environment. The original implementation of the *DCA* is a highly stochastic algorithm due to the number of random-based elements. These are some examples of random-based elements: i) The pre-defined weight parameters W_x used in the *Context Assessment* stage (the way cumulative output signals are calculated) are very context-dependent; ii) The overlapping nature of some input signals, such as *Danger* and *PAMP*, or simply by considering other secondary signals, such as *Ck*, may impose difficulties in extracting all signals from data points during the *Pre-processing & Initialization* stage; iii) output context value of an individual *Dc*, i.e., *semi* and *mat* are redundant and can be reduced to only one factor in the *Context Assessment* stage.

As a result, Greensmith *et al.* [167] suggest the deterministic version of the *DCA* (*dDCA*) to better examine its performance by removing most of the stochastic parameters in the algorithm. This algorithm's anomaly identification is based on the context and antigen correlation. Input signals to the *dDCA* are pre-classified depending on their origin, which can be reduced to only two signals (an activating signal and an inhibitory signal):

- Ds ³⁸ - Activating signal. The presence of *Danger signals* may indicate an anomalous situation. However, the probability of an anomaly is higher than under normal circumstances.

³⁶Quantity of Mature Antigen

³⁷Quantity of Antigen Presented

³⁸Danger Signal

- Ss^{39} - Inhibitory signal. This signal increases in value in conjunction with observed normal behavior. Hence, the presence of *Safe signals* almost certainly indicates that no anomalies are present.

The csm^{40} and the k^{41} , which indicate the concentration of semi-mature and mature signals, are the output signals. The k takes the place of both *semi* and *mat* calculations. These output signal concentrations are calculated using Eq. (3.3).

$$\begin{bmatrix} csm \\ k \end{bmatrix} = \begin{bmatrix} Ds + Ss \\ Ds - 2Ss \end{bmatrix} \quad (3.3)$$

The csm output signal is utilized in Eq. (3.3) to identify when a Dc has reached the end of its lifespan and is thus ready for migration. The k output signal determines the context of Dcs . Suppose the value of k is larger than zero. In that case, the Dc is given a context value of 1, implying that the Ag gathered may be abnormal. Otherwise, if k is less than zero, the Dc is assigned a context of 0, suggesting that the Ag gathered is most likely normal. Pseudocode for the $dDCA$ is given in Algorithm 1. Despite having less random-based variables, some parameters for the $dDCA$ still need to be supplied, namely:

- $\#Dc^{42}$, used to define the size of cell population.
- $\#Iter^{43}$, used during the migration process within the *Context Assessment* phase.
- At , used in the *Classification* phase for a binary classification.
- t_m^{44} , used to define the maximum lifespan a cell can have when created.

The $dDCA$ begins by creating Ags , and input signals Dss and Sss based on the data given. Each Ag s is the data tuple to be classified, and the input signals are the data tuple's features. The $dDCA$ then continues to initialize every Dc in an immature state, which includes initializing the Dc profile, such as: the output signals csm and k as 0, the lifespan (estimated time to live of the Dc) as a random number within a minimum and maximum threshold values, and an empty pool of sampled Ags .

Then, in a process known as Dc migration, Ags created will be sampled by Dcs . The cumulative output signals are updated using Eq. (3.3). Each Dc profile is revised, specifically incrementing its own csm and k with the cumulative output signal results, decreasing the lifespan by the value corresponding to the csm , and updating the pool of sampled Ags correspondingly.

If the lifespan of a certain Dc hits zero throughout this procedure, the Dc migrates, which indicates it exits the immature stage. The k value of the Dc is examined to determine if it enters a mature or semi-mature condition. If k is positive, the Dc migrates as mature. Otherwise, it

³⁹Safe Signal

⁴⁰Co-stimulatory Signal

⁴¹Context Output Value

⁴²Number of Dendritic Cells

⁴³Number of Iterations

⁴⁴Migration Threshold

Algorithm 1: Pseudocode of the *deterministic Dendritic Cell Algorithm* implemented in this Thesis.

```

input      : #Dc, #Iter, At, tm
generate: Ag, Ds, Ss

1 Ag ← generate: Ds, Ss
2 for i ≤ #Dc do
3   | Dc ← InitializeDC (tm)
4 for j ≤ #Iter do
5   | forall Ag do
6     | forall Dc do
7       | ExposeCell (Ag)
8       | csm ← Ds + Ss
9       | k ← Ds − 2Ss
10      | if Dc.lifespan ≤ 0 then
11        | M ← Migrate (Dc)
12        | Reincarnate (Dc, tm)
13        | if k ≥ 0 then
14          | Dc.state ← mat
15        | else
16          | Dc.state ← semi
17 forall Ag do
18   | forall M do
19     | #Ag ← Count (Ag)
20     | if Dc.state == mat then
21       | #mAg ← Count (M)
22   | mcav ← #mAg/#Ag
23   | if mcav ≥ At then
24     | context ← 1
25     | Ag ← 'Anomaly'
26   | else
27     | context ← 0
28     | Ag ← 'Normal'

```

migrates as semi-mature. The *Dc* migration is repeated for the number of times specified in the *#Iter*.

Finally, it follows the *Ag* categorization step of the dDCA. The number of times a relevant *Ag* was sampled by a migrating *Dcs* is assessed. The number of *Dcs* in a mature state is also counted. Finally, the *mcav* is computed by dividing the number of *Ag*s sampled by mature cells, by the total number of sampled *Ag*s. Suppose the *mcav* exceeds the stated *At*. In that case, the *Ag* sampling context is 1, indicating that the corresponding *Ag* is likely to be anomalous. Otherwise, if the context is 0, it indicates that the *Ag* is likely normal.

3.5 Conclusion

In the existing literature, applications of AISs include computer security, anomaly detection, fault-tolerant control, pattern recognition, clustering, and optimization. These types of techniques fall under the *Biologic Inspired Computation* field and contribute significantly to enabling autonomy in engineering/computer systems, such as industrial processes.

Although a system does not need to be inspired on biological approaches to be Self-immune, most of the approaches that introduce Self-immunity capabilities are based on AISs, because they enable learning, adaptation and memory features. The relationship is natural since AISs applications are mainly concerned with anomaly/intrusion detection, thus keeping the system safety proprieties when facing malicious external actions. Also, while detecting anomalies/intrusions, AISs can develop tolerance over time due to its immune memory properties. This can be viewed as a bio-inspired learning component addressing the algorithm's continuous adaptation to normal system conditions to effectively tackle anomalous and harmful actions.

This chapter's main contribution is the comprehensive presentation of the main immune-inspired techniques, such as NSA, CSA, AIN and DT. Considering the anomaly/intrusion detection problem, DT is the most recent immune-model proposal, which addresses several biological and engineering limitations of the previous immune models, especially NSA. Since the most expressive DT-based algorithm is the DCA, a particular focus is given to a detailed description of DCA and its most known variation, the dDCA. DCA is used for developments described in further Chapters.

Part II

Research

Part II - Research: Introduction

Part II presents the contributions of this Thesis. It starts by providing a *Systematic Literature Review* about existing applications of different immune models in *Smart Manufacturing Systems*.

Then, requirement analysis and solution specification are provided, considering the research strategies related to *Intrusion Detection Systems* based on the *Dendritic Cell Algorithm*. With this in mind, the following sections focus on presenting our newly developed approaches regarding immune-based industrial *Intrusion Detection System* and a *Function Block*-based *Anomaly*

Intrusion Detection System. The first contribution describes the proposal of a new industrial-oriented dataset for network intrusion detection problems. The second contribution is a new variant of the *Dendritic Cell Algorithm*, designated *Incremental Dendritic Cell Algorithm*, suitable for online data processing and classification. The third one describes a *Function Block* architecture, based on the IEC 61499 standard, for the *Incremental Dendritic Cell Algorithm* deployment in a *Cyber-Physical Production System*. Finally, a comprehensive experimentation design is followed, to validate both *deterministic* and *incremental Dendritic Cell Algorithms* performances, and the *Anomaly Host Intrusion Detection System* Pipeline architecture to be deployed in the *Cyber-Physical Production System* Edge layer.

Chapter 4

Application of AIS in SMS

4.1 Introduction

In this chapter, the reader will learn about the AISs¹ applications in SMSs² reported in the scientific literature over the years. On the one hand, as seen in Chapter 2, CPPSs³ have reached a level of complexity, where the human intervention for operation and control is becoming increasingly difficult. Thus, SMSs require increasingly autonomic methodologies for management. On the other hand, considering Chapter 3, *Autonomic Computing*-related approaches, such as AISs, allow the system to adapt itself, eliminating the need for human intervention.

So, the objective of the SLR⁴ presented in Section 4.2 is two-fold. First, the study collects and analyses the scientific contributions that report industrial AIS applications on multiple research areas to tackle different problems. Second, these applications are assessed based on the TRL⁵, and LOA⁶ introduced to the system, taking into account a taxonomy of system autonomy presented by Peres *et al.* [125]. To this end, we intend to find answers to the following research questions:

RQ T.1 *What is the research trend of Artificial Immune System applications in Smart Manufacturing Systems?*

RQ T.2 *What is the status of Artificial Immune System applications in Smart Manufacturing Systems?*

RQ T.3 *What are the main motivations for using Artificial Immune Systems in Smart Manufacturing Systems?*

RQ T.4 *What are the research gaps and future areas within Smart Manufacturing Systems based on Artificial Immune System solutions?*

¹Artificial Immune Systems

²Smart Manufacturing Systems

³Cyber-Physical Production Systems

⁴Systematic Literature Review

⁵Technology Readiness Level

⁶Level of Autonomy

As the main output of the study presented in this chapter, we proposed a conceptual framework (Appendix B) to bridge the gap between research in the AIS and SMS fields while discussing key challenges and opportunities to be addressed by future research. With this research study, we aim to build a body of knowledge for researchers and manufacturers regarding AIS solutions under the umbrella of SMSs, while suggesting directions for understanding the requirements for designing and managing autonomic industrial applications supported by AISs. Section 4.3 concludes the chapter by resuming the study findings used to provide answers to the research questions mentioned before.

Partial results of the presented work in this chapter have been submitted at:

- Pinto, R., & Gonçalves, G. (2021). Application of Artificial Immune Systems in Advanced Manufacturing. In Array. Elsevier, (In Press). [1]

4.2 Systematic Literature Review

We investigated the prior literature review work on both SMSs and AISs topics to underline the importance of this SLR in the scientific community. Table 4.1 provide the comparison of similar work.

In terms of SMSs, various review-type publications addressing different views of I4.0 have already been published. Zhong *et al.* [177] gave an overview of I4.0 related international movements and governmental initiatives. The authors outline linked subjects and major technologies, such as IoT, CPSs, Cloud Computing, Big Data, and ICT⁷, which are employed to allow intelligent manufacturing. This study was conducted at a period when the notion of I4.0 was still in its infancy. Later, Oztemel and Gursev [25] provided a full assessment of I4.0 and associated technologies, defining the idea explicitly and providing a taxonomy of I4.0, which can be used to assist the application of I4.0 design principles. Furthermore, Qu *et al.* [178] gave a complete analysis of SMSs, summarizing the evolution, definition, objectives, functional requirements, business needs, technological requirements, and components of SMS. More recently, Stock *et al.* [181] highlighted how current ICT architectures and infrastructure for CPPSs might allow their inherent Self-* capabilities to pave the way to greater degrees of autonomy aided by data-driven technologies.

In light of SMS-related SLRs, Peres *et al.* [125] gave a thorough evaluation of existing Industrial AI literature, with an emphasis on its application in real industrial contexts. Radanliev *et al.* [180], on the other hand, gave a literature analysis of existing and future problems in the usage of AI in CPSs, and propose a novel conceptual framework for analyzing the evolution of AI decision-making in these types of systems.

Furthermore, numerous review-type studies have received attention in the AIS research area thus far. Timmis [169] reviewed the status of AIS techniques and reflected on the field hurdles for future growth to examine available immune models and associated inspired methodologies. Timmis *et al.* [171] analysed AISs in the context of transdisciplinary research, based on an established

⁷Information and Communication Technology

Table 4.1: Comparison of the related work.

Work	Year	Study Type	SMSs	Self-*	AIS	Purpose
Timmis [169]	2007	Position	✗	✗	✓	Immune Models
Shafi and Abbass [170]	2007	Survey	✗	✓	✓	Bio-inspired NIDS
Timmis <i>et al.</i> [171]	2008	Survey	✗	✗	✓	Immune Models
Zheng <i>et al.</i> [172]	2010	Survey	✗	✗	✓	Applications
Dasgupta <i>et al.</i> [131]	2011	Survey	✗	✗	✓	Immune Models
Haktanirlar Ulutas <i>et al.</i> [155]	2011	Literature Review	✗	✗	✓	CSA
Muhamad and Deris [173]	2013	Survey	✓	✗	✓	AIS in Job-shop Scheduling
Bayar <i>et al.</i> [174]	2015	Survey	✗	✗	✓	AISs in FDDR
Bere and Muyingi [175]	2015	Survey	✓	✗	✓	AISs for ICS Security
Raza and Fernandez [176]	2015	Literature Review	✓	✗	✓	AISs in Robotics
Zhong <i>et al.</i> [177]	2017	Literature Review	✓	✗	✗	Generic I4.0
Oztemel and Gursev [25]	2018	Literature Review	✓	✗	✗	Generic I4.0
Qu <i>et al.</i> [178]	2019	Survey	✓	✓	✗	Smart Manufacturing
Aldhaheeri <i>et al.</i> [138]	2020	SLR	✗	✗	✓	AISs for IoT Security
Al-Khatib and Doush [179]	2020	Survey	✗	✗	✓	AIS Applications
Peres <i>et al.</i> [125]	2020	SLR	✓	✓	✗	AI in I4.0
Radanliev <i>et al.</i> [180]	2020	SLR	✓	✓	✗	AI in CPS
Stock <i>et al.</i> [181]	2020	Survey	✓	✓	✗	Self-* CPSs
Alrubayyi <i>et al.</i> [182]	2021	Survey	✗	✗	✓	AISs for IoT Security
This Study	2022	SLR	✓	✓	✓	AISs in SMSs

conceptual framework. This framework includes mathematical and computational immunology modeling, abstraction, and the production of engineering systems. Finally, Dasgupta *et al.* [131] examined the critical works in the AIS research area, with a focus on recent achievements in applied AIS techniques during the previous years. Other studies concentrate on AISs application cases primarily. Zheng *et al.* [172] analyzed immune uses of the AIS approach and provided several recommendations to the AIS community to bring the field forward. Haktanirlar Ulutas *et al.* [155], on the other hand, presented a literature overview of the CSA, its variations (basic and hybrid techniques), and the primary applications. More recently, Al-Khatib and Doush [179] extensively summarized AIS research and categorized it depending on the application problem to

understand the current trend of this category of methods utilization.

There are various AIS review articles in the security sector. Shafi and Abbass [170] examined some major work in the *Complex Adaptive Systems* field, with a primary focus on biologically-inspired adaptive methods, including AISs, to the NIDS challenge. Bayar *et al.* [174] offered an overview of biological immunity and emphasized the key ideas and processes that are especially relevant to FDDR problems. Aldhaheeri *et al.* [138] and Alrubayyi *et al.* [182] more recently offered a SLR and a survey with a complete examination of empirical research on AIS techniques to secure IoT environments. Furthermore, and also linked to SMSs, Bere and Muyingi [175] presented an investigation of AIS methods to protect ICSs against persistent attacks. Finally, Muhamad and Deris [173] examined production scheduling issues, concentrating on how AISs may be utilized to tackle job-shop and flexible job-shop scheduling problems. Raza and Fernandez [176], on the other hand, gave a complete overview of immuno-inspired robotic applications in an attempt to suggest relevant research areas to explore.

To the best of our knowledge, none of the previously mentioned reviews provided insights into the use of AISs approaches in SMSs. In general, we may discover studies that focus solely on the SMSs domain or the AISs domain. Some SMS-domain evaluations also discuss autonomy in industrial systems (Self-*), but do not include AIS applications explicitly in the research. AIS-domain reviews, on the other hand, aim to survey the key immunological models or specialized AIS applications. These application possibilities, however, are unrelated to the industry.

Nonetheless, we came across three identical reviews, which incorporate both the AISs and SMSs domains [175, 173, 176]. These studies, however, are confined to specific application settings and research areas, such as ICT security, job-shop scheduling, and robotics. As a result, there is a need for the proposed systematic review. The major contribution is an extensive analysis of AIS models applied to various application settings and research categories within SMSs. This work will provide readers with a thorough knowledge of the role of AISs in industrial contexts over the last years.

4.2.1 SLR Methodology

This SLR was carried out between July and October 2021 and then updated in July 2022, following the parameters given in the PRISMA⁸ statement. Figure 4.1 depicts the PRISMA flow chart describing the various steps of the systematic review process.

During the *Identification* step, a search string was created using the core principles related with AISs and SMSs. Two groups of keywords were defined, and at least one element from each group was present in the search string. The search string contained a mix of *OR* and *AND* operators relating the different keywords. We then customized the search string for each electronic database used in this study. In this instance, the databases are: i) IEEE⁹ Xplore; ii) Elsevier Scopus; iii)

⁸Preferred Reporting Items for Systematic Review and Meta-Analysis

⁹Institute of Electrical and Electronics Engineers

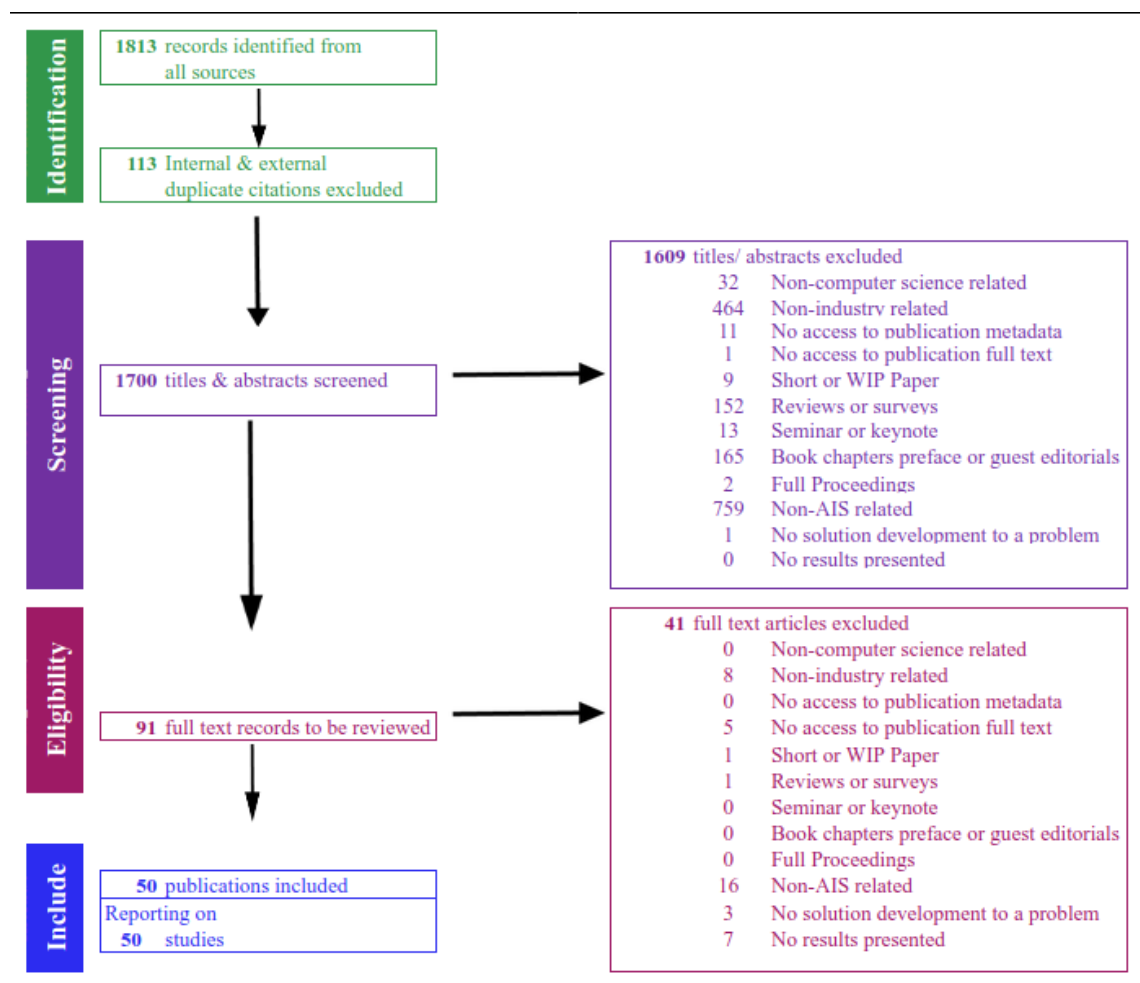


Figure 4.1: *Preferred Reporting Items for Systematic Review and Meta-Analysis* flowchart of study inclusions and exclusions for the *Systematic Literature Review*.

Elsevier ScienceDirect; iv) Springer; v) [ACM¹⁰](#) Digital Library; vi) Engineering Village (Inspec & Knovel); vii) ISI Web of Knowledge.

Appendix A has further information on the search method, i.e., the electronic database search strings employed. The search included academic research that: i) contained at least one term from each group (see Table 4.2) in either the abstract, title, or keywords; ii) was published in peer-reviewed journals, conference proceedings, or book chapters; and iii) was written in English. The entries returned by the search were aggregated and duplicated work was deleted during the *Screening* stage. Following that, one independent reviewer conducted a screening procedure using the eligibility criteria depicted in Figure 4.1.

First, the *Screening* stage was carried out only based on the published metadata, including the title, abstract, and keywords. Following that, in the *Eligibility* step, all remaining articles from the first screening had their whole text assessed in greater depth, depending on the previously provided eligibility criteria. Finally, in the *Include* step, we examined each of the articles suitable for

¹⁰Association for Computing Machinery

Table 4.2: General string queries used in the search string.

Group 1	Group 2
Artificial Immune Systems	Cyber-Physical System
AIS	Cyber-Physical Systems
Dendritic Cell Algorithm	CPS
DCA	Cyber-Physical Production System
	Cyber-Physical Production Systems
	CPPS
	Industrial Cyber-Physical System
	Industrial Cyber-Physical Systems
	Supervisory Control and Data Acquisition
	SCADA
	Distributed Control System
	Distributed Control Systems
	Industrial Internet of Things
	IIoT
	Internet of Industrial Things
	Industry
	Industry 4.0
	I4.0
	Manufacturing
	Manufacturing System
	Manufacturing Systems
	Added Value Manufacturing
	AVM

inclusion in the study according to the research questions stated, taking into account the screening findings.

The eligibility criteria used is presented next:

- | | |
|---------------------------------------|--|
| 1. Non-computer science-related | 7. Seminar or keynote |
| 2. Non-industry related | 8. Book chapters preface or guest editorials |
| 3. No access to publication metadata | 9. Full Proceedings |
| 4. No access to publication full text | 10. Non-AIS related |
| 5. Short or WIP Paper | 11. No solution development to a problem |
| 6. Reviews or surveys | 12. No results presented |

4.2.2 Characterization of AIS Applications in SMS

The literature review results are described in this section concerning the previously established research questions. Following the PRISMA principles and the previously outlined methods, we found 1 813 items via database searches. 113 internal and external duplicate entries were removed,

yielding 1 700 unique records. We filtered the 1 700 items following the exclusion criteria. While just titles/abstracts were screened, 1 609 entries were discarded, leaving 91 full-text papers to be evaluated. 41 articles were eliminated following the full-text screening, resulting in 50 publications in this study. Table 4.3 highlights the outcomes of this evaluation, extracting the critical information needed to answer the previously established research questions.

4.2.2.1 Research Trends

This section attempts to answer the **RQ T.1** *What is the research trend of Artificial Immune System applications in Smart Manufacturing Systems?* by presenting the analysis results of the papers included in the SLR. To gain a broad perspective of the examined literature, we analyzed the papers based on: i) year-wise publication; ii) contributions from publishers; iii) affiliation-wise publishing; and iv) common keywords.

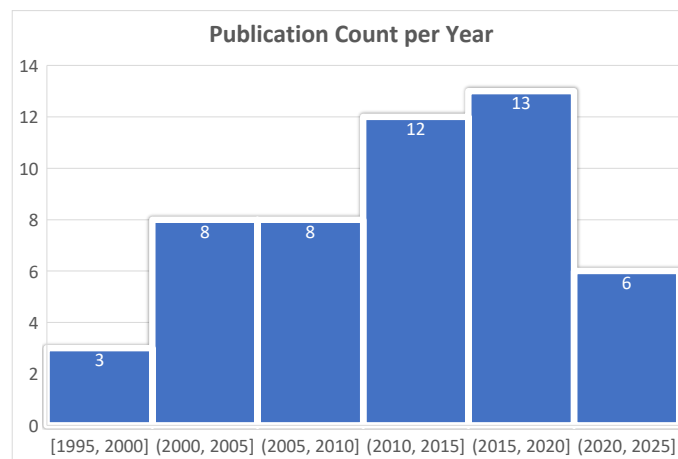


Figure 4.2: Year-wise publications.

Year-wise Publication Analysis Figure 4.2 depicts a year-by-year examination of the publications. The analysis shows a stable trend of published papers between 2000 and 2010, while there was an increasing number of publications after 2010 throughout recent years, especially in the range from 2015 and 2020. Between 1995 and 2010, there was a mean of 1 publication per year, while from 2010 forward, until 2022, there was a mean of 3 publications per year. The AIS research field emerged in the '90s, so it is natural to find studies in the area since 1995. The academic community was more active after 2010 because the emergence of the Industry 4.0 happened in 2011 and other national initiatives in the following years. This implies that the topic of AIS applications in SMSs is attracting more and more attention from the academic community.

Contributions from Publishers The distribution of contributions from various publishers is depicted in Figure 4.3. With an 66 percent share, the IEEE is the biggest contributing publisher. This is understandable given that IEEE is today one of the largest associations of technical experts in

Table 4.3: Overview of publications included in the *Systematic Literature Review*.

Publication	Application Context	Research Category	TRL	LOA	Algorithms
P01 [5]	Manufacturing	IDS	3-4	0	DT
P02 [183]	Manufacturing	Autonomic Systems	3-4	3	NSA, CSA & AIN
P03 [184]	CPS	Anomaly Detection	3-4	0	DT
P04 [185]	Robotics	Operations Research	4-5	4	CSA
P05 [186]	Electric Power Industry	IDS	3-4	0	DT
P06 [187]	SCADA/ICS	Control Systems	-	-	CSA
P07 [188]	CPS	Autonomic Systems	3-4	0	Ensemble
P08 [189]	CPS	Autonomic Systems	3-4	0	Ensemble
P09 [190]	Electric Power Industry	Anomaly Detection	3-4	2	NSA
P10 [191]	Manufacturing	Operations Research	5-6	3	Ensemble
P11 [192]	Electric Power Industry	Fault Diagnosis	3-4	2	NSA
P12 [193]	Electric Power Industry	Anomaly Detection	4-5	2	CSA
P13 [194]	Robotics	Control Systems	5	4	Other
P14 [195]	CPS	Autonomic Systems	3-4	3	DT
P15 [196]	Robotics	IDS	3-4	2	Ensemble
P16 [4]	Manufacturing	IDS	3-4	0	DT
P17 [197]	Manufacturing	IDS	4-5	0	NSA
P18 [198]	Electric Power Industry	Control Systems	3-4	3	CSA
P19 [199]	Manufacturing	Anomaly Detection	5-6	1	Ensemble
P20 [200]	CPS	Operations Research	4-5	3-4	Ensemble
P21 [201]	CPS	Fault Diagnosis	4-5	1-2	Ensemble
P22 [202]	SCADA/ICS	Fault Diagnosis	4-5	1-2	Ensemble
P23 [203]	SCADA/ICS	Control Systems	3-4	3	AIN
P24 [204]	Electric Power Industry	Autonomic Systems	3-4	3	Other
P25 [205]	Electric Power Industry	Operations Research	4-5	2	CSA
P26 [206]	Electric Power Industry	Control Systems	3-4	3	AIN
P27 [207]	SCADA/ICS	Fault Diagnosis	3-4	2	Ensemble
P28 [208]	SCADA/ICS	Control Systems	3-4	3	AIN
P29 [209]	Robotics	Control Systems	3-4	4	AIN
P30 [210]	Electric Power Industry	Control Systems	3-4	3	Other
P31 [211]	Robotics	Control Systems	5	4	Ensemble
P32 [212]	Robotics	Control Systems	3-4	4	Other
P33 [213]	Electric Power Industry	Operations Research	4-5	2	Ensemble
P34 [214]	Robotics	Control Systems	3-4	4	Ensemble
P35 [215]	Robotics	Control Systems	3-4	4	CSA
P36 [216]	Electric Power Industry	Islanding Detection	3-4	1	Other
P37 [217]	Electric Power Industry	Control Systems	3-4	3	CSA
P38 [218]	Electric Power Industry	Operations Research	3-4	1-2	Other
P39 [219]	Robotics	Control Systems	3-4	4	Other
P40 [220]	Robotics	Control Systems	3-4	4	AIN
P41 [221]	Electric Power Industry	Operations Research	3-4	2	CSA
P42 [222]	Robotics	Operations Research	3-4	3-4	AIN
P43 [223]	Robotics	Control Systems	3-4	4	Ensemble
P44 [224]	Electric Power Industry	Control Systems	3-4	3	Other
P45 [225]	CPS	Anomaly Detection	4-5	0	Ensemble
P46 [226]	SCADA/ICS	Control Systems	3-4	3	Ensemble
P47 [3]	CPS	Anomaly Detection	3-4	0	Ensemble
P48 [2]	CPS	IDS	4-5	0	DT
P49 [227]	Manufacturing	Anomaly Detection	3-4	0	Ensemble
P50 [228]	Manufacturing	Fault Diagnosis	4-5	0	NSA

fields such as electrical and electronic engineering, telecommunications, computer engineering, and related disciplines. Elsevier (14%) and Springer (10%) follow as they publish many publications in several journals and conferences in engineering-related subjects each year. Contributions published in the IET¹¹, PSCC¹², ACS¹³, and the State University of Maringá, are out the top three.

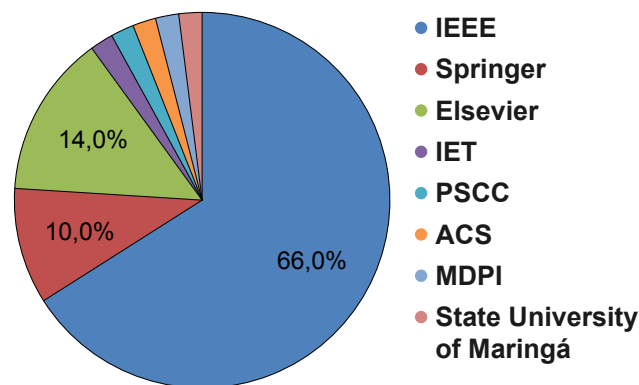


Figure 4.3: Contributions from publishers.

We included journal and proceedings articles to complete the collection and analysis of publications while understanding the article publishing pattern. In this situation, roughly 62% of the manuscripts were published in conference proceedings, with the remaining 38% published in scientific journals. According to *SCImago*¹⁴, 73.6 percent of journals present an impact factor that places them in the Q1 quartile, 10.5 percent Q2, and 15.8 percent Q4. Table 4.4 displays the journals considered as well as the *SCImago* categorization. Also, the considered conferences are graded using the H-Index measure. According to *SCImago*, this ranges from 2 to 71.

Affiliation-wise Publication Analysis The top contributing authors and countries are depicted in Figure 4.4. The study only considers corresponding or first authors of at least two articles. In this example, the most contributing author is *Pinto*, with 4 publications. The other top contributing authors are *Lokesh* and *Lau*, with 3 publications each, and *Lima*, *Li* and *Lee* with 2 publications each. In terms of author contributions by nation, the data reveals that China (40 authors), Brazil (19 authors), Portugal (16 authors), India (14 authors), and South Korea (12 authors) are the top five contributing countries.

Frequent Keywords Analysis Keywords are used to categorize the relevant literature. The distribution of top-rated keywords used in the selected publications is depicted in Figure 4.5. The top five frequent keywords encountered in the publications included in this study, according to the keyword analysis, are *Artificial Immune Systems* (20.6%), *Distributed Control* (9.9%), *Multi-Agent Systems* (9%), *Cyber-Physical Systems* (9%) and *Robot Systems* (6.6%).

¹¹Institution of Engineering and Technology

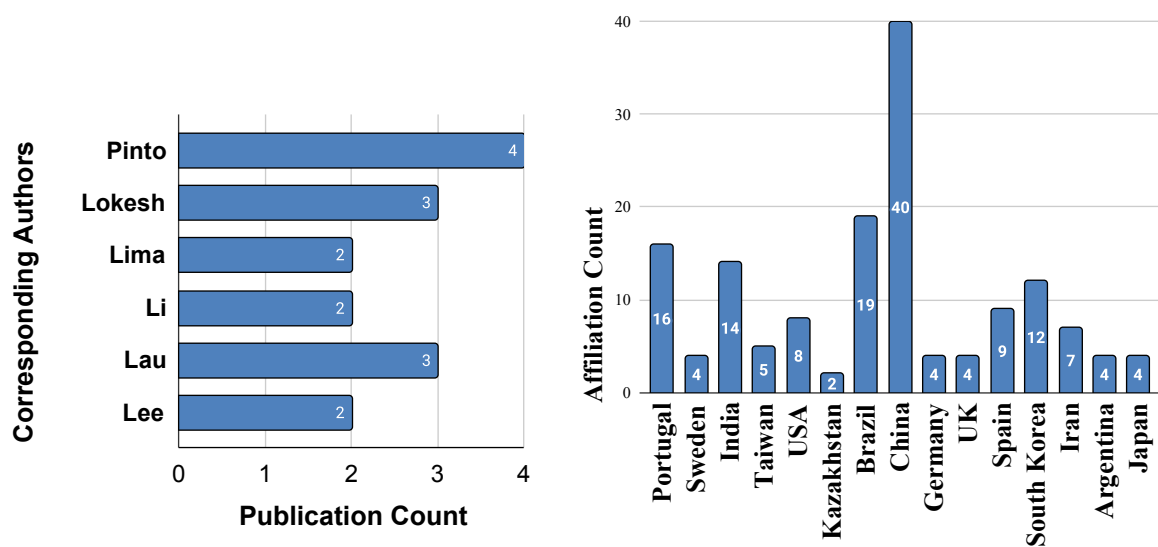
¹²Power Systems Computation Conference

¹³American Chemical Society

¹⁴<https://www.scimagojr.com/>

Table 4.4: Journals encompassed in the study.

Journal	SCImago Classification Quartile
Journal of Intelligent Manufacturing	Q1
IEEE Transactions on Industrial Informatics	Q1
Electric Power Systems Research	Q1
IET Generation, Transmission & Distribution	Q1
IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans	Q1
IEEE Transactions on Industrial Electronics	Q1
Industrial & Engineering Chemistry Research	Q1
Journal of Loss Prevention in the Process Industries	Q1
Applied Soft Computing	Q1
Computers in Industry	Q1
Cybersecurity	Q1
Electric Power Systems Research	Q1
International Journal of Critical Infrastructure Protection	Q2
Algorithms	Q2
Lecture Notes in Networks and Systems	Q4
Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering	Q4
Acta Scientiarum. Technology	Q4



(a) Contributing authors.

(b) Contributions by country.

Figure 4.4: Affiliation-wise publication analysis.

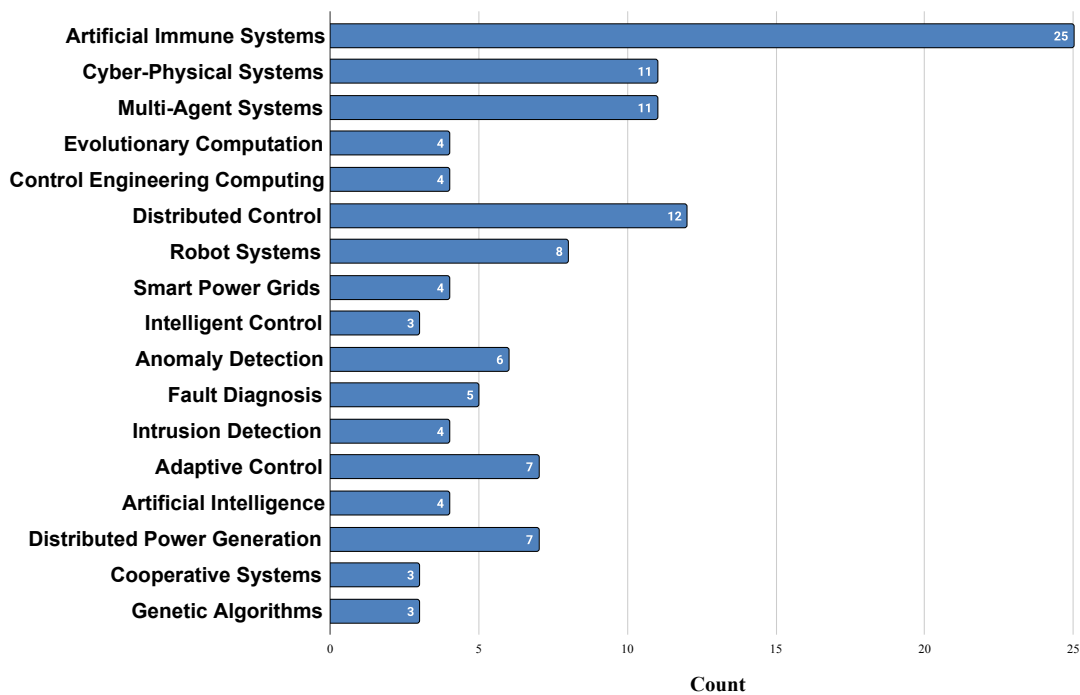


Figure 4.5: Frequently used keywords.

4.2.2.2 Research Fields

This section attempts to answer the **RQ T.2** *What is the status of Artificial Immune System applications in Smart Manufacturing Systems?*. This is done by collecting and analyzing data, such as the primary research categories and application areas found in selected papers, as well as the proposed solution **TRL** and **LOA** introduced into the system by the immune model.

Every article chosen for this **SLR** identifies a problem in a particular scientific field and presents a solution based on immunological processes. In addition, this solution is designed and verified in a specific application scenario. As a result, each manuscript was further classified based on the study topic, application area, and immunological mechanisms. These interactions are mainly based on immune-based techniques used. Considered in the selected publications, the research fields, solution application fields, and immune model utilized are:

Research Fields	Application Fields	Immune Models
<i>Autonomic Systems</i>	<i>Electric Power Industry</i>	<i>Danger Theory</i>
<i>Control Systems</i>	<i>SCADA/ICS</i>	<i>Clonal Selection Algorithm</i>
<i>Islanding</i>	<i>Manufacturing</i>	<i>Negative Selection Algorithm</i>
<i>IDS</i>	<i>Robotics</i>	<i>Artificial Immune Network</i>
<i>Anomaly Detection</i>	<i>CPPS</i>	<i>Immune Inspired</i>
<i>Fault Diagnosis</i>		<i>Ensemble</i>
<i>Operations Research</i>		

The data relationships between research areas and application fields are depicted in the characterization matrix, as illustrated in Figure 4.6.

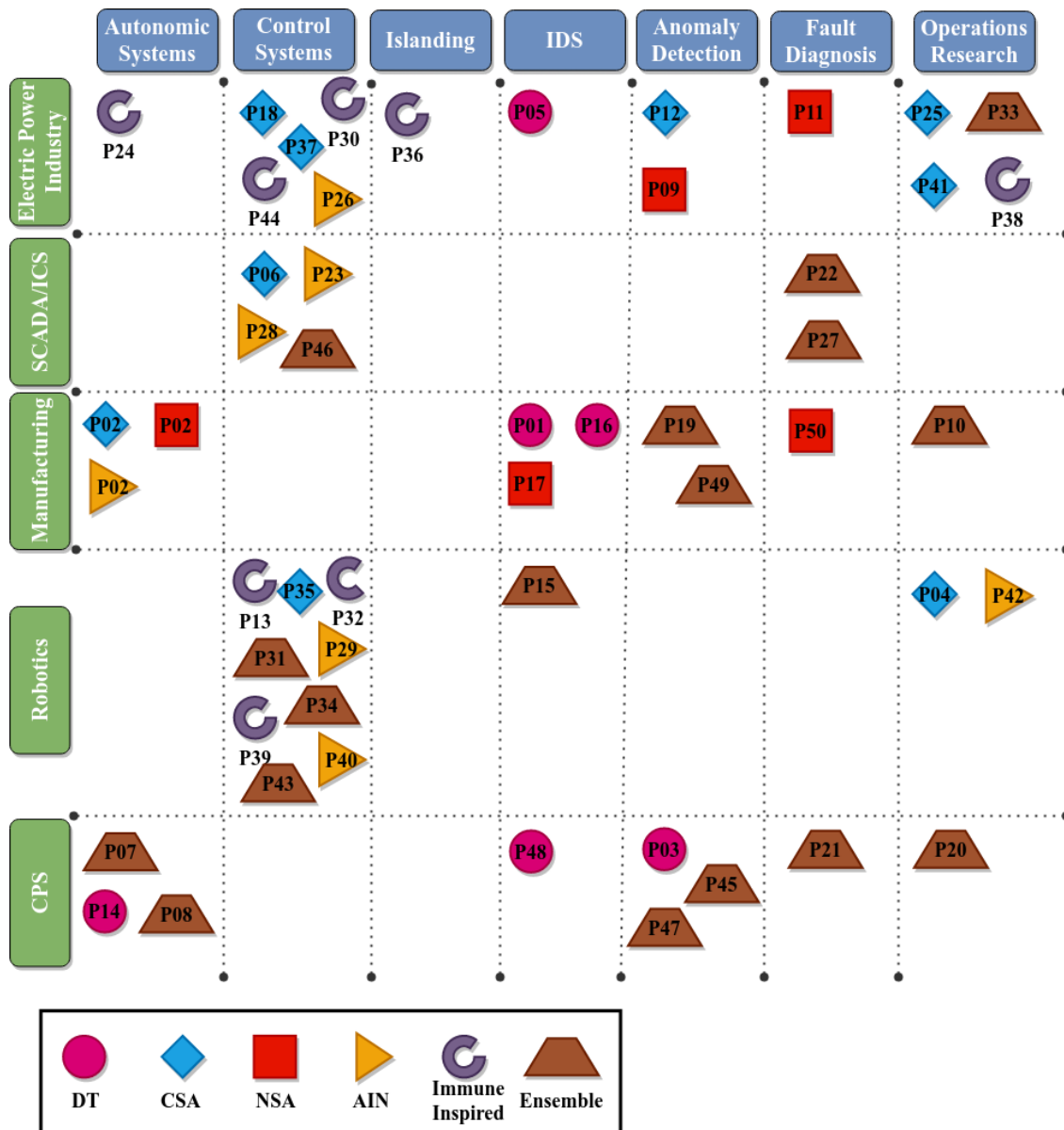


Figure 4.6: Characterization matrix of the studies selected.

Based on the general categorization of the five application sectors, it is feasible to conclude that the *Electrical Power Industry* (30 percent share) has received more attention than the other application areas. Selected research in this subject includes network communication protocols in Smart Grids, electrical distribution systems, DMSs¹⁵, DG¹⁶, and power optimization and management. This is followed by research in the *Robotics* field (24% share), which comprises mobile

¹⁵Distribution Management Systems

¹⁶Distribution Generation

robots, [AGVs](#)¹⁷, robot manipulators, and distributed multi-robot systems. The [CPS](#) field (18% share) takes the third position, which comprises various challenges in cyber-physical processes. The domains *Manufacturing* (16% share) and [SCADA](#) (12% share) come next, which broadly cover difficulties at the industrial process level and control.

On the other hand, it further shows that the various contributions address seven research categories regarding the overall classification of the study disciplines. The majority of the [AIS](#) applications (38% share) were used to develop *Control Systems*, such as overall distributed and cooperative system control and [PID](#)¹⁸ controller tuning. Following that, numerous issues under *Operations Research* (16% share) are discussed, including meta-heuristics, optimization, planning, and scheduling. *Anomaly/Intrusion Detection* (14% and 12% share) and *Fault Diagnosis* (10 percent share) are highly comparable research areas since they rely on similar methods to identify and analyze system assaults and failures. These can be used for system status awareness, online system diagnostics and prognosis, and other purposes. The introduction of Self-* characteristics in industrial systems is referred to as *Autonomic Systems* (10 percent share). Finally, only the *Electrical Power Industry* has a 2 percent share of *Islanding Detection*.

Finally, when it comes to the immune mechanisms used to develop solutions in these various research categories and application areas, most of the contributions (32.7 percent) use an ensemble method, which entails using multiple learning algorithms to achieve better performance than the learning algorithms alone. The ensemble methods in this situation can be a mixture of two or more separate [AIS](#) approaches, other [ML](#)¹⁹ or bio-inspired techniques. The four primary immune inspired mechanisms (including basic and variations) are utilised, notably [CSA](#) - 17.3 percent, [AIN](#) - 13.5 percent, [DT](#) - 11.5 percent, and [NSA](#) - 9.6 percent. Finally, many immune-inspired solutions (15.4%) adhere to one or more immunological principles but do not strictly follow the broad rules of a specific immune model.

Autonomic Systems As previously explained, *Autonomic Systems* in this context refers to incorporating Self-* characteristics into industrial systems. The most common application areas discovered are [CPSs](#), the *Electric Power Industry*, and *Manufacturing*. Overall, all of the ideas found were validated with datasets for proof of concept or small lab testbeds, yielding a [TRL](#) of 3-4. On the other hand, the [LOA](#) of most proposals (using dataset as a mean of validation) is 0 since the suggested solution is shown in a simulated dataset (no system autonomy achieved). Taking into account the work validated on testbeds, a [LOA](#) of 3 is feasible, implying that, despite the main pre-programmed functionality, the system is capable of self-adjusting its own behavior (within certain bounds) in response to the perceived environment condition.

In the context of the [CPS](#) application area, Rammig *et al.* [195] (P14) suggested an online model checking approach that may be used to identify proactively needs to adjust the presently operating software system. For continuous system supervision, a [DT](#)-based technique is adopted

¹⁷*Autonomous Guided Vehicleless*

¹⁸*Proportional Integral Derivative*

¹⁹*Machine Learning*

in this situation. It is feasible to develop a lightweight solution for Self-adaptation in [RTOS](#)²⁰ in this manner. The suggested method is illustrated and evaluated in an abstracted emulation framework in a paravirtualization environment (virtual machines). Lokesh *et al.* [188] (P07), on the other hand, present a [MAS](#)²¹-[CPS](#) model-based integrated with an ensemble immune model (combination of [DT](#) and [AIN](#)). By providing state awareness, self-tolerance, and self-healing, this approach is utilized to incorporate resiliency and self-adaptation into the system. The suggested method is illustrated as a general framework and applied to a simulated dataset containing data from multiple [CPS](#) levels, including monitor & actuator, communication, and computation & control. The authors present a comparable solution to the same problem[189] (P08). However, in this case, the ensemble immune model utilized was a mix of [DT](#) and [AIRS](#)²², which is a method from the [CSA](#) family.

In the *Electric Power Industry*, Li *et al.* [204] (P24) propose a general immune-inspired [MAS](#), designated *Multi-agent Immune Algorithm*, to handle the distribution service restoration problem in the case of a large-scale blackout in a power distribution network with [DG](#). This approach makes the power system gains flexibility, resilience, self-tolerance, and self-healing capabilities. The solution is proven by running a simulated scenario with real-world variables using a power flow analysis program. In the *Manufacturing* application domain, Rocha *et al.* [183] (P2) investigated industrial methods to distributed machine diagnostics employing a [MAS](#) paradigm mixed with [AIS](#). Lightweight [NSA](#), [AIN](#) and [CSA](#) models based on fuzzy logic are created for this work, enabling distributed system flexibility, robustness, and self-tolerance. In simulation settings, the models were shown and tested.

Control Systems *Control Systems* use control loops to govern the behavior of other system elements. In this case, they are primarily found in *Robotics*, but also present in [SCADA](#) & [ICS](#)s and in the *Electric Power Industry*. Overall, the bulk of work is verified in lab contexts (mainly simulations) for proof of concept, with a [TRL](#) of 3-4 achieved. There are, however, two robotics proposals that have been proven in real industry-relevant situations, obtaining a [TRL](#) of 5. In terms of autonomy, every robotics-related work achieves a [LOA](#) of 4 since the robot system is capable of operating as an adaptive and autonomic system, as well as continuous self-learning within recognized system limitations. Proposals connected to [SCADA](#) & [ICS](#)s and the *Electric Power Industry*, on the other hand, obtain a [LOA](#) of 3 since the system is capable of self-adjusting its behavior. Still, a human is responsible for reviewing system decisions.

Robotic immuno-based control systems are becoming increasingly common since, in early years, we could find already ensemble methods (P34, P29, P43), integrating [CSA](#) and [AIN](#) and establishing cooperative control in a distributed autonomic robotic system [214, 209, 223]. Based on the [MAS](#) paradigm, the suggested approach enables adaptive group control as a swarm strategy, where robots may identify and perform tasks that are dispersed across the environment. Validation

²⁰*Real-Time Operating Systems*

²¹*Multi-Agent System*

²²*The Artificial Immune Recognition System*

is carried out in a simulated setting concerning a cooperative robot search problem. Michelan and Von Zuben [220] (P40), on the other hand, study an autonomic control system of a mobile robot based (in this work, non-distributed), utilizing an [AIN](#) model. The robot navigates to complete a multi-objective goal, mainly garbage collecting, while adapting to environmental variables to construct a route free of impediments. The approach was validated using simulation, according to the authors. Furthermore, in the context of mobile robots, Lau and Wong [212] (P32) describe a completely distributed and adaptive control framework based on generic immunity theories for managing, coordinating, and scheduling a fleet of agents in an [AGV](#) system. The authors verified the architecture in a simulation study of an intelligent transport system controlled by a fleet of [AIS](#) agents in a warehouse environment. Later, the authors presented a mathematical model of the control framework [219] (P39).

On the other hand, Lau and Ng [215] (P35) present the development of the [MAS](#)-based control paradigm, inspired by the [CSA](#), for the distributive trajectory control of a robot manipulator. The authors attempted to provide cobot qualities such as adaptability, resilience, and robustness. They verified the method in a MATLAB simulation of the *Whole Arm Manipulator* system, a four degrees-of-freedom robot placed at the University of Hong Kong's Intelligent Systems Laboratory. Ko *et al.* [194] (P13) offer a distributed control method for a [MSR](#)²³ robot arm based on immunosuppression theory. The suggested framework, known as the *General Suppression Framework*, is adaptable, resilient, robust, and self-tolerant. It allows robot modules to produce emergent group behaviors by demonstrating aggressive or tolerant behavior in response to environmental changes. The framework is validated using computer simulation (MATLAB) and in a hardware implementation of the [MSR](#) manipulator with seven modules. Huang [211] (P31) has suggested an online motion control of four-wheeled redundant Swedish mobile robots utilizing *FS-PBFOAIS*, a fuzzy system merged with the Bacterial Foraging Optimization metaheuristic approach and [AISs](#). To self-adjust the polar-space trajectory tracking, the approach was proven in an experimental four-wheeled redundant mobile robot with one [FPGA](#)²⁴. The results demonstrate self-tolerance and robustness.

Kim [203] (P23) and Khoie *et al.* [226] (P46) suggested tuning methodologies for [PID](#) controllers in the context of the [SCADA](#) & [ICS](#) application areas. The first method employs [AIN](#) to make the system adaptable and self-tolerant to its external environment. At the same time, the system allowed a resilient parallel distributed processing network to complete patterns in response to environmental circumstances. The method is tested in a simulated power plant system. The second approach presents a Genetic-[AIS](#) algorithm for [PID](#) tuning that employs multi-objective optimization. The authors also used simulation control test scenarios to evaluate the system. Furthermore, Wang *et al.* [208] (P28) offered a distributed [AIN](#)-based method for controlling complicated industrial processes. The program may learn control knowledge from diverse operators and optimize current control knowledge in real-time. Validation takes place in a virtual environment (a beer fermentation process). Recently, Samigulina and Samigulina [187] (P06) leverage the [CSA](#) model

²³Modular Self-Reconfigurable

²⁴Field-programmable Gate Array

in conjunction with the **MAS** paradigm to build distributed and self-adaptive complex object control capabilities in Honeywell **DCS**²⁵. However, it is unclear the conditions during the solution experimentation and validation.

Finally, some control-related contributions are available in the *Electric Power Industry*. In the context of distributed generators, Jin *et al.* [210] (P30) offer an immune-inspired approach for power system damping control in **DG** systems. The authors verified the solution in a simulated example of a power system with large-scale dispersed generation integration. Bhuvaneswari *et al.* [217] (P37), on the other hand, provides the implementation of a **MAS** distributed control system in a microgrid operation to optimize the production of local **DG** systems. This control system is built on the **CSA**, and it has been tested in a simulated environment, considering genuine market pricing for electricity and bids from dispersed generators reflecting realistic operating expenses. Rimal and Belkacemi [224] (P44) provide an immune-inspired technique for modeling autonomous generation control in a two-area power system. The concept is to introduce system flexibility and self-tolerance to lessen mechanical oscillations. The authors tested the solution in a simulated environment. Guerrero *et al.* [198] (P18) recently suggested a **CSA** technique for an optimal real-time solution of the Coordinated Volt/VAr Control problem in distribution networks with different control equipment. The authors verified the approach in a simulated scenario that mimicked the interoperability activities that are routinely established among the critical functionalities of a modern **DMSs** during the real-time operation of a Coordinated Volt/VAr Control. Finally, Lizondo *et al.* [206] (P26) proposes a **MAS** distributed control system based on **AIN** to address the Peak Load problem in Smart Grid situations. A simulated scenario (Air Conditioner devices) was employed to validate the approach.

Islanding *Islanding Detection* is a popular study topic in the *Electrical Power Industry* because it attempts to detect islanding events. *Islanding* is a condition in which a distributed generator continues to power a location even though external electrical grid power is no longer present. Based on digital signal processing and harmonic signature detection, Yin [216] (P36) provides a robust immune inspired technique to identify islanding of **DG**. The method was tested in a simulated grid-connected inverter based on a digital signal processor, attaining a **TRL** of 3-4 and **LOA** of 1. This indicates that while the system may suggest goal-oriented changes, a human must approve them before they can take effect.

Anomaly Detection *Anomaly Detection* identifies a system's undesirable behavior by comparing it to a preset intended behavior, often known as the standard or baseline behavior. The premise is that anomalies, or faults, are dataset outliers that always deviate from the expected behavior of a typical dataset. We found studies on *Anomaly Detection* in **CPSs**, *Manufacturing*, and the *Electric Power Industry*. Overall, **TRL** is relatively high when compared to other study domains because various proposals have been demonstrated and validated in relevant situations. However, certain

²⁵*Distributed Control System*

concepts have only been validated in laboratory settings (both dataset and simulation). The overall **LOA**, on the other hand, is reasonably low. One can find work in the **CPS** field with no system autonomy achieved (**LOA** = 0). Also, the manufacturing-related proposal achieves a **LOA** of 1, while in the *Electric Power Industry* it is found **LOA** = 2 (it presents partial system autonomy).

In terms of **CPS** applications, Lokesh and Kumaraswamy [184] (P03) proposed a **DCA**²⁶ strategy for anomaly detection towards **CPS** state awareness. The essential contribution was to describe system robustness using the a **MAS** paradigm on the a **CPS** (together with the **DCA**). Diez-Olivan *et al.* [225] (P45) proposed an adaptive learning strategy based on an ensemble method, i.e., **DCA** for drift detection and a **DNN**²⁷ that dynamically adjusts to new operating circumstances. The approach is used for industrial prognosis, i.e., predicting failures of an industrial asset, and is validated using operational data from an experimental benchmark encompassing a real-world industrial scenario. Finally, Pinto *et al.* [3] (P47) propose an ensemble approach, the **CDCA**²⁸, which merges the **DCA** with a clustering algorithm for flexible monitoring and anomaly detection in industrial processes. The **CDCA** tackles some **DCA** original problems, such as manual characterization of signals (using a K-Means approach), and online data analysis and classification. The approach was validated in two industrial-oriented datasets regarding physical and network-related data.

On the other hand, Guo and Yang [199] (P19) proposed an anti-jamming ensemble technique that integrates **DT** with **AIN** for distributed machine energy leakage diagnostics in complicated contexts such as parallel-machine job shops for a manufacturing application. Several tests were carried out on China's tire vulcanization factory floor to detect steam leaking from steam traps to evaluate the suggested technique. The findings indicate that the lightweight immune strategy promotes robustness and self-tolerance. Moreover, Kim *et al.* [227] (P49) propose an ensemble approach, merging the **NSA** with the **CSA**, to enable a multiclass anomaly detection algorithm. The proposed algorithm was validated using an intelligent maintenance bearing dataset and a vacuum deposition equipment dataset.

In the *Electric Power Industry*, Leao *et al.* [193] (P12) provide an ensemble technique for estimating fault sections in *Electric Power Systems* based on the unconstrained binary programming model and the **CSA**. The suggested approach is tested in a simulated setting employing a portion of the South Brazilian electric power system. Lima *et al.* [190] (P09) offer a technique for voltage and current anomaly detection in electrical distribution systems based on **NSA**, which is especially useful in smart grid situations. The authors used two simulated distribution systems to validate the solution.

Intrusion Detection System An **IDS** is a software program that monitors harmful behavior to identify intrusions by gathering and analyzing data. A generic **IDS** gathers and interprets system data before responding to the outcomes of the analysis. In this scenario, the analysis is carried out

²⁶Dendritic Cell Algorithm

²⁷Deep Neural Network

²⁸Cursory Dendritic Cell Algorithm

utilizing AIS methodologies. Furthermore, the detection analysis is anomaly-based (A-IDS²⁹). In this sense, intrusion detection might be considered a sub-category of anomaly detection. The abnormalities are caused by malicious activity, such as cyber and physical attacks. IDSs are utilized in *Manufacturing*, *CPSs*, *Robotics*, and in the *Electric Power Industry*. Overall, the bulk of the concepts discovered was verified using datasets for proof of concept or small lab testbeds, thus reaching a TRL of 3-4 and 4-5, respectively. However, the LOA of the majority of proposals (dataset verified) is 0 because no system autonomy was established. Considering the work tested on lab testbeds (in the robotics field), a LOA of 2 is feasible. This indicates that the system can self-adjust its behavior (within certain limitations), but the human retains complete decision-making authority.

Pinto *et al.* [5] (P01) offer an IDS strategy in manufacturing scenarios, based on the deterministic version of the DCA (dDCA³⁰). A testing scenario was created by implementing and injecting several attacks on an OPC UA³¹ based CPS testbed. This was used to assess the efficiency of the dDCA. The immune model introduces adaptive, robustness, and self-tolerance features. Later, Pinto *et al.* [4] (P16) introduce the iDCA³² for online incremental detection. The iDCA was also validated using datasets. Finally, Clotet *et al.* [197] (P17) offer a MAS real-time IDS based on the NSA to identify physical component assaults at the industrial process level. A water treatment facility laboratory dataset was used to validate the solution.

Regarding CPSs applications, Pinto *et al.* [2] propose a MBE³³ approach to enable security-by-design during the development of CPSs, using the IEC³⁴ 61499 standard. The security features introduced rely on an IDS based on the iDCA. The authors used a CPSs testbed to validate the proposed approach. Results show that the solution is very lightweight and can dramatically reduce design and code complexity while improving application maintainability.

Degeler *et al.* [196] (P15) offer a DT-based IDS to safeguard a fleet of AGV from cyber-attacks that might cause serious physical damage to the production system in the robotics area. The system is evaluated on a simulated manufacturing floor using a distributed scenario (three AGVs working at the same time). Finally, in the Electric Power Industry, Igbe *et al.* [186] (P05) propose an IDS in industrial communication situations based on dDCA. The authors employed the DNP3³⁵ protocol in this example. Also, they presented a dataset for a Smart Grid scenario to test the approach.

Fault Diagnosis *Fault Diagnosis*, also known as fault isolation, is the process of monitoring a system and identifying flaws as they occur, pinpointing the type of fault and its location, and identifying one or more underlying causes of issues, to the point that corrective action may be performed. Typically, *Fault Diagnosis* comprises the discovery of anomalies. We found *Fault Diagnosis* in *Electric Power Industry* applications such as *SCADA* & *ICSs*, *Manufacturing* and *CPSs*.

²⁹ Anomaly Intrusion Detection System

³⁰ deterministic Dendritic Cell Algorithm

³¹ OPC Unified Architecture

³² Incremental Dendritic Cell Algorithm

³³ Model-based Engineering

³⁴ International Electrotechnical Commission

³⁵ Distributed Network Protocol 3

Overall, the **LOA** is 2, indicating that the suggestions bring partial autonomy to the system while humans retain complete decision-making authority. On the other hand, we found work validated in lab situations (mainly simulations in **SCADA** & **ICS**s and *Electric Power Industry* applications for proof of concept), obtaining a **TRL** of 3-4, and validation in partially applicable settings (**SCADA** & **ICS**s and **CPS**s), attaining a **TRL** of 4-5.

Kayama *et al.* [207] (P27) suggest an ensemble (combining **AIN** with **LVQ**³⁶) for distributed detection and diagnosis of defective sensor outputs in control plants for **SCADA** & **ICS** applications. The solution is validated in a simulated environment (temperature sensors for a heating furnace plant of a hot strip mill line). Zhao *et al.* [202] (P22), on the other hand, provide an online fault diagnostic system for safety in chemical processes by merging **ANN**³⁷ with an immune inspired model. The authors created the solution for a lab-scale distillation process, and it possesses adaptable, resilient, and self-tolerance qualities. Furthermore, on **CPS** application situations, Zhu *et al.* [201] (P21) offer an **AISFD**³⁸, paired with Bayesian estimate, for regulating alarm floods during chemical process transitions. A pilot-scale distillation process was deployed to verify the solution.

On Manufacturing applications, Outa *et al.* [228] propose the **NSA** for failure detection during the maintenance of pressure vessels. Detection relies on the recognition of vibration signals. The approach was validated in a small lab testbed that represents the behaviour of the flow of a fluid and the respective vibration of the system. Experimental results show that the approach is robust in precision and signal recognition and for classifying the degree of severity and probability of failure. Finally, Lima *et al.* [192] (P11) offer a new technique to detecting and classifying voltage disturbances in electrical distribution systems based on a combination of wavelet transform and **NSA**. The authors simulated two distribution systems to assess the performance of the suggested technique.

Operations Research *Operations Research* is concerned with developing and implementing sophisticated analytical methods to enhance decision-making in complicated problems, where optimum or near-optimal solutions are generally discovered. *Operations Research* may be found in various applications in this **SLR** context, primarily for optimization and scheduling problems. In these solutions, a **AIS** model is employed as a meta-heuristic technique, such as in the *Electric Power Industry*, *Robotics*, *Manufacturing*, and **CPS**. We found **TRL** accomplishment to be good (**TRL** = 4-5 or 5-6) since the majority of the work is validated and shown in real relevant contexts. However, in the *Electric Power Industry* and *Robotics* fields, specific proposals have been validated in simulation settings for proof of concept (**TRL** = 3-4). On the other hand, there are both very low and very high **LOA**. In the *Electric Power Industry*, for example, the greatest **LOA** discovered was two since technologies only provide limited autonomy. We found **LOA** of 3 and 4

³⁶Learning Vector Quantization

³⁷Artificial Neural Network

³⁸Artificial Immune System-based Fault Diagnostic

in both *Manufacturing* and *Robotics* areas since the system already has various autonomic capabilities, including much of the decision-making on its own.

There are various sorts of optimization challenges in the *Electric Power Industry*. Aghaebrahimi *et al.* [205] (P25) developed an optimization strategy for tackling the **DG** placement problem. The technique is based on the **CSA** and has been validated in Tehran's Khoda-Bande-Loo distribution test feeder. Xiaobo and Guoqing [213] (P33) concentrated on multi-objective distribution network design that takes into consideration lowering the investment cost of **DG** and distribution network power loss. The technique is based on the *IPSO* algorithm, which combines Particle Swarm Optimization with immunological principles. Validation takes place in a distribution network. Hanumantha Rao and Sivanagaraju [221] (P41), on the other hand, investigated the problem of multiple **DG** unit placement. To estimate the ideal **DG**-unit size and position, a **CSA**-based model is developed. In a hypothetical case, the solution is validated. More recently, Yuan *et al.* [218] (P38) has suggested a multi-objective optimization immune-inspired method for coordinating the settings of a distributed static synchronous series compensator controller in power grids. The solution is validated in a simulated scenario.

Li and Cai [191] (10) provide an ensemble solution (**ANN** combined with **DT**), designated **I²S**³⁹, utilized to achieve optimum sustainability in industrial systems. This sustainability optimization refers to optimizing production lifecycles for reduced global energy consumption through adaptive job re-scheduling. The **I²S** system has been proven through real-world industrial implementations in Sweden, the United Kingdom, and Spain. More recently, in the **CPS** field, Semwal and Nair [200] (P20) provided a decentralized technique, based on the combination **CSA**, **AIN** and **DT**, applicable to numerous distributed problems in **CPS**s. Despite being designed for **CPS**s, the approach has been tested in both emulated (**CPS**) and real-world (robot) contexts. In this scenario, a true mobile multi-robot system in which robots must learn the optimal path to follow.

Finally, Yunyuan and Zhizeng [222] (P42) suggested an **AIN** model for effective cooperative task allocation in a multi-robot system with autonomously distributed architecture. The dynamic allocation approach for collaborative robots was exhibited and examined in the emergency response simulation. Xu *et al.* [185] (P04), on the other hand, suggested an **AIS**-fuzzy approach (fuzzy systems paired with an **CSA** model) for a hybrid optimization methodology to achieve distributed formation control. For solution validation, the System-on-a-programmable-chip technique is used, with each robot implemented as an independent **CPS** in one **FPGA** chip.

4.2.2.3 AIS Conceptualization

This section attempts to answer the **RQ T.3** *What are the main motivations for using Artificial Immune Systems in Smart Manufacturing Systems?* by gathering all the Self-* and specific **AIS** features introduced by the implemented immune-models from the selected papers. This data is analyzed and correlated in light of the primary **SMS** needs (more information can be found in Section 1.1). This way, it is feasible to determine whether the intended **SMS** requirements are

³⁹Intelligent Immune System

satisfied and what attributes have been included to make this possible. Table 4.5 summarizes the AIS conceptualisation. The nomenclature used in the table is the following: **SMSs Requirements**: *STD*=Standardization, *OM*=Optimized Management, *NET*=Communication Network Infrastructure, *SS*=Safety & Security, *HM*=Human Management, *UP*=Upskilling, *RF*=Regulatory Framework, *RE*=Resource Efficiency, *AU*=Autonomy, *PEQ*=Process Efficiency & Product Quality, *DA*=Data Analytics, *AD*=Adaptability; **Self-* Properties**: *SAD*=Self-adaptive, *SH*=Self-healing, *SAW*=Self-awareness, *SI*=Self-immunity, *SOR*=Self-organizing, *SC*=Self-configuration, *SOP*=Self-optimizing; **AIS Properties**: *AD*=Adaptive, *RO*=Robust, *RES*=Resiliency, *ST*=Self-tolerance, *LH*=Lightweight, *DS*=Distributed.

In a prior review study, Oztemel and Gursev [25] provide 12 major goals to achieve I4.0 (see more detail in Section 1.1). The most significant goals addressed in this SLR are: *Data Analytics* (25.5%), *Adaptability* (19.4%), *Autonomy* (18.9%), *Optimized Management* (14.3%) and *Process Efficiency & Product Quality* (10.2%). *Data Analytics* refers to the capacity to handle large amounts of data and execute well-defined analysis to manage the system and accomplish specific goals. This incorporates a significant sensorization of the system for continuous data collecting. *Autonomy* and *Adaptability*, on the other hand, relate to the generation of self-behaving systems (with little human input) and system response to changes. These objectives are linked because *Adaptability* can only be achieved by Big Data analysis. Furthermore, system *Adaptability* is one of the most significant features when trying to reach a Self-behaving system. This is the desired approach to manage very complex systems (*Optimized Management*). Finally, *Process Efficiency & Product Quality* refers to the dynamic interaction that may occur between process devices, in a M2M⁴⁰ communication situation (for process optimization), or between equipment and product (for product quality control). This can only be accomplished with a substantial digitalization component, mostly through the use of MAS architectures.

A Self-behaving system is one that implements one or more Self-* qualities to accomplish a certain LOA. The top Self-* features discovered in the chosen studies are *Self-awareness* (35.6%) and *Self-adaptive* (28.1%). Other uncommon qualities include *Self-healing* (9.6%), *Self-configuration* (8.9%), *Self-optimization* (6.7%), *Self-immunity* (5.9%) and *Self-organizing* (5.2%). All of these Self-* attributes attempt to give the system some autonomy, which is in line with the *Autonomy SMS* goal.

Self-awareness refers to a system's knowledge of its state and surrounding environmental factors to modify and adapt to changes and external activities. Because continuous system monitoring is feasible (sensor data is continuously available), the *Self-awareness* attribute is added and inspired by the *Data Analytics SMS* objective. *Self-adaptive*, on the other hand, refers to a property in which the system adapts to changes in the environment by modifying its internal attributes correspondingly. This has a significant correlation with the *Adaptability SMS* goal. Finally, in general, the other Self-* qualities relate to systems that may modify their configurations and system architecture to restore or enhance their global state (in response to a set of external activities).

⁴⁰Machine-to-Machine

Table 4.5: Artificial Immune System conceptualization.

ID	SMS Requirements											Self-* Properties							AIS Properties								
	STD	OM	NET	SS	HM	UP	RF	RE	AU	PE	QD	AA	DA	AD	SAD	SH	SAW	SI	SOR	SC	SOP	AD	RO	RES	ST	LH	DS
P01	X	X	✓	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	X	X	X	✓	✓	X	✓	X	X
P02	X	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	X	X	✓	X	✓	✓	✓	✓
P03	X	✓	X	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	X	X	X	X	✓	X	✓	✓	X	X
P04	X	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	X	X	✓	X	X	✓	✓	✓	X	✓	✓
P05	✓	X	✓	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	X	X	X	✓	✓	X	✓	X	X
P06	X	✓	X	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	X	✓	X	X	✓	X	✓	X	✓	X	X	✓
P07	X	✓	X	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	X	X	X	X	✓	X	✓	✓	X	X
P08	X	✓	X	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	X	X	X	X	✓	X	✓	✓	X	X
P09	X	X	X	X	X	X	X	X	✓	X	✓	X	✓	X	X	✓	✓	X	X	X	X	X	✓	X	✓	X	X
P10	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	X	✓	X	X	X	✓	✓	✓	X	✓	X	X
P11	X	X	X	✓	X	X	X	X	✓	X	✓	X	✓	X	X	X	✓	✓	X	X	X	✓	X	✓	✓	X	X
P12	X	X	X	X	X	X	X	X	✓	X	✓	X	✓	X	X	✓	✓	X	X	X	✓	X	✓	✓	✓	X	X
P13	X	✓	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	X	✓	X	X	✓	✓	✓	✓	✓	✓
P14	X	✓	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	X	X	X	X	✓	X	X	X	✓	X
P15	X	✓	X	✓	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	✓	X	X	X	✓	X	✓	✓	X	✓
P16	X	X	✓	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	X	X	X	✓	✓	X	✓	X	X
P17	X	✓	X	✓	X	X	X	X	X	✓	✓	X	✓	X	X	✓	✓	X	X	X	X	X	X	✓	✓	X	X
P18	X	✓	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	✓	X	X	✓	X	✓	✓	✓	X	X
P19	X	✓	X	X	X	X	✓	✓	X	✓	X	✓	X	✓	X	X	✓	✓	X	X	X	X	X	✓	✓	✓	✓
P20	X	✓	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	X	X	X	✓	✓	✓	✓	✓	X	✓	✓
P21	X	X	X	✓	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	✓	X	X	X	✓	✓	✓	✓	✓	X
P22	X	X	X	✓	X	X	X	X	✓	X	✓	✓	✓	✓	✓	✓	✓	✓	X	X	X	✓	✓	✓	✓	✓	X
P23	X	X	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	✓	✓	X	X	X	✓	✓	✓	✓	✓	X	X
P24	X	X	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	X	X	X	X	✓	X	✓	✓	X	X
P25	X	✓	X	X	X	X	X	X	✓	X	✓	X	✓	X	X	✓	✓	X	X	X	✓	X	✓	✓	✓	X	✓
P26	X	✓	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	X	X	✓	✓	X	✓	✓	✓
P27	X	✓	X	✓	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	✓	X	X	X	✓	X	X	✓	X	✓
P28	X	✓	X	X	✓	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	X	✓	X	✓	✓	✓	X	✓
P29	X	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	✓	X	✓	X	✓	✓	X	✓
P30	X	✓	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	✓	X	X	✓	X	✓	✓	✓	X	✓
P31	X	X	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	✓	X	X	✓	X	✓	✓	✓	✓	X
P32	X	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	✓	X	✓	X	✓	✓	✓	✓
P33	X	X	X	X	X	X	✓	✓	X	✓	X	✓	X	✓	X	X	✓	X	X	✓	X	X	✓	X	✓	X	X
P34	X	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	✓	X	✓	X	✓	✓	✓	✓
P35	X	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	X	X	✓	✓	✓	✓	✓	X
P36	X	X	X	✓	X	X	X	X	✓	X	✓	X	✓	X	✓	✓	✓	X	X	X	X	X	✓	X	✓	X	X
P37	X	✓	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	X	✓	✓	✓	X	✓	✓	✓
P38	X	X	X	X	X	X	X	X	✓	✓	✓	X	✓	X	✓	X	✓	X	X	✓	✓	X	✓	X	✓	X	X
P39	X	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	X	X	✓	X	✓	✓	X	✓
P40	X	X	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	✓	X	X	✓	X	X	✓	✓	X	X
P41	X	✓	X	X	X	X	X	X	✓	X	✓	X	✓	X	X	✓	✓	X	X	X	✓	X	✓	✓	✓	X	X
P42	X	✓	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	✓	X	X	✓	X	✓	✓	✓	✓	✓
P43	X	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	X	✓	X	✓	X	✓	✓	✓	✓
P44	X	✓	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	✓	X	X	✓	X	✓	X	✓	X	X
P45	X	X	X	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	X	✓	✓	X	X	X	✓	✓	✓	✓	X	X
P46	X	X	X	X	X	X	X	X	✓	X	✓	✓	✓	✓	✓	X	✓	✓	X	X	X	✓	✓	✓	✓	X	X
P47	X	X	X	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	X	X	X	✓	✓	✓	✓	X	X
P48	X	X	✓	✓	X	X	X	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	X	X	X	✓	✓	✓	✓	✓	X
P49	X	X	X	X	X	X	X	X	X	X	✓	X	✓	X	✓	X	✓	✓	X	X	X	✓	✓	✓	✓	X	X
P50	X	X	X	X	X	X	X	X	X	X	✓	X	✓	X	✓	X	✓	✓	X	X	X	X	✓	✓	✓	✓	X

These behaviors are consistent with both *Optimized Management* and *Process Efficiency & Product Quality* SMS goals because the management of complex systems, especially MAS-related, require a high level of adaptation (configuration and settings update) to achieve an optimum global state. Because of their commonalities, AIS techniques based on MAS architectures have a common link.

Looking closer at the main AIS features identified in all the immune models implemented, the top three are *Self-tolerance* (27.7%), *Adaptive* (23.5%) and *Robust* (18.7%). *Self-tolerance* in the BIS⁴¹ refers to the elimination of auto-reactive immune cells, which ensures that the immune system does not attack its body cells. In engineering contexts, this quality indicates that the system clearly distinguishes between normal system behavior/states/conditions and abnormal system behavior/states/conditions while reacting solely to the anomalies. This attribute is critical for meeting the SMS aim of *Autonomy*, as human monitoring and decision-making are kept to a minimum. The *Adaptive* AIS attribute, on the other hand, alludes to continual learning to generate adaptable behaviors, which aligns with the *Adaptability* SMS goal. Finally, the *AIS Robust* attribute relates to the capacity of the system to operate in the presence of data scarcity or low quality. Because the data gathering procedure might be inaccurate in some cases, this characteristic is critical for meeting the SMS goal *Data Analytics*.

Based on the research provided, it is reasonable to conclude that the primary objective for utilizing immune models is to provide flexible decision-making processes for dynamic and complicated settings such as SMSs. Given the fundamental AIS qualities mentioned and the Self-* attributes obtained, it is evident that there is a direct cause-effect relationship between immunological models and system autonomy. The majority of this autonomy can be found in state awareness and adaptation scenarios, which are strongly connected with the SMS goals of *Autonomy* and *Adaptability*. Furthermore, because immunological models provide *Self-tolerance* and *resilience*, they are an excellent fit for SMS applications. On the one hand, the desired capacity of no human involvement and monitoring is only attainable if the system decision-making process is dependable, tolerating favorable conditions and reacting to negative ones.

4.2.2.4 Research Gaps

By analyzing the correlated data between Self-* and AIS attributes, and SMS goals, this section attempts to answer the **RQ T.4** *What are the research gaps and future areas within Smart Manufacturing Systems based on Artificial Immune System solutions?*. Using Table 4.5 and the analysis presented in the AIS conceptualization, it is feasible to identify research gaps in terms of less addressed SMS objectives. The SMS requirements with less research work, considering the application of AIS approaches, are *Safety & Security* (6.1%), *Resource Efficiency* (2.6%), *Communication Network Infrastructure* (2%), *Standardization* (1%), *Human Management* (0.5%), *Regulatory Framework* and *Upskilling* (0%).

⁴¹ *Biological Immune System*

Safety & Security *Safety* refers to the lack of threats to human operators, i.e., equipment, manufacturing facilities, or products that should not endanger humans or the environment. On the other hand, *Security* relates to the protection of digital systems, i.e., the system itself must be safeguarded against abuse and cyber-attacks.

The majority of methods tackling the security goal are in the *IDS* research field, i.e., the challenge of safeguarding a system by detecting assaults against it. Pinto *et al.* [5, 4] (P01, P16, P48) and Igbe *et al.* [186] (P05) proposed *NIDS*, which identify network threats by analyzing network data. Clotet *et al.* [197] (P17), Degeler *et al.* [196] (P15) and Pinto *et al.* [3] (P47), on the other hand, propose a *HIDS*⁴², i.e., detection focusing on assaults targeting physical components at the industrial process level by analyzing host data. Regarding the safe topic of this goal, Yin [216] (P36) provides a robust immune-inspired strategy to identify islanding scenarios, which are a hazard to people and the environment.

The trend toward employing immune-related techniques for *Safety & Security* focuses on detection rather than traditional ways for preserving the integrity of information, equipment, and assets (data encryption, digital signatures, password authentication, public keys encryption, among others.). An autonomic and adaptive attack/intrusion detection is especially crucial given the complexity of *SMS* designs and the worldwide connection of shop-floor systems, which exposes these types of systems to far more vulnerabilities than previous systems. Furthermore, given the advantages of *BIS* operations and the difficulty of detecting attacks/intrusion in computer systems, *AIS* has enormous potential to be exploited for autonomic *IDS*s. However, this study is still in its early stages. Previous work can be found for more information on bio-inspired *IDS*s and *AIS*-related security in *SMS* enabling technologies such as *IoT* and *CPS*s [170, 175, 229, 138].

Resource Efficiency Resource utilization efficiency refers to using new materials, processes, technologies, and other measures to improve resource utilization efficiency while reducing and balancing environmental pollution. Some work on *AIS* application for resource utilization efficiency in *SMS*s can be found, particularly in *Manufacturing* and *Electric Power Industry* application settings.

Li and Cai [191] (10) present the *I²S* solution in the *Manufacturing* application field to achieve optimum sustainability, i.e., optimization of manufacturing lifecycles for global energy consumption reduction. Guo and Yang [199] (P19), on the other hand, provide a detection technique for machine energy leakage in complicated contexts. Given the resource utilization goal, both strategies aim to reduce global energy consumption (job re-scheduling or energy leakage detection).

Bhuvaneswari and Srivastava [217] (P37), and Lizondo *et al.* [206] (P26) present control solutions for the *Electric Power Industry* application area. The first study, microgrid operation control, attempts to optimize the power generation of local *DG* systems. The second paper suggests a control method to address the Peak Load problem. Both proposals centered on energy demand management, which governs the size of generators, transmission lines, transformers, and circuit breakers at various times. Xiaobo and Guoqing [213] (P33), on the other hand, were concerned

⁴²*Host-based Intrusion Detection Systems*

with the development of distribution networks while taking into consideration the cost minimization of DG and power losses.

Despite the prior study, further research is required since the industrial sector remains one of the world's leading sources of greenhouse gas emissions. Developing lean/green techniques can enable manufacturers to reduce expenses, particularly energy costs, improve process efficiency, and even offer a safer working environment for staff. Furthermore, it is conceivable to provide resource adaptability in energy use by incorporating immunological techniques. More detail about lean-green manufacturing can be found in previous work [230, 231, 232].

Communication Network Infrastructure *Communication Network Infrastructure* refers to the infrastructure required by communication networks to enable increased volume or quality data exchange. Overall, this goal's issues are connected to: i) scalability, ii) security, and iii) availability. According to IoT principles, all industrial assets in the SMS environment are connected through the Internet. As a result, wireless communication technologies play an essential role since they enable ubiquitous internet access and enhanced contact (availability). Furthermore, as the number of networked linked devices grows, the network architecture must be scalable. Finally, the increased number of connections increases the possibility of malicious cyber-attacks deployed in the system (security).

Three articles in this SLR deal with network-related concerns, including NIDSs [5, 4, 186, 2] (P01, P16, P05 and P48), all of which focus only on network infrastructure security. This amount of study activity is insufficient to make an influence on this SMS goal. Other techniques may also be required for the other two challenges (scalability and availability).

Standardization *Standardization* and open standards for a reference architecture refer to the normalization of cooperation mechanisms and the information that is to be exchanged between system devices. Reference architectures are essential for the development, integration, and operation of SMS technical advancements. In manufacturing system's networked devices, such as Edge devices, a joint standardization effort exists, which is related to the *Communication Network Infrastructure* SMS goal.

In this SLR, Igbe *et al.* [186] (P05) and Pinto *et al.* [2] (P48) address the *Standardization* goal. On the one hand, Igbe *et al.* [186] (P05) focuses on the HIDS on networks that use the DNP3 communication protocol, which is part of the IEEE Standard for Electric Power System Communications. On the other hand, Pinto *et al.* [2] (P48) uses the IEC 61499 standard to provide security-by-design features during the development of a CPS.

Moreover, Pinto *et al.* [5, 4] (P01 and P16) provide detection solutions in field networks using OPC UA. Despite continued specification work, substantial user support, and implementations, OPC UA may be characterized at this moment as a standardization attempt rather than an established standard, despite significant marketing efforts to boost its adoption.

Human Management *Human Management* refers to the structure and design of work that considers the role of employees. SMS transformation should not be limited to technological innovation and deployment. One should also consider how workers will be involved in the SMS transformation, as well as how the presence of technology in the workplace will affect employees' job processes (especially in light of novel situations such as HMI⁴³). *Human-Centered Manufacturing* [233] or *Operator 4.0* [234] are other names for this paradigm.

Bio-inspired human-centered manufacturing has the potential to be a hot topic considering the challenges regarding complex system design considering the human operator or even HMI scenarios for collaborative tasks. In this case, robots need to learn and adapt their task execution considering the operator's work profile and task performance within the overall collaboration. This implies real-time adaptation, self-awareness, and self-organization capabilities. The regulatory framework refers to the requirement for new technology to be introduced in SMSs to be regulated. This legislation considers the legality of new technology or the attendant liability and data protection problems that might prevent its acceptance and hinder the innovation process.

Upskilling The manufacturing workforce and management, such as professionals and executives, must obtain high-demand skills in new transformative technologies inside the SMS paradigm. These systems are altering the employment requirements for various roles within industrial firms. As a result, multiple production processes and jobs are being modified to meet the needs of this new paradigm. This means that the current workforce and management will confront new hurdles in terms of skill development. As a result, the deployment of SMSs introduces new problems for new skills and academic training for continued professional growth and upskilling, also known as *Education 4.0* [235].

Regulatory Framework The regulatory framework refers to the requirement for new technology to be introduced in SMSs to be regulated. This legislation considers the legality of new technology or the attendant liability and data protection problems that might inhibit its acceptance and hinder the innovation process.

4.2.3 Conceptual Framework for Industrial AIS Applications

Despite the extensive research efforts and various implementations of current AIS models, its actual prevalence in SMSs remains relatively low. Because the average TRL is between 3 and 4, the results of this study show that the majority of research is executed and verified as a proof of concept, utilizing datasets, simulation settings, or, at most, proven in laboratory contexts. Furthermore, the influence of such technologies on system autonomy is underwhelming since the typical LOA is between 2 and 3 (partially/limited autonomy).

On the one hand, some academics have claimed in recent years that AISs has not had the success of other bio-inspired approaches [236]. This explains why there are more ensemble models

⁴³*Human-Machine Interaction*

than basic models. Furthermore, the high stochastic nature of immunological systems makes them difficult to explain and assess, posing a challenge to the study area because these models are difficult to employ outside of the field. This is not the case with many traditional ML approaches, especially the ones related to XAI⁴⁴ [237]. Implementing AISs in SMSs situations, on the other hand, may necessitate a significant digital transformation in the manufacturing business, which will demand time and money to install various adjustments to current systems. According to the SMS vision, these changes may be connected to the technological implementation of CPSs, Edge device manipulation, embedded systems (FPGA), and Big Data analysis for data-driven scenarios. This may not be possible in some systems, such as *Electrical Power Systems*. That is why simulation and emulation are essential technologies for validating implementations.

Because there is no significant evidence of effective AIS applications in SMSs, industrial adoption of the technology is delayed. However, the influence that such bio-inspired technologies may have on the SMS paradigm has great potential. Industrial businesses must grasp not only the enabling technologies of AIS implementation and deployment but also the operational and organizational needs to accomplish the complete autonomy goal in SMSs. Thus, Figure B.1 in Appendix B illustrates a conceptual framework of industrial AIS applications, offering manufacturers with some guidance for using AISs in their manufacturing applications to tackle various challenges. The authors propose a conceptual framework by compiling the data from this SLR study in terms of application domains, research categories, and immune models that drive AISs in SMSs.

Moreover, the leading enabling technologies found in the literature review behind the industrial application of AISs are described next:

- **MAS:** A *Multi-Agent System* is by definition a system that exhibits autonomic behaviors by expressing one or more Self-* attributes. These systems comprise numerous interacting intelligent software agents that produce complicated behaviors even though each agent has a basic individual strategy (like a swarm). The autonomy level (Self-awareness), knowledge of just local viewpoints, and distributed control are crucial aspects of such entities. Because these qualities are pretty similar to the attributes of immunological models, the use of both technologies combined in application development is pretty widespread.
- **Edge Devices:** The Edge computing paradigm takes computation and data processing to the level of the edge device, i.e., devices that gather sensor data. It is feasible to perform real-time processing of real-time data supplied by sensors in this way. Data may transit between multiple dispersed nodes linked through the Internet in Edge computing situations, such as an IoT scenario. These nodes might potentially be resource-constrained devices, similar to a typical embedded system. Edge Devices are found in the lab or even relevant environment validation settings when considering AIS applications. They complement each other well since immunological models are lightweight, durable, and widely disseminated.

⁴⁴Explainable Artificial Intelligence

This makes them excellent for use in decentralized resource-constrained devices where real-time raw data must be analyzed with minimum latency.

- **Simulation/Emulation:** Simulation models are representations of real-world physical systems that depict the essential properties or behaviors of the chosen system or process. In the context of AISs, simulation was commonly used to verify immunity applications in SMSs, particularly in *Electric Power Systems*. This is explained considering that the system could not be engaged since it was inaccessible, unsafe, or inappropriate. These models replicate data sources with specific behaviors, whereas data is the primary input for immunological model analysis. Furthermore, some immunity approaches may inject their outputs back into the simulation model to see how the final actual repercussions of different situations and courses of action play out. Finally, simulations may be utilized to mimic the immunological model (as a natural system) and get insight into how it works.
- **Big Data:** The term Big data refers to methods for collecting and analyzing vast and complicated datasets, especially considering dispersed sensor data. As previously stated, real-time processing solutions are necessary if this gathering and analysis are undertaken at the Edge level. Typical solutions to sized data problems may feature a multi-layered and distributed design. These characteristics are shared by several immune models, making them suited for application in large data settings.

Finally, and since several proposals are validated using datasets, the main datasets found in the literature review for application validation are:

- *M2M using OPC UA* [34, 5, 4]: This dataset was generated by implementing and injecting various attacks on an OPC UA based CPPS testbed, and it aims to enable users to evaluate the effectiveness of different techniques for developed IDSs in the manufacturing context. The dDCA and the iDCA models were validated using this dataset. More detail about this dataset in Section 6.2.1.
- *Random number dataset* [184, 188, 189]: This dataset contains status monitoring measurements of different CPS layers within specific periods for anomaly detection purposes. The dDCA model was validated using this dataset.
- *DNP3 smart grid dataset* [186]: Dataset generated after the implementation of several security threats to the DNP3 communication protocol for Smart Grids. The dDCA model was validated using this dataset.
- *KDD Cup99* [88, 4]: This dataset is used for IDS purposes and contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. The dDCA and the iDCA models were validated using this dataset.
- *SWaT⁴⁵ testbed* [238, 197]: SWaT represents a scaled-down version of a real-world industrial water treatment plant. The dataset contains network and physical (process) data on

⁴⁵Secure Water Treatment

the different number of sensors and actuators in all stages of the testbed for anomaly- IDS purposes. The NSA model was validated using this dataset.

- *SKAB*⁴⁶ Dataset [239, 3]: The development of SKAB is based on sensor readings deployed in a water circulation system from an IoT testbed, located in the Skolkovo Institute of Science and Technology. Anomalies are introduced by inserting different physical faults in the system, such as closing valves, increases in temperature, and rotor imbalances. The CDCA model was validated using this dataset.

4.3 Conclusion

The research presented in this chapter is based on a comprehensive evaluation of journal and conference articles in the fields of AISs and SMSs. We recognized the necessity for this study by describing prior surveys and reviews in terms of their goal and issues addressed (SMSs, Self*, and AIS). We carried out the systematic review in accordance with PRISMA criteria, and the papers under consideration are indexed in the IEEE Xplore, Scopus, ScienceDirect, Springer, ACM, Inspec, Knovel, and Web of Science databases. Given the exclusion criteria, this review has a few limitations, such as the possibility that important papers were not considered in the research. For starters, restricting the inclusion of English-language publications means excluding non-English content. Second, several articles were excluded because we could not obtain critical metadata or the complete text. Third, and perhaps most importantly, some publication exclusions were left out because the work was not directly industry-related. However, most of the time, these studies were focused on SMS enabling technologies and other research topics that were utilized in industry situations (to some extent). Fourth, several articles were excluded merely because the work did not show outcomes, despite including solution proposals. Finally, the generic string searches for each database search may have restricted the overall search results.

The research results achieved yield a thorough characterization of AIS applications in SMSs. We begin by examining the research trend concerning these topics, followed by an appraisal of the current state of such work. With this study, it was feasible to outline the primary motives for employing AISs in SMSs, as well as highlight research gaps and remark on potential future research trends in these issues. Overall, the industrial adoption of the AIS technology is very slow. However, there is an enormous potential of the impact that such bio-inspired technologies may have on the SMS paradigm. Several SMSs are covered by functionalities enabled by AISs, especially if Industrial businesses want to accomplish full autonomy in their systems. Moreover, results suggest that the average TRL is between 3 and 4. Most of the performed research is implemented and validated as a proof of concept, using datasets, simulation environments, or, at most, demonstrated in laboratory environments. Also, the impact of such technologies in terms of system autonomy is disappointing since the average LOA is between 2 and 3 (partially/limited autonomy). Possible reasons may be:

⁴⁶Skoltech Anomaly Benchmark

- Stochastic nature: The strong stochastic aspect of immune mechanisms makes them hard to explain, evaluate, and use in different contexts. This poses a difficulty to the research field since it is hard to use these models outside the area.
- Cost & effort of industrial transformation: Implementing AISs in SMS scenarios may require an extensive digital transformation in the manufacturing industry. This involves time and money to deploy several changes to existing systems.

Thus, a conceptual framework (Appendix B) is proposed based on a thorough evaluation of current AISs in SMSs literature and empirical evidence. This conceptual framework offers a holistic picture of the main parts of implementing AISs in SMSs, such as application areas, study categories, and immunological models produced. This way, we provide some guidelines for manufacturers to adapted AISs in their manufacturing applications to solve different problems.

Chapter 5

Immune-based Industrial IDS

5.1 Introduction

In this chapter, the reader will learn about the specification of the proposed approach regarding a [DCA](#)¹-based [IDS](#)² to be applied in industrial networks. For this, in Section 5.2, we start by describing both the industrial and immune contexts for the intrusion detection problem and the scope of an [IDS](#) solution within these domains. On the one hand, within the immune context, we provide more detail about the Danger Model-based [IDS](#)s. On the other hand, within the industrial context, the reader will learn about [CPPS](#)³ security by design using [IEC](#)⁴ 61499.

Based on the contextualization, we provide a comprehensive requirement analysis about [DCA](#)-based [IDS](#)s, since the [DCA](#) is the most representative Danger Model approach. The requirement analysis is presented in the form of a [SWOT](#)⁵. Using the [SWOT](#) analysis, in Section 5.4, we overview the research strategies and the related literature by focusing on the research gaps, opportunities for further contribution, and existing work considering these strategies. Thus, in this section, we aim to find answers to the following research questions:

***RQ T.5** What are the primary strengths and weaknesses when developing Dendritic Cell Algorithm-based solutions for classification problems?*

***RQ T.6** What research strategies can be adopted for addressing opportunities and threats in industrial anomaly detection?*

The main output of this chapter is the comprehensive analysis of the research gaps and strategies for further contribution in both [AIS](#)⁶ and [SMS](#)⁷ fields. This was possible by identifying the primary [DCA](#)-based [IDS](#) strengths and weaknesses, along with the opportunities and threats of

¹Dendritic Cell Algorithm

²Intrusion Detection System

³Cyber-Physical Production System

⁴International Electrotechnical Commission

⁵Strengths, Weaknesses, Opportunities and Threats

⁶Artificial Immune System

⁷Smart Manufacturing System

applying such solutions in SMS related scenarios. Based on the existing work in the literature and the research strategies identified, in Section 5.5, we propose a specification for a novel DCA for NIDS⁸ in SMSs. Finally, Section 5.6 concludes the chapter by resuming the study findings used to provide answers to the research questions mentioned before.

5.2 Context and Scope

5.2.1 Industrial Context

According to the NIST⁹, security requirements of CPPSs differ greatly from those of classical ICT¹⁰, since CPPSs encompass a network of several types of control systems used in industry, such as SCADA¹¹, DCSs¹², PLCs¹³ [240] and ICS¹⁴. Some special considerations when referring to CPPS security are [241, 242]:

- **Real-Time:** CPPSs are generally time-critical and operate in real-time, contrary to ICT, which admit some degree of delay. In CPPSs, deviations may damage physical equipment and lead to industrial accidents. This means that any CPPS security solution, specially IDSs, should be able to operate in an online fashion, i.e., process data and output results as soon as a new data point is available to be processed.
- **Availability:** CPPS processes may be continuous and hence need to be highly available. Moreover, since industrial equipment needs to operate continuously, stopping the CPPS for bug fixing, running system updates, or even executing new system configurations may be challenging. On the one hand, the security solution must protect the system so that the system functionality is not interrupted. On the other hand, when executing the security solution, specially IDSs, it should not negatively impact the underline system.
- **Risk Management:** In a typical ICT system, data confidentiality and integrity are usually the priorities. However, for an CPPS, human safety and fault tolerance are the primary concerns. This means that a security solution, specially IDSs, should detect attacks and prevent those from potentially impacting the physical devices' functionality negatively.
- **Physical Effects:** Related to the previous point, CPPSs often have interactions with physical processes, so consequences in the CPPS domain will most likely manifest in physical events.
- **Communications:** CPPSs use a variety of industrial protocols for control and communications that are typically different from ICT systems. These industrial protocols, which were initially "secure" in closed environments, become even more vulnerable to cyber attacks in

⁸Network-based Intrusion Detection System

⁹National Institute of Standards and Technology

¹⁰Information and Communication Technology

¹¹Supervisory Control and Data Acquisition

¹²Distributed Control Systems

¹³Programmable Logic Controllers

¹⁴Industrial Control Systems

open environments. This means that a security solution, specially **IDS**, supports and analyses data of such protocols.

- **Computing Resources:** **CPPSs** include in the shop-floor a variety of sensors and actuators with limited computing power, which increases the difficulty of executing sophisticated/-complex software with security purposes.
- **Availability of Training Data:** There are few publicly available datasets for industrial environments under attack to be used for assessing detection technique's performance. Many practical learning approaches demand enormous amounts of training data, which may not be available in most industrial scenarios. Furthermore, relying on labeled data may eventually result in strategies that do not adapt well to the system's normal behavior evolution in time. In this case, not only "new" normal data should be collected, but the system must also be submitted again to a training phase, stopping the **IDS** functionality of actual anomaly detection.

The continuous transformation of industrial systems with the emerging **IIoT**¹⁵ trends to emphasize, even more, the considerations mentioned before, since it is creating new vulnerabilities and attack opportunities, exposing industrial systems to new types of threats and cyber-attacks. Therefore, the topic of **IDS** application in industrial scenarios has drawn significant attention among the research community, and many **IDS** solutions for **CPPSs** and **SMSs** have been proposed over the past years. One can find several reviews of **IDSs** in industrial ecosystems [243, 242, 244, 245, 246].

So, considering this, the design and implementation of an **IDS** to be used in an industrial context should follow specific requirements [46]:

1. The **IDS** should be anomaly-based in order to continuously adjust a normal baseline system behavior and be effective against known and unknown attacks, i.e., enabling zero-day attack detection;
2. The **IDS** should perform in a distributed way by enabling the cooperation of multiple components, e.g., Edge devices;
3. The **IDS** should perform in an online manner, i.e., data processing and the detection of anomalies should be done in real-time. One approach to this is redesigning typical anomaly detection methods considering incremental learning approaches;
4. The **IDS** should be lightweight, so it will not negatively impact resource-constrained Edge devices when executing while maintaining a good detection performance;
5. The **IDS** should be compliant with industrial protocol data, i.e., should be capable of deep packet inspection of network messages using a specific industrial protocol, such as **OPC UA**¹⁶.
6. The **IDS** should avoid as much as possible a training phase, especially when training requires to stop the **IDS** functionality. Moreover, the training should avoid labeled data since collecting normal system data beforehand is very complicated, i.e., free of attacks and faults.

¹⁵Industrial Internet of Things

¹⁶OPC Unified Architecture

In this context, the proposed detection solution presented in Chapter 6 (see Section 6.3 and Section 6.4) assumes that the IDS solution scope to the problem considers an anomaly-based nature, suitable to analyze network data using industrial protocols. Moreover, the detection technique must perform online while being lightweight, i.e., suitable for running on constrained devices.

5.2.2 Immune Context

According to Han *et al.*[46], considering the different *Hard Computing* approaches to anomaly-based IDS, namely, statistical, knowledge, and ML¹⁷-based (see Figure 2.7 of Section 2.4 in Chapter 2), it is clear that no existing methodology can satisfy consistently all of the requirements described before. Despite being the simplest and fastest, rule-based approaches fail to tolerate noisy data and provide security against unforeseen threats. Statistical approaches are often designed to achieve a compromise between performance and complexity. When compared with ML-based approaches, they are better at learning but worse at classifying. Overall, ML strategies produce the best detection, tolerance, and generalization performance but with the highest computing cost. Also, they have to be adapted to online and distributed scenarios.

However, there are differences between *Hard* and *Soft Computing*. *Hard Computing* follows mathematical methods strictly, making it inefficient to solve real-world problems most of the time. This is due to excessive computation time or the fact that it requires exact input data. On the other hand, *Soft Computing* [247] emerges from the study of biological processes, and it is suitable to approach problems in imprecise scenarios, achieving approximate, robust, and low-cost solutions. Within anomaly-based (*Soft Computing*) IDS, this literature review addresses specifically AIS techniques.

Over the years, many AIS methodologies have been employed to generate IDSs. One can find several reviews of immune-based IDSs [248, 120, 249, 182]. AIS-based intrusion detection systems are commonly utilized as anomaly detection models. Tokarev *et al.* [250] claims that AIS is one of the most promising contemporary solutions to the problem of anomaly detection. These techniques are generally more generic when compared with ML, more computationally efficient, and use fewer resources. Also, they do not require a training stage. The learning process may be simpler because it uses an evolutionary approach instead of the traditional AI¹⁸ approach.

Costa Silva *et al.* [251] argues that the usage of immune-inspired methodologies for anomaly detection has been accepted in the literature due to its parallel with bodily resistance offered by the BIS¹⁹ against disease-causing *Pathogens*. According to them, the applicability of AIS in anomaly detection is mainly based on self/nonself discrimination and Danger Models, which relate to the NSA²⁰ [252] and DCA [139, 253]. NSA techniques have existed since the '90s, and extensive work had and continues to be done in this field. On the other hand, Danger Models emerged several years after, being the DCA a significant contribution in this area.

¹⁷Machine Learning

¹⁸Artificial Intelligence

¹⁹Biological Immune System

²⁰Negative Selection Algorithm

There are critical elements of the NSA, such as:

- **Detectors:** how to represent detectors (e.g., real-valued arrays, strings). The nature of the detectors is also coupled to the input data that is to be classified. Depending on the nature of detectors, the following also has to be defined:
 - **Affinity:** procedure that, given a detector and a point to classify, results in a binary decision of detection ("detected" or "not detected"). E.g., common n-contiguous bit-s/elements, euclidean distance compared to a threshold.
 - **Detector generation:** procedure that allows the generation of new detectors, e.g., random generation of detectors.
- **Stop criterion:** when to stop generating detectors during training phase. E.g., n number of detectors reached; estimated search space coverage.

These elements are the main targets of extensive research since different NSA implementations use different strategies for such elements to achieve the central tolerance analogy. This analogy has shown great potential of NSA implementations to detect host and network intrusions effectively. However, overall, NSA presents scalability issues and poor detection performance when the malicious and normal behaviors are very close. Also, on a biological mindset, the central tolerance analogy cannot explain, e.g., self-immune diseases and the immune tolerance developed towards the gut microbiota. In these cases, self *Antigens* are triggering immune responses while non-self *Antigens* are suppressing immune responses.

With these limitations in mind, Danger Models emerged as a promising approach to intrusion detection. According to Danger Theory, immune system response is only triggered when *Antigens* are correlated with the presence of *Danger Signal* (signs of an ongoing infection). On contrary of NSA that detects *Antigens* patterns by affinity matching with detector *T-cells*, the Danger Theory focus on the correlation between *Antigens* collected and the perceived environment signals (*Danger Signal* and *Safe Signal*). This approach makes it possible to avoid scalability issues since the detector generation phase is eliminated.

5.2.3 Danger Models for IDS

The most representative algorithm under the Danger Theory family is the DCA. A comprehensive description of the algorithm is given in Section 3.4 in Chapter 3. Other existing algorithms (inspired in different immune processes) are the TLR²¹ [165] and NKC²² [166] algorithms.

Aickelin *et al.* [254] suggested using the Danger Theory paradigm to build the next generation of AIS-based IDSs back in 2003 for the first time. Later, Greensmith *et al.* [163] described in detail the DCA while validating the theoretical model in the context of computer security by modeling port scanning as an intrusion (ICMP 'ping'-based port scans). Experimental results show the

²¹*Toll-Like Receptor*

²²*Natural Killer Cell*

importance of signal selection on the algorithm's behavior. Vella *et al.* [255] compiles several aspects of the Danger Theory analogy that are suitable for the intrusion detection problem:

- *Detecting novel malicious patterns:* As explained in section 2.4.1 in Chapter 2, one of the main advantages of A-IDS²³ is that new types of intrusions can be detected, instead of using signature matching. Similarly, the feature of detecting novel attacks is easily achieved with the Danger Theory, since previously unknown attacks are detected from the damage they cause in the system to be protected (*Danger Signals*), instead of using their exhibited molecular pattern. Note that, during an attack, some form of damage is expected and should be measurable using the *Danger Signals* equivalent values.
- *Safe nonself tolerance:* Following the previous point, the main disadvantage of A-IDS is the high false-positive rate, i.e., reacting and detecting as anomaly non-malicious *Pathogens*. In Danger Theory, if a *Pathogen* reaction occurs only when it is associated with *Danger Signals*, this means that it will tolerate *Pathogens* that do not cause damage to the underlying system. This way, a Danger Theory-based IDS has the potential to keep a reduced false positive rate by design, since detection will always rely on the measure of damage induced by a *Pathogen*, so safe behaving nonself will not be flagged as malicious.
- *Autonomy:* A-IDSs identify a bad behavior by comparison to a predefined intended behavior, which is considered to be the standard or baseline behavior. This means that an initial mechanism (maybe an external component) provides information about the characterization of self profiles (normal behavior). A Danger Theory-based IDS is entirely autonomous since it is prepared to learn new types of *Pathogen* patterns without relying on predefined normal baselines. The only requirement would be to correlate the monitored *Antigen* with the environmental signals (*Danger* and *Safe*). This way, a Danger Theory-based IDS works great as an A-IDS, but it does not specify any normal baseline behavior to support detection.
- *Part of a multi-layered defense system:* As mentioned before in Section 2.4 in Chapter 2, an IDS usually belongs to a multi-layered defense system. While traditional network security solutions represent the first line of defense, IDSs can be seen as a second line of defense to guarantee network security. A Danger Theory-based IDS, inspired by the BIS, is by nature a multi-layered defense system. According to the analogy, the biological first line of defense is represented by physiological barriers and corresponds to traditional network security solutions in the engineering problem. Moreover, both innate and adaptive immune systems recognize *Pathogenic* material. The first is a generic immune reaction, using built-in knowledge, and can be represented by PAMP²⁴ signals (known infectious agents). In contrast, the second is a targeted *Pathogen*-specific immune response, after the Dc²⁵ present to T-Cells the *Pathogens* found. This way, the Danger Theory-based IDS functionality is very similar to the innate and secondary responses of the BIS.

²³Anomaly Intrusion Detection System

²⁴Pathogen-Associated Molecular Pattern

²⁵Dendritic Cell

Despite the strong suitability of Danger Theory principles used for the design of A-IDSs, there are some limitations with this analogy [255]:

- *Danger Signal characterization*: It was mentioned previously that, in the presence of a *Pathogen*, the Danger Theory always assumes some form of damage to the system while being measurable using *Danger Signals*. However, this works if the *Pathogen* always causes the same type of distress (a process of cell death known in biology as *Necrosis*), which does not apply to engineering systems. It is known that different cyber attacks have different impacts on the system, so the damage characterization, i.e., *Danger Signals* specification is not always the same. This poses a limitation of fully *Danger Signals* coverage, i.e., there may be attacks that damage the system in such ways that are not represented by any *Danger Signal*. Moreover, it is not yet clear if all types of attacks result in observable and measurable damage, i.e., if their impact can be represented as one or more *Danger Signals*.
- *Safe Signal characterization*: Complementing with the previous point, the Danger Theory defines *Safe Signals* as the opposite of *Danger Signals*, i.e., used to characterize safe contexts. *Pathogens* collected in safe contexts will not trigger an immune response. *Safe Signals* help in activating tolerance so that no reaction will occur against no dangerous nonself. The problem is that, based on this analogy, *Safe Signals* may not be enough to avoid false positives in the detection. Similar to *Danger Signals*, it is not clear how to specify *Safe Signals* together with the other input signals and how to relate those with different problem features. This is a rather empirical process, so different problems and implementations characterize and specify *Safe Signals* (and *Danger Signals*) according to experimental results.
- *Time-base correlation*: In the Danger Theory, the correlation between *Pathogens* and the environmental signals is done on a time basis. This makes sense because, when collecting a *Danger Signal*, due to infected body cells, the *Pathogen* responsible is most likely present in the neighborhood of the infected cells. So, we easily can correlate damage with the responsible *Pathogen*. However, this is not always the case in engineering problems. In typical anomaly detection, the collection of *Danger Signals* and respective *Pathogen* may be unsynchronized, i.e., a distress signal associated with an attack may be observed long after the malicious *Antigen* has been collected. This delay may result in the correlation of the *Danger Signals* with *Pathogens* that are not responsible for the respective distress signals.
- *Antigen sampling*: During a *Pathogenic* infection within the human body, it is natural the presence of a large number of the same type of *Pathogens*. The *Antigen* correlation process is lightweight because we can assume only a sample of the *Pathogens* are considered for correlation purposes. However, this *Antigen* sampling analogy may not be valid in engineering systems since it is normal to have multiple different *Antigens*, according to the input data of the problem considered. This means that considering only a sample of the *Antigens* will not be effective due to the absence of sufficient *Antigens*.

- *Inefficient detection*: In case of infection due to a new type of *Pathogen*, the human body falls ill while the *BIS* collects the distress signs and presents the respective *Antigens* for an effective immune response (creation of *Antibodies* that can neutralize the *Antigens* presented). The human body is expected to recover from the illness when this happens. When referring to engineering systems, this analogy may not be suitable. Note that, by the time a successful detection and deployment of a mitigation response is achieved, it could already be too late to prevent irreversible damage from occurring to the underlying system. This means that the detection speed is crucial and can make a difference in whether the system can recover from the attack.

5.3 Requirement Analysis

Since the *DCA* is the most representative Danger Model approach, the majority of the research focuses on the specification and development of *DCA*-based approaches. There were proposed several variants of the *DCA* over the past years. Despite maintaining the original functionality, these variants improve specific components of the algorithm, typically to overcome one or more limitations. On the other hand, variant proposals may also be developed considering specific features of different application scenarios, such as intrusion detection in *WSNs*²⁶ and *IoT*²⁷ environments. Based on the context described before, this section provides a requirement analysis in the form of a *SWOT* analysis.

5.3.1 SWOT Analysis

Considering the *DCA* approach, Gu *et al.* [256] and Chelly and Elouedi [257, 168] provide a comprehensive analysis of the *DCA* and identify the main beneficial features and weaknesses of using the *DCA* in supervised *ML* anomaly detection or binary classification problems (usually in computer security scenarios). Next, it is listed all the internal factors (*DCA* strengths and weaknesses) and all the external factors (opportunities and threats) related to anomaly-based detection in industrial contexts.

5.3.1.1 Strengths

1. The algorithm is empirically lightweight in terms of running time since it employs computational and algorithmic simple linear functions for the classification.
2. The algorithm can remove high-frequency noise of the input data due to its filtering properties.
3. The algorithm performs temporal correlation that links anomalies to the potential causes, achieving a cause-effect relationship property.

²⁶Wireless Sensor Networks

²⁷Internet of Things

4. No training data is required.
5. The algorithm is insensitive, in terms of true positives, to a range of variations in system parameters, such as antigen vector size and transforming weight values.

5.3.1.2 Weaknesses

1. Lack of a formal definition and computational complexity analysis of the [DCA](#).
2. The algorithm's analysis and actual classification phases are performed once at the end of the algorithm's runtime, considering the entire dataset entries.
3. The preprocessing phase of the algorithm is performed manually, considering expert knowledge of a given problem domain. This includes feature selection/reduction and signal categorization (mapping between selected features and signal categories).
4. Variations in system parameters may lead to the generation of false positives.
5. The algorithm is considered only as a binary classifier.
6. Despite not relying on pre-defined baseline normal behavior, knowledge of what is normal and anomalous requires the categorization of input signals.
7. Because most of the research focused on updating different algorithm components for classification and intrusion detection performance, work about the application architecture was left behind. This resulted in uncertainty about how to use the algorithm in distributed scenarios.
8. Use of weights for signal transformation contributes to the algorithm's stochastic nature.
9. Despite enabling the algorithm to be lightweight, the simple linear functions (classifiers) limit the usage of the algorithm on complex and non-linear problems.
10. The algorithm is sensitive to the data order in the datasets, i.e., data should be ordered by the data classes (normal and anomaly) to avoid reduced classification performance.

5.3.1.3 Opportunities

1. New intrusion's variations are constantly being deployed towards industrial enterprises.
2. Implementation of [IIoT](#) environments are resulting in an increased attack surface towards industrial systems, usually associated with critical physical damage.
3. There is an increasing usage and deployment of industry-oriented [IP](#)²⁸-based protocols, such as [OPC UA](#).
4. Enable smart services and enable "intelligence" in shop-floor Edge devices is an increasing trend in [CPPSs](#).

²⁸Internet Protocol

5. There is now more consensus about data science as an independent discipline. Nowadays, it is a significant field of research and a tool for businesses and companies of all sizes. Modern data science technologies and techniques enable breaking down big data into usable information. Data scientists widely use *Python* coding language since it has in-built mathematical libraries and functions, making it easier to calculate mathematical problems and perform data analysis.
6. Online learning is becoming widely adopted in areas of **ML** where it is computationally infeasible to train over the entire dataset, in situations where the algorithm must adapt to new patterns in the data dynamically, or when the data itself is generated as a function of time.

5.3.1.4 Threats

1. Unavailability of public industrial datasets and benchmarks to validate **NIDS**s applied in the industrial context.
2. There is a tendency of embedding to existing industrial processes and equipment Edge devices, usually resource-constrained.
3. In industry, there are usually restrictions in training phases, since collecting only normal samples in that period may be difficult and, most of all, makes the system vulnerable to possible attacks.
4. The time of attack detection and corresponding reaction is critical in **IDS** applied in the industrial context.
5. Implementing **AIS** in **SMS** scenarios may require a significant digital transformation in the manufacturing industry, which involves undesirable costs and efforts to deploy several changes to existing systems.

5.4 Research Strategies

Based on the previously listed **DCA** strengths and weaknesses, and the opportunities and threats for industrial **A-IDS**, Table E.1 in Appendix E presents a **SWOT** analysis, considering the main strengths and weaknesses of the **DCA** that can be used to take advantage of opportunities and avoid threats of industrial **A-IDS**. Considering this analysis, next it is presented the literature review considering the four main strategy domains considered in the **SWOT** matrix, namely *Opportunities-Strengths*, *Opportunities-Weaknesses*, *Threats-Strengths*, and *Threats-Weaknesses*. With this, we want to identify the gaps in the existing research regarding **DCA** for **IDS** and opportunities for further contribution.

5.4.1 Opportunities-Strengths

The *Opportunities-Strengths* strategies highlights are the development of **NIDSs**, based on the **DCA**, which can be deployed in Edge devices of an **IoT** environment, and they are capable of processing and analyzing industrial network data. To the best of our knowledge, there is no work that applies **DCA** in **IIoT** environments. However, there are several approaches describing immune-models in a more generic **IoT** environment, which will be described next. Roman *et al.* [258] analyzed the requirements for the development of immune-based approaches for **IoT** and proposed a security architecture, designated virtual immune cells, that satisfies these requirements, such as on-demand immune services, flexibility, lightweight, and adaptability to novel threats. Al-rubayyi *et al.* [182] provides a review on the recent advances in employing **AIS** for the improved detection of malware in **IoT** networks. On the other hand, Pamukov [259] reviews and compares different **NSA** and **DCA** algorithms applied in **IoT** environments.

In 2006, Kim *et al.* [260] discussed for the first time ideas for the application of the **DCA** in **WSNs** and the detection of specific attacks that can disrupt multiple data paths in the network. The proposed approach, designated **UDCA**²⁹, tries to match the specific structure and functional requirements of **WSNs**. Despite presenting remarks about the approach's success, unfortunately, validation results are not presented to support those remarks. Later, Liu *et al.* [261, 262] provide **AIS**-inspired algorithms for intrusion detection in **IoT**, and included some Danger Theory principles. Despite dynamically detecting security threats in **IoT** environments, all the experiments, including the **IoT** environment, are simulated, i.e., no real attacks were used in this experiment, and the approach was not implemented in a real **IoT** scenario. Also, little results and conclusions about the work were given. More recently, Parrend *et al.* [263] set the foundation of *Artificial Immune Ecosystems* as distributed architectures capable of decentralized sensing, analysis, and reactions to anomalies. They proposed the **AWA**³⁰ to model the architecture of such ecosystems. **AWA** can detect intrusions in specific **IoT** architectures but is not suitable to detect outliers–anomalies.

Considering **DCA**-based approaches, Aldaheri *et al.* [264] proposed the **DeepDCA**³¹, a novel Deep Learning (**SNN**³²) and **DCA**-based **IDS** framework for **IoT** environments. The authors used **IG**³³ for feature selection/reduction and **SNN** for signal categorization, which overlaps with *Threats-Weaknesses* strategies (Section 5.4.4) regarding automation of the **DCA** pre-processing stage. The *Bot-IoT* dataset was used in the method's validation. Results show that **DeepDCA** outperforms classifiers such as **SVM**³⁴, **KNN**³⁵, and Naive Bayes in detecting accurately **IoT** attacks with low false positives.

Regarding industrial applications of **DCA**-based **NIDSs**, according to the analysis provided

²⁹Ubiquitous Dendritic Cell Algorithm

³⁰Artificial Awareness Architecture

³¹Deep Learning and Dendritic Cell Algorithm

³²Self Normalizing Neural Network

³³Information Gain

³⁴Support Vector Machine

³⁵K Nearest Neighbor

in the SLR³⁶ in Chapter 4, there are two proposals, one in the Robotics [196] and the other in the Smart Grid [186] domains. Degeler *et al.* [196] offer a DT³⁷-based IDS to safeguard a fleet of AGVs³⁸ from cyber-attacks that might cause severe physical damage to the production system in the Robotics area. The system is evaluated on a simulated manufacturing shop-floor using a distributed scenario (three AGVs working at the same time). Furthermore, in the Electric Power Industry, Igbe *et al.* [186] assess the efficacy of a dDCA³⁹ implementation in detecting cyber-attacks on a dataset for a Smart Grid scenario by using the DNP3⁴⁰ protocol. However, the mentioned study treats the problem as a typical classification problem, validated using a DNP3 traffic dataset. Despite being applied in an industrial context, the approach is unsuitable for analyzing data streams and using it in an online setting. More recently, Pinto *et al.* [3] proposed the CDCA⁴¹, a novel variation of the DCA, developed to detect anomalies by analyzing equipment's physical sensor data. Despite being developed initially to monitor physical industrial processes, the authors also tested the approach successfully in the M2M using OPC UA [34], making it suitable to be used as a NIDS.

In short, there is a gap in the literature about DCA-based IDSs in IIoT environments. Despite the solutions deployed in more generic IoT environments, as mentioned before, there are specific industrial requirements that must be considered when developing and deploying IDS solutions in such scenarios. Thus, we propose an IDS strategy in manufacturing scenarios, based on the dDCA [5] and the novel DCA variant iDCA⁴² [4] (more information in Section 6.3 of Chapter 6). Moreover, we propose a deployment strategy of such solutions in CPPSs and IIoT environments, following the industrial standard IEC 61499 (more information in Section 6.4 of Chapter 6).

5.4.2 Opportunities-Weaknesses

The most critical *Opportunities-Weaknesses* strategies are: i) DCA formal analysis and definition; ii) enable online detection capabilities in the DCA; iii) automation of the DCA pre-processing stage; iv) enable security by design in IIoT applications, by deploying IDS solutions in industrial applications following industry standards such as IEC 61499.

Both the automation of the DCA pre-processing stage and the security by design in IIoT applications topics are overlapping with *Threads-Weaknesses* strategies. So a detailed literature review about these topics is provided in Section 5.4.4. Next, it is presented state of the art regarding DCA formal analysis and definition, and online detection capabilities in the DCA.

³⁶Systematic Literature Review

³⁷Danger Theory

³⁸Autonomous Guided Vehicleless

³⁹deterministic Dendritic Cell Algorithm

⁴⁰Distributed Network Protocol 3

⁴¹Cursory Dendritic Cell Algorithm

⁴²Incremental Dendritic Cell Algorithm

5.4.2.1 DCA Formal Analysis and Definition

Regarding **DCA** formal analysis and definition, Greensmith *et al.* [163, 265] made the first attempt of formal specification of the **DCA** by describing in detail the algorithm and its use as an anomaly detection approach. The algorithm quickly gained much attention from the research community and was used not only in anomaly detection scenarios but in a wide range of real-world applications. Because the **DCA** is a highly stochastic algorithm, and due to the interest of the research community, a variety of **DCA**-inspired approaches were proposed. On the one hand, the lack of a better formal specification led to ambiguities and misunderstandings about the functionality of the original algorithm. On the other hand, despite overall presenting good performance results, the highly stochastic nature of the algorithm, i.e., the high number of random-based elements, led to several constraints regarding the analysis of the algorithm and understatement of the relationships between performance sensitivity and parameter adjustment.

For this reason, Greensmith and Aickelin [167] proposed the **dDCA**, a version of the **DCA** that has a controllable amount of stochastic parameters. This way, by eliminating most stochastic parameters and simplifying the signal processing calculation present in the original implementation of the **DCA**, the authors were able to maintain intact all the theoretical fundamentals supporting the algorithm. To this day, the **dDCA** is the most used algorithm to support other approaches based on **DCA** principles. Despite providing a clear description of the modified features and main differences between **dDCA** and **DCA**, a formal specification of the **dDCA** is missing.

Later, Gu *et al.* [266] presented a formal mathematical formulation of the **dDCA** in order to comprehend the whole algorithm better and provided a runtime analysis to reveal its theoretical features complexity. Because prior studies focused mostly on empirical factors, Gu *et al.* work assisted the research community in understanding the algorithm and avoiding ambiguity during the design and development phases. More recently, after concluding that the **DCA** was still described in multiple different ways, sometimes resulting in incorrect implementation, Greensmith and Gale [267] suggested a **dDCA** definition using functional programming, designated **hDCA**⁴³. The **hDCA** is specified in the *Haskell* programming language, which, unlike the prior work, is aimed at the computer science audience.

As a result of Gu *et al.*, Greensmith and Gale work, we used the **dDCA** specification as basis to develop the **iDCA**, which is an **OOP**⁴⁴ approach to the **dDCA** implemented in *Python*. More detail is provided in Chapter 6 (Section 6.3).

5.4.2.2 Online Detection in the DCA

The **DCA** algorithm can potentially process an unlimited number of input data points in an online fashion. After the pre-processing stage, the algorithm collects data points, which are converted to an *Antigen* form, and are correlated with the environmental signals (*Danger* and *Safe*). Each *Dc* profile active in the population is updated and, later, occurs the classification of sampled *Antigens*.

⁴³*Haskell-based Dendritic Cell Algorithm*

⁴⁴*Object-Oriented Programming*

However, although the data points are being collected and processed in an online fashion, suitable for data streams, this does not mean that the classification of the same data points is done in an online fashion. The algorithm only moves to the classification phase when all the *Antigens* are sampled and processed. This means that the algorithm must collect all the data points in the dataset before beginning the classification. When monitoring real-time data, the trick is to define the length of the monitoring session, corresponding to the number of data points the user wants to process and classify. This method ensures that results are analyzed after the length of the monitoring session.

The shortcoming of the previously mentioned method is the unavailability of providing classification in real-time, unlike the processing of *Antigen* and events that are processed in real-time. When the algorithm collects and processes an *Antigen*, the classification only occurs when all the *Antigens* are processed. The algorithm can take up several hours on a large-scale dataset until starting the classification phase. This is highly undesirable and unsuitable to anomaly/intrusion detection problems, where the acceptable delay between data collection and classification should not surpass a few seconds/minutes.

Considering the limitations of using *DCA* in online classification tasks such as anomaly/intrusion detection, Gu *et al.* [268] presented a real-time or near real-time analysis component by segmenting the current *dDCA* output into slices. The authors used the *dDCA*, because of its simplicity for further result analysis, but had the necessity of using components from the original *DCA*, such as the *PAMP* signal and weights for signal transformation. The new segmentation component can be *Antigen* or time-based, and it is deployed when mature *Dcs* present processed *Antigens*. This enables the algorithm to perform periodic analysis, i.e., the classification stage is performed in parallel in multiple smaller segments. The suggested technique was evaluated in the *SYN scan* dataset [269], a sizable real-world dataset based on a medium-scale port-scan of a university computer network. The results suggest that segmentation can be embedded to the *dDCA* for real-time analysis since it generates better results in some circumstances than the traditional approach.

As for the downsides of this approach, even though results demonstrate that segmentation may increase *dDCA* performance when compared to a version without segmentation, segments are the equivalent of mini data batches. Therefore this approach still follows a batch learning strategy. This means that the approach does not yet provide real-time analysis, as the segmentation still occurs after a time delay, i.e., *Antigens* are not classified as soon as they are collected. Also, due to the complexity of adding a segmentation approach, the authors needed to reintroduce to the *dDCA* stochastic elements from the original *DCA*. Thus, this approach may not be generalized to some industrial applications with actual real-time constraints.

Later, Yuan and Chen [270] also proposed a real-time analysis to the *DCA*, in which *Antigens* collected by a given *Dc* are analyzed as soon as the *Dc* migrates to a mature or semi-mature state. This means the *mcav*⁴⁵ is calculated based on the number of times each *Antigen* has been assessed and the times it has been assessed as an anomaly. The suggested method was validated in the

⁴⁵Mature Context Antigen Value

UCI Wisconsin Breast Cancer dataset [271]. The results are promising, demonstrating that the algorithm achieves a high level of accuracy. However, there are no results regarding the time of classification. The proposed solution is similar to the work of [268]. Despite avoiding segments, the proposed approach is still batch-based, so *Antigens* will not be classified immediately. Also, the approach is based on *DCA* and not on *dDCA*, which means that all the stochastic algorithm's variables were considered.

A similar approach is proposed by Pinto *et al.* [3]. To enable near real-time anomaly detection, the authors calculate *mcav* values of all *Antigens* collected each time a *Dc* matures. The main difference is that the approach consists of a variation of the *dDCA* and validation was performed in industrial-oriented datasets for anomaly and intrusion detection.

In short, there is a gap in the literature about *DCA*-based *IDS*s suitable for online anomaly detection. Despite the efforts of segmentation and near real-time classification of *Antigens* analyzed by a given *Dc*, such approaches are undesirable in industrial scenarios, which require continuous and instant classification of data points collected. Thus, we propose the *iDCA* [4] that enables online incremental detection in the *dDCA*, with little classification performance compromising and avoiding the stochastic elements of the original *DCA* (more information in Section 6.3 of Chapter 6).

5.4.3 Threats-Strengths

Within the *Threats-Strengths* strategies, strengths may be used to overcome threats. Considering the tendency of using embedded hardware, most of the time resource-constrained, at the Edge layer, and the restrictions of training phases, the application of *DCA*-based solutions for intrusion detection seems appropriate. The most notable strategy is the proposal of an industrial dataset containing *OPC UA* protocol-specific network traffic. The unavailability of public industrial datasets and benchmarks to validate such intrusion detection solutions is a threat to overcome.

According to Antón, Gundall, Fraunholz & Schotten [272], there are challenges in implementing industrial *IDS*s since there are limited publicly accessible datasets that can be utilized to evaluate intrusion detection algorithms. The *KDD Cup99* [88], *NSL-KDD* [273] and *UNSW-NB15* [91] are commonly used for benchmark *IDS*s. However, these datasets are based on characteristics of home and office-based networks and do not include typical network traffic based on industrial protocols, which presents specific characteristics.

Morris and Gao [274] proposed four datasets containing *MODBUS* network data that was collected in a laboratory environment using *SCADA* water and gas storage systems. The first dataset contains gas pipeline system data, while the second dataset contains data from a water storage tank. The other two datasets are a reduced size (10%) of the first two mentioned. All the datasets are in the *ARFF*⁴⁶ format and contain both network traffic (communication patterns in *SCADA*) and payload content (current state of *SCADA* systems) features. Moreover, the datasets contain 28 attacks of 4 different attack categories, such as reconnaissance, response injection, command

⁴⁶Attribute Relationship File Format

injection, and DoS⁴⁷. The downside of these datasets is the lack of original features regarding packets since the authors already provided aggregated and pre-processed data. On the other hand, despite being an essential industrial communication protocol, *MODBUS* is not an oriented IIoT protocol so the dataset may leave out important features about the dynamic of such systems. Finally, the datasets are only available in the *WEKA* data mining software [275].

Later, Lemay and Fernandez [276] presented several *MODBUS*-based datasets generated in a *SCADA* sandbox that are publicly available to security researchers⁴⁸. These datasets include different configuration values and provide different attack scenarios to increase the reliability of proposed intrusion detection scenarios. Furthermore, the datasets contain class labels that make them suitable for supervised learning approaches. The main drawback of these datasets is that they were built based on simulated industrial processes.

More recently, Antón, Gundall, Fraunholz & Schotten [272] provide three datasets comprising operational *OPC UA* traffic from an industrial process, into which numerous attacks have been added. The testbed used for collecting data is based on a water level control process. Three different attack scenarios were considered: using reconnaissance and scanning attacks, response and command injection, and physical fault/damage injection in the process results. Moreover, the attacks were deployed using synthetic data (injected offline after collecting the network data) and real-world attacks deployed by the application hardware in real-time. In a distinct work, the same authors utilized the created datasets to test the performance of *ML* classifiers, such as *SVM* and Random Forest algorithms [277]. The main downside of this work is that the datasets are small-sized, and each one presents distinct attack scenarios. This may be ok to analyze root-cause effects while applying *IDS*, considering specific attack scenarios. However, a larger dataset containing several attack scenarios is missing to test the robustness of classifiers. Also, most of the information included in the data set is process-based, neglecting network-based information. To this date, the datasets are not available publicly.

Finally, Igbe, Darwish & Saadawi [186] also presented a dataset for a Smart Grid scenario that included *DNP3* traffic. In this case, the dataset was used to assess the effectiveness of a *dDCA* implementation for the detection of cyber-attacks on Smart Grids. The main drawback is the public unavailability of the dataset. Also, despite being an essential protocol in Smart Grid scenarios, *DNP3* is not found in discrete manufacturing applications. To the best of our knowledge, this was one of the few industrial datasets used to validate a *DCA*-based approach for anomaly detection.

Thus, by analyzing state of the art, there is a gap in the literature regarding publicly available industrial-based datasets that contain specific protocol network data, such as *OPC UA*, for intrusion detection purposes. For this reason, we propose a new *OPC UA* dataset, designated *M2M using OPC UA* [34]. This dataset focus on typical *M2M*⁴⁹ communication in IIoT scenarios. In

⁴⁷Denial of Service

⁴⁸https://github.com/antoine-lemay/Modbus_dataset

⁴⁹Machine-to-Machine

this case, it was created a CPPS testbed to collect network data to be further included in the dataset. More information can be found in Section 6.2.1.

5.4.4 Threats-Weaknesses

The main *Threats-Weaknesses* strategies are the automation of the DCA pre-processing stage and the IIoT security by design, developing security solutions following industry standards such as IEC 61499. Next, we start by presenting the literature review regarding the automation of the DCA pre-processing stage. Then, the topic of security by design in IIoT applications is addressed in more detail.

5.4.4.1 DCA Pre-Processing Automation

Many works focused on investigating the DCA data pre-processing phase, considering the algorithm's limitation of requiring manual expert knowledge for feature selection/reduction and input signal categorization. This is an undesirable characteristic of the algorithm since data is most likely over-fitted to the algorithm. Most of these works tackle ensemble approaches, where the DCA-based algorithm is intersected with a statistical or ML-based approach for the pre-processing data phase. Back at 2009, Gu *et al.* [278] used PCA⁵⁰ for data reduction and generating new features for further signal categorization. Later, other data reduction approaches were defined by the same author, which concluded that PCA presented better results [256]. More recently, Aldhaheer *et al.* [264] proposed DeepDCA, and used IG for feature selection/reduction and SNN for signal categorization. The drawback of dimensionality reduction techniques is the destruction of the underlying meaning of the original features. This makes the signal categorization impractical to understand. The reason is that the DCA, during signal categorization, needs to map each originally selected feature to each signal category.

For this reason, most of the research shifted from data reduction to feature selection techniques. Here, the majority of the work is proposed by Chelly and Elouedi. They first started by studying the effectiveness of using RST⁵¹ for the data pre-processing task [279, 280, 281]. The RST is used to remove unnecessary features while keeping the reduct, i.e., the subset of most informative features. The reduct and its core concepts are used for signal categorization. The first approach, designated RST-DCA⁵² [279] generates all possible reducts and keeps the reduct that has the minimal set of features. Then, one feature is used to categorize both *Danger* and *Safe* signals. Results show that RST provides much better performance in terms of accuracy in the data pre-processing task when compared with PCA.

Later, a second approach designated RC-DCA⁵³ is proposed [280]. The feature selection task works the same as the RST-DCA. However, while RST-DCA uses one feature to categorize both *Danger* and *Safe* signals, RC-DCA maps each feature to a signal category. Further results

⁵⁰Principal Component Analysis

⁵¹Rough Set Theory

⁵²Rough Set Theory-based Dendritic Cell Algorithm

⁵³REDUCT and CORE-based Dendritic Cell Algorithm

show that **RC-DCA** outperforms **RST-DCA**. However, both approaches were criticized because generating all the possible reducts seem pointless when, most of the time, only one reduct is required since the subset of features used belongs to that reduct. Discovering all possible reducts leads to a level of run time complexity that is prohibited in constraint Edge devices. To overcome these limitations, **QR-DCA**⁵⁴ [281] adopts the QuickReduct algorithm to enable only one reduct to be generated. **QR-DCA** can find a trade-off between classification accuracy and preserving the lightweightness of the solution, which makes this algorithm a good alternative to **RST-DCA** and **RC-DCA**.

RST-based approaches show great potential, but they suffer from one problem. Both approaches rely on crisp datasets to work correctly. When dealing with real values, these methods cannot explain whether two features are similar or how similar they are. Data discretization can be used, enabled by a Quantization technique, to discretize values beforehand. One significant downside is the information loss associated with the discretization process. Thus, Fuzzy logic is used to create associated fuzzy sets that can be viewed as a pre-categorization step. The first fuzzy-rough **DCA**, designated **FRST-DCA**⁵⁵ [282], includes Fuzzy logic to the previously **RST**-based approaches. Feature selection is performed after generating membership functions and the fuzzy equivalent class for each feature. Results have shown that it is more appropriate to use fuzzy-rough solutions instead of just rough solutions. A similar solution, designated **FLA-DCA**⁵⁶ [283], uses a different concept for feature selection, i.e., the fuzzy lower approximation and the fuzzy similarity relations. On the other hand, **FBR-DCA**⁵⁷ [284] is another fuzzy-rough solution for feature selection by using the fuzzy lower boundary region and fuzzy similarity relations.

Later, Zhou and Liang [285] proposed a new approach for the **DCA** signal categorization based on numerical differential approach. The numerical differential can be seen as a method of studying data changes and trends to perceive danger (signal acquisition and fusion). Results show that this approach outperforms related methods, such as **PCA**, **RST** and **FLA-DCA** based. More recently, Pinto *et al.* [3] proposed a data-driven approach, namely K-means, to signal extraction in the **dDCA** pre-processing stage. The authors claim that this clustering algorithm is suitable because it can process new and unseen multi-dimensional data in real-time, provides a real-valued score (to be used as signals), and is scalable to data dimension and size. Provided results are promising.

5.4.4.2 IIoT Security by Design

Engineering **CPPS**s has been studied using many software designs and development methodologies, including component-based, formal models, agent-based, service-oriented architecture, design patterns, and **MBE**⁵⁸ [286]. **MBE** emerges as a possible alternative for cost reduction in

⁵⁴QuickReduct algorithm-based Dendritic Cell Algorithm

⁵⁵Fuzzy Rough Set Theory-based Dendritic Cell Algorithm

⁵⁶Fuzzy lower approximation-based Dendritic Cell Algorithm

⁵⁷Fuzzy boundary region-based Dendritic Cell Algorithm

⁵⁸Model-based Engineering

the design, validation, and verification of IIoT systems. It is feasible to decrease the engineering work for code development while preventing mistakes during development by employing and reusing models throughout the software design process. Furthermore, design validation may be accomplished early in the system's life cycle, especially important for Edge devices with limited computing resources.

Lindgren *et al.* [287] addressed the fundamentals of real-time execution of FB⁵⁹-based designs on IoT devices, i.e., lightweight controllers with limited memory and CPU⁶⁰ resources. Muthukumar *et al.* [288], on the other hand, offered a MBE technique for an IIoT architecture design, verification, and auto-code generation of control applications in process industries. In this scenario, the MBE technique was utilized to accomplish design and verification of the IIoT enabled control inside the benchmark problem 'quadruple tank process,' based on numerous perspectives of the system.

Until recently, security in manufacturing systems was primarily concerned with the interactions of operating network devices with the outside world, which overlapped with usual security solutions used in non-manufacturing applications. Digital cyber-attacks would primarily harm digital assets such as data and intellectual property in this scenario. However, with the rise of the IIoT and the increasing coupling of resource-constrained Edge devices with the digital world, cyber-attacks are increasingly being used to disrupt real-world processes [70].

This implies that while building and deploying CPPSs, security risk assessment and impact mitigation are critical. Sierla *et al.* [289] proposed a security risk analysis technique that included vulnerability and effect assessments. The study is performed on the communication network architecture of a co-simulation-based IEC 61499-based electric grid automation system. In this example, the concept was shown using a Smart Grid automation case study of FLISR⁶¹.

Tanveer *et al.* [290] proposed embedded encryption in the communication between PLCs, utilizing the IEC 61499 standard, in a study linked to design-level support for security in CPPSs, with an emphasis on inter-PLC communications. CL4FB⁶² is a suggested solution that consists of a security layer that enables encrypted data transmission using AES⁶³ and KE⁶⁴ by encoding IEC 61499 compliant CFBs⁶⁵ and SIFBs⁶⁶. The CL4FB's applicability was proven in a Smart Grid test case scenario utilizing an IEC 61499-based solution for protection and control functions in electric power distribution. The feasibility analysis evaluated the solution's influence on the overall system performance, primarily characterized by the added latency. The results reveal that, while the CL4FB introduces delay, most of the PLC's real-time restrictions can be satisfied.

Tanveer *et al.* [291] investigates the argument for providing security protection at the application level of PLCs in a separate work. Using IEC 61499 SIFB, the technique comprises of

⁵⁹Function Block

⁶⁰Central Processing Unit

⁶¹Fault Location, Isolation, and Service Restoration

⁶²Confidentiality Layer for Function Blocks

⁶³Advanced Encryption Standard

⁶⁴Key Exchange

⁶⁵Composite Function Blocks

⁶⁶Service Interface Function Blocks

introducing IDPS⁶⁷ capability to the PLC application level. The IDPS is introduced to the application during compilation and is used to avoid conventional PLC attacks on the application and device-level by analyzing network traffic between the network interface and the system's logic components. Based on the *Snort* tool [292], the strategy was evaluated in terms of Snort packets lost over a period while running in an IEC 61499 environment and installed in Wago PFC200 PLCs. Experiments demonstrate that the SIFB's IDPS-like feature, which provides active security protection against unknown (zero-day) threats, may successfully log and block assaults at the application level. When the severity of attacks is increased, the *Snort* performance analysis shows that the IDPS misses more packets and loses vital data for analysis. Actually, at greater assault intensities, the PLC device fails, resulting in total DoS.

The MBE technique was also utilized by Dowdeswell *et al.* [293, 294] to integrate the design and development of fault identification and diagnostic capabilities. The suggested solution, known as FDE⁶⁸, was meant to recognize and diagnose defects in IEC 61499 FB Pipelines, rather than detecting assaults and protecting communication channels, as prior studies had done. The solution may monitor the system's behavior using suitable fault detection methodologies, which are based on diagnostic agents that interface with the IEC 61499 SIFB. The FDE's viability was evaluated by running it with several agent instances in a HVAC⁶⁹ test case scenario. Results show no actual performance issues in the HVAC.

Tanveer *et al.* [295] have introduced abstract design extensions to the IEC 61499 development standard, designated *Secure Links*, that include both lightweight and conventional security measures into ICSs. These *Secure Links* assist in the consistent and reusable incorporation of various communication security techniques into IEC 61499 applications, depending on unique security needs. These requirements are described in standards like IEC 62443-4-2 and may be maintained with the TORUS⁷⁰ framework. This enables the construction of applications with security by design characteristics, reducing the need for manual security mechanism code. These security measures include TLS⁷¹ and AEAD⁷². Because they are completely compatible with the IEC 61499 standard, the generated applications may later be installed onto target Edge devices such as PLCs.

According to experimental data, *Secure Links* considerably minimize design and code complexity while improving application maintainability and requirements traceability. On the one hand, the delay of the encryption and key exchange processes is used to compare lightweight and TLS security techniques. System design complexity, on the other hand, is quantified in terms of structural S^* ⁷³, DC^* ⁷⁴, and C^* ⁷⁵, as well as the MI ⁷⁶ of the suggested solution. A higher MI

⁶⁷Intrusion Detection and Prevention System

⁶⁸Fault Diagnostic Engine

⁶⁹Heating, Ventilation, and Air-Conditioning

⁷⁰Traceability of Requirements Using Splices

⁷¹Transport Layer Security

⁷²Authenticated Encryption with Associated Data

⁷³Structural Complexity

⁷⁴Data Complexity

⁷⁵System Complexity

⁷⁶Maintainability Index

Table 5.1: *Dendritic Cell Algorithm-based Intrusion Detection System* literature gaps.

Proposed Approaches	Research Strategies					
	(a)	(b)	(c)	(d)	(e)	(f)
Gu <i>et al.</i> [266]	✗	✗	✓	✗	✗	✗
hDCA [267]	✗	✗	✓	✗	✗	✗
Gu <i>et al.</i> [268]	✗	✗	✗	✗	✗	✗
Yuan and Chen [270]	✗	✗	✗	✓	✗	✗
Antón, Gundall, Fraunholz & Schotten [272]	✗	✗	✗	✗	✓	✗
Gu <i>et al.</i> [278]	✗	✗	✗	✗	✗	✓
RST-based [279, 280, 281]	✗	✗	✗	✗	✗	✓
Fuzzy-based [282, 283, 284]	✗	✗	✗	✗	✗	✓
Numerical Differential [285]	✗	✗	✗	✗	✗	✓
CDCA [3]	✓	✗	✓	✓	✗	✓
This work	✓	✓	✓	✓	✓	✗

value suggests a better maintainable software, but a significant C^* value indicates a more complex system.

Table G.1 in Appendix G presents a summarized overview for comparison purposes of the main features found in the previous proposals described regarding IEC 61499-based security solutions. Also, this comparison includes the features of the solution proposed in this Thesis (more detail in Section 6.4). These features are related with: i) *Methods*: the security methods implemented; ii) *Application Scenario*: the context of the application of the methods implemented; iii) *Data Analyzed*: Type and origin of the data collected and analyzed by the security method; iv) *Tools*: Technologies used to develop and deploy the proposed solution; v) *Attack/Fault Spectrum*: Types of cyber-attacks or anomalies supported by the security method methodology; vi) *Assessment*: Evaluation methodology and results achieved while validating the proposed approach.

5.5 Proposed Approach

According to the previous SWOT analysis about the DCA implementation to the intrusion detection problem, especially considering the demanding requirements of such solutions in industrial applications. The main gaps are: a) DCA-based IDS in IIoT environments; b) CPPS security by design using IEC 61499; c) DCA formal analysis and definition; d) DCA online detection; e) NIDS OPC UA-oriented dataset; f) DCA automatic pre-processing stage. Table 5.1 summarizes these gaps, along with existing strategies and solutions in the literature.

This study addresses the majority of open strategy gaps in the literature identified before. The main output is a DCA-based IDS solution applied in IIoT environments (strategy-(a)), designated iDCA. Later, this approach is reformulated following the IEC 61499 for CPPS security by design

(strategy-(b)). The **iDCA** was developed as a new approach for the online detection (strategy-(d)), by implementing an incremental learning approach. Also, despite not being covered in the proposed approach, the **iDCA** uses an automatic strategy for the pre-processing stage (strategy-(f)), based on feature correlation, according to recent trends in the literature.

Figure 5.1 represents the **iDCA** activity diagram, assuming the four main stages in **DCA**-based approaches, namely *Pre-processing & Initialization*, *Detection*, *Context Assessment* and *Classification* (see more information in Section 3.4 - Chapter 3).

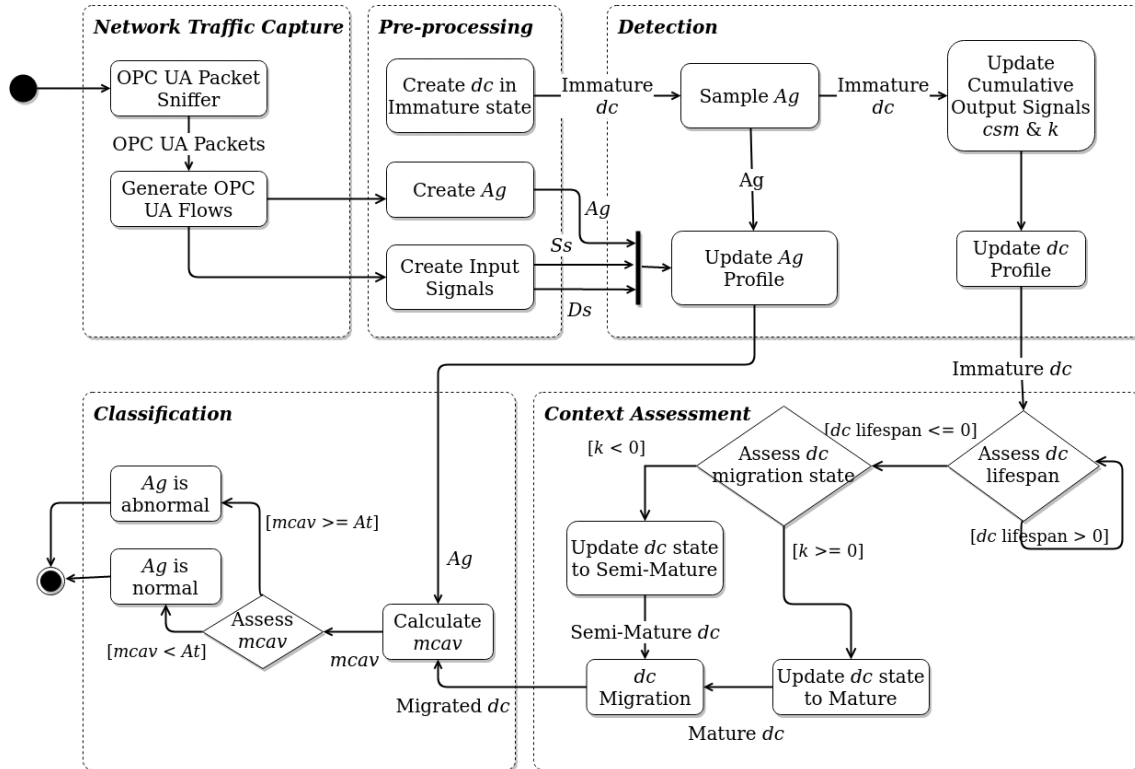


Figure 5.1: *Dendritic Cell Algorithm-based Intrusion Detection System in Industrial Internet of Things environments activity diagram.*

These four stages represent the **DCA**-based detection approach, designated **iDCA**. Besides the **iDCA** stages, and considering that the proposed **NIDS** solution is suitable for **OPC UA** network analysis, an initial *Network Traffic Capture* stage was considered, addressing the collection of **OPC UA** network packets and generation of flows. With the *Network Traffic Capture* stage, it was possible to create an **OPC UA**-oriented dataset (strategy-(e)), which is later used to validate both **dDCA** and **iDCA**. Finally, the proposed solution was formally defined to be used in object-oriented languages (strategy-(c)), considering all the stages represented in Figure 5.1.

5.6 Conclusion

A **SWOT** analysis is a technique frequently used as a structured planning method that evaluates the *Strengths, Weaknesses, Opportunities and Threats* of a project. We used this simple but power-

ful framework to handle the current situation of the literature, research gaps, and strategies about the development of a **DCA**-based approach for intrusion detection in industrial scenarios. We were able to identify both Strengths and Weaknesses of the immune approach while exploring Opportunities and recognizing potential Threats. This way, we can pinpoint and explore the capabilities of **DCA** to explore the Opportunities and avoid potential Threats considering industrial intrusion detection. Also, we can reveal Weaknesses and find a way to mitigate them, according to Opportunities to be explored.

Based on the **SWOT** analysis, the main research gaps are:

- a) **DCA**-based **IDS** in **IIoT** environments;
- b) **CPPS** security by design using **IEC** 61499;
- c) **DCA** formal analysis and definition;
- d) **DCA** online detection;
- e) **NIDS OPC UA**-oriented dataset;
- f) **DCA** automatic pre-processing stage.

Thus, the proposed solution tries to address most of these gaps. The proposed solution is described in the following chapter (Chapter 6), focusing on the **OPC UA** data collection, **iDCA** as detection approach, along with its **OOP** definition, and the **iDCA** support in **IEC** 61499 **FBs**. Later, Chapter 7 focuses on the comprehensive evaluation and assessment of the proposed solution.

Chapter 6

IDS Proposal

6.1 Introduction

In this chapter, the reader will learn about all the technical specifications and development details regarding the proposed approach described in Section 5.5 (Chapter 5). We describe in detail the different proposal components. They are: i) an OPC UA¹-oriented dataset for NIDS² validation; ii) the iDCA³ for online detection and a formal definition considering OOP⁴ concepts; iii) A-HIDS⁵ Pipeline, compliant with the IEC⁶ 61499 standard for CPPS⁷ security by design.

Section 6.2 focuses on the description of the OPC UA data collection methodology and the proposal of a new dataset, namely the *M2M Using OPC UA* [34]. Then, in Section 6.3, we present the iDCA, a novel DCA⁸-based algorithm, enhanced with OPC UA network data processing and classification in an online fashion. The iDCA is also formalized using an OOP approach. Finally, considering the iDCA proposed before, in Section 6.4 we present the A-HIDS architecture for embedding bio-inspired detection into industrial Edge devices, by exploring the IEC 61499 standard. In this case, the IDS⁹ architecture is deployed at the host level, i.e., at the Edge layer of the CPPS, and includes the iDCA for network intrusion detection as default.

Thus, the first output of the chapter is the *M2M using OPC UA* dataset [34], which can be used to assess the classification performance of NIDS techniques. The second output is the iDCA, a variant of the dDCA¹⁰ suitable to process OPC UA network data in an online fashion while detecting cyber-attacks deployed in CPPSs. The third output is a bio-inspired A-HIDS Pipeline, compliant with the IEC 61499 standard, which enables the iDCA by design in the CPPS for network intrusion detection purposes.

¹OPC Unified Architecture

²Network-based Intrusion Detection System

³Incremental Dendritic Cell Algorithm

⁴Object-Oriented Programming

⁵Anomaly Host Intrusion Detection System

⁶International Electrotechnical Commission

⁷Cyber-Physical Production System

⁸Dendritic Cell Algorithm

⁹Intrusion Detection System

¹⁰deterministic Dendritic Cell Algorithm

Partial results of the presented work in this chapter have been published in or submitted/accepted at:

- Pinto, R., Gonçalves, G., Delsing, J., & Tovar, E. (2021). Incremental Dendritic Cell Algorithm for Intrusion Detection in Cyber-Physical Production Systems. In *Intelligent Computing* (pp. 664-680). Springer, Cham. (Published) [4].¹¹
- Pinto, R., Gonçalves, G., Tovar, E., & Delsing, J. (2020, September). Attack Detection in Cyber-Physical Production Systems using the Deterministic Dendritic Cell Algorithm. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* (Vol. 1, pp. 1552-1559). IEEE. (Published) [5].¹²
- Pinto, R., Gonçalves, G., Delsing, J., & Tovar, E. (2021). Enabling Data-Driven Anomaly Detection by Design in Cyber-Physical Production Systems. *Cybersecurity*. (Published) [2].

6.2 OPC UA Data Collection

It was already mentioned that the proposed solution is framed as a **NIDS**, i.e., audit data consisting of network traffic. In this case, the **NIDS** is suitable to handle protocol-specific network data, namely **OPC UA**. Thus, this section addresses mainly the **OPC UA**-oriented dataset strategy (see Section 5.5) by providing detail about the data collection and pre-processing processes. Considering the activity diagram of a **DCA**-based **IDS** in **IIoT**¹³ environments (see Figure 5.1 in Section 5.5), the *Network Traffic Capture* stage consists of two main activities, namely the *OPC UA Packet Sniffer* and the *Generate OPC UA Flows*. The main objective in this stage is to collect **OPC UA** packets while sniffing such packets in a network and generate bidirectional **OPC UA** flows that consist of both request/reply packets between the same network nodes.

The *OPC UA Packet Sniffer* relies on *Pyshark*, a *Python* wrapper for the *TShark* tool [296] for live packet capture, by specifying a network interface for sniffing purposes, or parse packets from a **PCAP**¹⁴ file. *TShark* is a network protocol analyzer that performs packet capturing with the *pcap* library. The library supports the specification of a filter expression, so packets that do not match that filter are discarded. Thus, in order to sniff all **OPC UA** traffic within a given network interface, a display filter is used. This way, only **TCP**¹⁵/**IP**¹⁶ packets that present in their *Application* layer the **OPC UA** syntax are collected.

The *Generate OPC UA Flows* activity receives **OPC UA** packets provided by the *OPC UA Packet Sniffer* and generate bidirectional **OPC UA** network flows. First, all the relevant **TCP/IP**

¹¹ Adapted by permission from Springer Nature Customer Service Centre GmbH: Springer Nature Intelligent Computing, Pinto, R., Gonçalves, G., Delsing, J., & Tovar, E., Incremental Dendritic Cell Algorithm for Intrusion Detection in Cyber-Physical Production Systems, ©2021.

¹² ©2020 IEEE. Reprinted, with permission, from Pinto, R., Gonçalves, G., Tovar, E., & Delsing, J., Attack Detection in Cyber-Physical Production Systems using the Deterministic Dendritic Cell Algorithm, September/2020.

¹³ *Industrial Internet of Things*

¹⁴ *Packet Capture*

¹⁵ *Transmission Control Protocol*

¹⁶ *Internet Protocol*

and **OPC UA** information is extracted from different layers of the packet. Then, a 5-tuple hash is generated by relating the source and destination **IPs**, ports, and protocol used in the packet. This hash is used to identify bidirectional connections between two nodes in the network uniquely. Moreover, another hash is generated, according to the **OPC UA** message type, i.e., *HEL*, *ACK*, *MSG*, *OPN* or *CLO* and message sequence order. These generated hashes will be used later to create *Antigens*, i.e., an *Antigen* in our problem is a bidirectional network flow. Also, all the network information collected from packets will be embedded in the network flow as features, which are later used to derive both *Danger* and *Safe* signals. All flows are represented by 32 features listed next:

- *src_ip* - Source **IP** address;
- *src_port* - Source port;
- *dst_ip* - Destination **IP** address;
- *dst_port* - Destination port;
- *flags* - **TCP** flag status;
- *pktTotalCount* - Total packet count;
- *octetTotalCount* - Total packet size;
- *avg_ps* - Average packet size;
- *proto* - Protocol;
- *service* - **OPC UA** service call type;
- *service_errors* - Number of service errors in **OPC UA** request responses;
- *status_errors* - Number of status errors in **OPC UA** request responses;
- *msg_size* - **OPC UA** message transport size;
- *min_msg_size* - Minimum **OPC UA** message size;
- *flowStart* - Timestamp of flow start;
- *flowEnd* - Timestamp of flow end;
- *flowDuration* - Flow duration in seconds;
- *avg_flowDuration* - Average flow duration in seconds;
- *flowInterval* - Time interval between flows in seconds;
- *count* - Number of connections to the same destination host as the current connection in the past two seconds;
- *srv_count* - Number of connections to the same port number as the current connection in the past two seconds;
- *same_srv_rate* - The percentage of connections that were to the same port number, among the connections aggregated in *Count*;

- *dst_host_same_src_port_rate* - The percentage of connections that were to the same source port, among the connections having the same port number;
- *f_pktTotalCount* - Total forward packets count;
- *f_octetTotalCount* - Total forward packets size;
- *f_flowStart* - Timestamp of first forward packet start;
- *f_rate* - Rate at which forward packets are transmitted;
- *b_pktTotalCount* - Total backwards packets count;
- *b_octetTotalCount* - Total backwards packets size;
- *b_flowStart* - Timestamp of first backwards packet start;
- *label* - Binary label classification;
- *multi_label* - Multi classification labeling.

6.2.1 M2M using OPC UA

This section describes the *M2M using OPC UA* [34] dataset, which was generated by capturing [OPC UA](#) network traffic in a laboratory [CPPS](#) testbed, by using the *Network Traffic Capture* activities described before. This laboratory [CPPS](#) testbed uses the [OPC UA](#) protocol in a [M2M](#)¹⁷ communication environment. Figure 6.1 illustrates the testbed configuration.

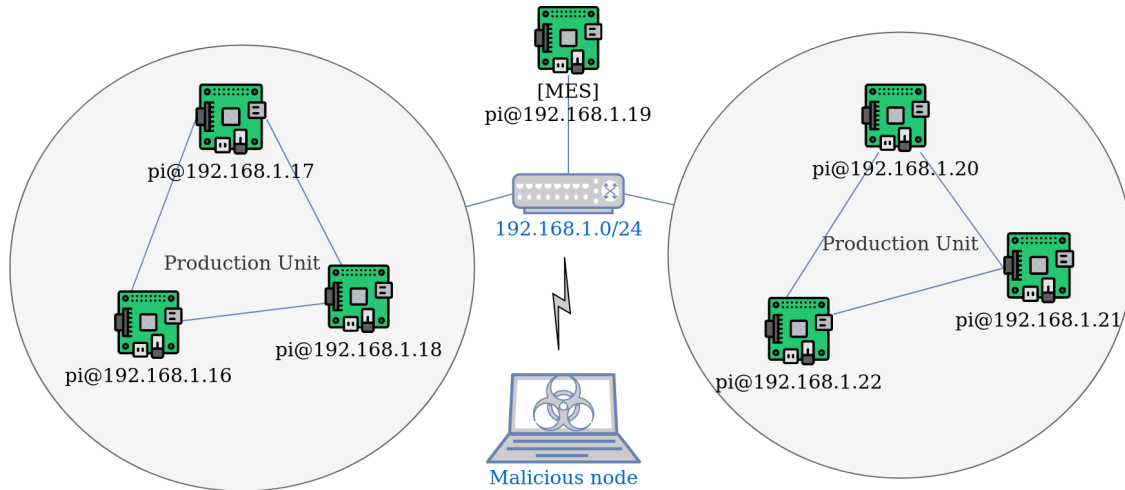


Figure 6.1: *Cyber-Physical Production System* testbed for the *M2M using OPC UA* generation.

Each network node consists of a *Raspberry Pi* device running the *Python FreeOpcUa* implementation [297]. In this configuration, there are two production units, each with three devices, and one connection node representing a [MES](#)¹⁸. A production unit can be seen as a logically isolated sub-network, where all nodes are visible and communicative. However, there is also the possibility

¹⁷Machine-to-Machine

¹⁸Manufacturing Execution System

of production units to be connected for message exchange, but there is a logical separation among their respective Edge devices. The only node that can 'see' all nodes and exchange messages transparently with multiple production units is the MES. Each node/Edge device implements a DINASORE¹⁹ instance, consisting of both an OPC UA server and a client. The server publishes updates of new sensor's readings to an OPC UA variable. In contrast, the client subscribes to all OPC UA variables from all other devices in the same production unit. The MES, on the other hand, only implements the OPC UA client, which subscribes to all OPC UA variables from all devices in both production units.

An attacker node is also connected to this network (it is assumed that the attacker already has gained access to the CPPS network). Thus, this node deploys several different network attacks during the CPPS operation. This attack injection in the network will enable the introduction of security vulnerabilities into the CPPS, so the dataset includes both normal operation data and anomaly data points, making it suitable to be used for NIDS performance evaluation.

After setting the CPPS testbed, the OPC UA Packet Sniffer activity captures network packets, while the Generate OPC UA Flows export OPC UA network flows to a tabular file format dataset (CSV²⁰). During the CPPS execution, the rogue node launches multiple attacks targeting different legit nodes, resulting in unusual behavior. Thus, the dataset will contain normal and deviant activity while capturing the network traffic. Attacks such as DoS²¹, MITM²², and Device Spoofing were introduced as a result of the malicious operations. A Python script that implements the Scapy [298] module for packet sniffing, injection, and manipulation is used to carry out the attacks mentioned before.

The produced dataset comprises 33 567 normal instances, 74 013 DoS attack instances, 50 Spoofing attack instances, and 7 MITM attack instances. This sums up to 107 634 instances. Furthermore, all attacks were classified as an anomaly, while the remaining instances were classified as normal, allowing binary classification techniques to be used in this dataset. The dataset is available in the IEEE²³ DataPort repository [34], and may be also accessed through a GitHub repository²⁴. Table 6.1 presents a comparison between the most known and public available NIDS datasets and the M2M using OPC UA.

In comparison with other network intrusion detection datasets, the M2M using OPC UA is the only one that contains industrial communication-related data, namely OPC UA oriented network traffic. In terms of attack types, the M2M using OPC UA includes only attacks that can be deployed typically by automated tools. The dataset size is acceptable, but it lacks the separation between training and testing sets. Also, in terms of attack ratio, the M2M using OPC UA is imbalanced, following identical attack ratios from other datasets. Finally, the number of features is significantly reduced compared with other datasets.

¹⁹Dynamic INtelligent Architecture for Software and MODular REconfiguration

²⁰Comma Separated Values

²¹Denial of Service

²²Man-in-the-Middle

²³Institute of Electrical and Electronics Engineers

²⁴<https://digi2-feup.github.io/OPCUADataset/>

Table 6.1: Comparison of different datasets for network intrusion detection.

	<i>KDD Cup99</i>	<i>NSL-KDD</i>	<i>UNSW-NB15</i>	<i>M2M using OPC UA</i>
Year	1999	2009	2015	2020
Attack Types			<i>Fuzzers</i>	
			<i>Analysis</i>	
			<i>Backdoors</i>	
	<i>DoS</i>	<i>DoS</i>	<i>DoS,</i>	<i>DoS</i>
	<i>Probing</i>	<i>Probing</i>	<i>Exploits</i>	<i>MITM</i>
	<i>R2L</i>	<i>R2L</i>	<i>Generic</i>	<i>Spoofing</i>
	<i>U2R</i>	<i>U2R</i>	<i>Reconnaissance</i>	
			<i>Shellcode</i>	
			<i>Worms</i>	
Train & Test Datasets	✓	✓	✓	✗
Reduced Size Version	✓(10%)	✓(20%)	✗	✗
Labelled	Binary and Multi-class			
Size	Train: 4 898 431 Test: 311 027	Train: 125 973 Test: 22 544	Train: 175 341 Test: 82 332	107 634
Attack Ratio	Train: 80% Test: 80%	Train: 47% Test: 57%	Train: 55% Test: 68%	69%
Number of Features	41	41	49	30
Protocols	TCP, UDP & ICMP			OPC UA (TCP)

6.3 Detection Approach

This section covers all the details regarding the proposed detection approach, designated **iDCA**. The **iDCA** embeds an incremental learning approach to the **DCA** to enable real-time data processing and intrusion detection. We start by describing a formal definition considering the **OOP** paradigm. Then, we explain in detail the algorithm by highlighting similarities and differences with the **dDCA**.

6.3.1 Object-Oriented Definition

OOP [299] is a fundamental part of software development and, when it first appeared, it was a huge leap forward from the traditional top-down method. With **OOP**, instead of writing a program, *Classes* are created, which contains both data and functions. This programming paradigm is based

on the concept of *Objects*, which can contain both information and behavior: the information, or data that characterizes the *Object*, exists in the form of data fields (often referred to as attributes or properties), and the behavior exists in the form of procedures (often known as methods or functions). Computer programs in **OOP** are meant to make *Objects* interact with one another. A class often defines a set of comparable *Objects* (those that have the same traits and behavior), implying that *Objects* are instances of *Classes*, which also determine the *Object* type.

Objects can be nearly everything in the real world, such as external entities (people, devices, systems), things (reports, displays, signals), events, roles, places, structures, among others. An *Object* is something that can be represented using properties and behaviors. On the other hand, *Classes* are "templates" that describe and define properties and behaviors of *Objects*. In this scenario, *Object* properties can be specified using *Class* attributes, while *Object* behaviours can be specified using *Class* methods. Finally, a constructor is a special method, with the same name as the *Class*, which is used to create a new *Object* of that *Class*.

The main advantages of **OOP** are:

- **Modularity:** *Objects* and *Classes* are self-contained. This means that troubleshooting is easier in **OOP** since the developer does not need to go line-by-line through all the code to find a bug or a specific problem.
- **Reuse of code:** In **OOP**, different *Objects* have commonalities between them. Thus, by considering the inheritance property, the developer can create one generic *Class* and then define subclasses that are to inherit the generic *Class*'s traits. This way, the developer saves time since it is much faster than developing all *Objects* separately. Also, it is possible to eliminate redundant code, so it saves both time and space.
- **Flexibility:** Despite sharing some traits, different *Objects* also present unique behavior. By using the polymorphism property, a single function can shape-shift to adapt to whichever *Class*. Thus, a developer can create one generic function in the parent *Class* that can work for all the specific subclasses.
- **Effective problem solving:** When tackling big problems, **OOP** is an easier and more intuitive approach. With **OOP**, the developer can break a big problem into smaller ones, which can be tackled with different *Classes* or modules. Also, these *Classes* can later be reused, which makes it even quicker to solve the following problem. Since this approach divides the entire task into several *Objects*, it reduces complexity as each *Object* is responsible for a particular sub-task.
- **Maintenance:** It is easy to modify and maintain existing **OOP** code as there is the capacity to create new *Objects* with minor differences to already existing ones.
- **Data privacy** – The data and functions that make up a *Class* can be hidden from other *Classes*. Thus, it helps a developer write securer programs.

- Easy to understand: Typically, due to its modularity, a developer can easily modulate an OOP code using standardized language for modeling systems, such as UML²⁵. With UML, developers can visualize, construct and document their OOP-oriented systems using integrated sets of diagrams, such as UML Class Diagram, UML Sequence Diagram, and many others.

Considering the previously mentioned benefits and the importance of OOP in software development, in this section is presented an OOP-oriented approach to the iDCA. By following a OOP approach, we could update and maintain the algorithm's source code quickly, since the new features, i.e., the behavior that we intended to introduce, are easily defined within specific *Classes*. Also, we could easily modulate the iDCA using UML, thus enabling the documentation of the software using a standardized modeling language. Consequently, this will help communicate and understand the algorithm among other engineers.

Figure C.1 in Appendix C represents the iDCA UML Class Diagram. The iDCA UML Class Diagram represent the structure of the implemented algorithm regarding its classes and objects. It is possible to identify class' attributes and operations, as well as relationships between classes and restrictions imposed by the system. In this case, there are 6 main classes, namely: *Sniffer*, *Packet*, *Flow*, *iDCA*, *Antigen* and *Cell*.

- *Sniffer*: is the Class that implements the input data collection, which can be network data (from both online and offline interfaces), or it can be from a dataset in a standard tabular format. The *LiveCapture* and *FileCapture* classes, belonging to the *TShark* library, are used if the data source is a live network interface or an offline PCAP file respectively. Also, the *Thread* class is used to enable a multi-thread approach since the network data collection should run in parallel with other tasks in the system, such as data processing and further classification.
- *Packet*: is the class that represents all packet objects. Actually, all TCP packets collected in the *Sniffer* (that use the OPC UA protocol in the *Application* layer) are processed as an object in the *Packet* class.
- *Flow*: is the Class that represents bi-directional flows, i.e., objects composed by one or more related packets. Typically these packets belong to the same request-reply sequence. Several network properties characterize each flow object (see Section 6.2). This Class enables writing the collected flows to a CSV file, implementing the *DictWriter* class (the *M2M Using OPC UA* dataset was created initially using this Class while sniffing a live network interface). Finally, the *Flow* class implements the *OrderedDict* class, which enables using ordered dictionaries to save temporal information about the flows collected. This is important because the flow object is what we want to classify as normal or anomaly, so the notion of the temporal relationships between different flows will be used later when creating *Antigens*.

²⁵ Unified Modelling Language

- *Antigen*: is the class that implements all the behavior related with *Antigens* and input signals (*Danger* and *Safe*), i.e., network flows. In this case, a *Antigen* is the unique identifier of the flow object (considering the temporal order of creation). At the same time, the input signals are a combination of the network features considered in the flow object (as mentioned before, the features to be used are selected after a correlation process). First, the values of the selected features are normalized, using the *MinMaxScalar* class from the *River* library [300]. Then, for each feature, the current mean (using the *Mean* Class from the same library) is used as a reference to comparison with the normalized current feature's value.
- *Cell*: is the class responsible for the management of the artificial *Dendritic Cell*, all the behavior of exposing *Antigens* to the *Dc*²⁶ population, and the consequent migration of the *Dc*. Thus, the cumulative *csm*²⁷ and *k*²⁸ signals are updated in this class, and the sampled *Antigens*' classification is performed.
- *iDCA*: is the Class that manages all the parameters of the algorithm and all the communication between *Antigen* and *Dc* objects. This Class provides the incremental logic to enable *Antigens* to be processed right after they became available (in sequence). Finally, this class implements the *Gaussian* class from the *River* library, which is used later in the *Antigen* classification stage to update the *mcav*²⁹ CDF³⁰ to compare it with the *At*³¹ in every iteration. More information about the functionality of the algorithm can be found in Section 6.3.2.

Different classes modulate two different complex processes. While *Sniffer*, *Packet*, *Flow* are related with the network data collection process, *iDCA*, *Antigen* and *Cell* are related with the behavior of the algorithm implemented. Since the UML Class Diagram is adequate to represent the system structure in terms of classes and objects, but not so good at representing the dynamics of the system, Figures C.2 and C.3 in Appendix C illustrate the UML Sequence Diagrams of both network data collection process and behavior of the *iDCA*. These diagrams identify system behavior, detailing high-level interactions between object instances in a collaborative context.

Considering the Network Data Collection UML Sequence Diagram in Figure C.2, there are several entities within the system that play an important role in this process. After instantiating a *Sniffer* object, designated *flowRecorder*, a *Flow* object is created. Then, depending on the operation mode, the *flowRecorder* can collect real-time data from a network interface (mode 1), offline packets from a PCAP file (mode 2) or read from a CSV dataset (mode 3). In both modes 1 and 2, *Flow* creates a *Packet* object for each network packet collected and, according to the packet, *Flow* objects may update their own state. This means that the recently created *Packet* object can be part of an existing *Flow* or, if not, a new *Flow* is considered. Finally, the *Flow* object instantiates an object of type *Antigen*. On the other hand, if operation is performed under mode 3, the dataset

²⁶*Dendritic Cell*

²⁷*Co-stimulatory Signal*

²⁸*Context Output Value*

²⁹*Mature Context Antigen Value*

³⁰*Cumulative Distribution Function*

³¹*Anomaly Threshold*

is collected as a Dataframe type. Then, by looping the Dataframe, each row corresponds to an *Antigen* object.

As for the *iDCA* UML Sequence Diagram in Figure C.3, it is represented the algorithm's behavior, which is explained in detail in Section 6.3.2. An *iDCA* object begins by creating a *Dc* population (instantiating *Cell* objects in an immature state). These cells are stored in the immature *Dc* list. Then, until a stop condition is met, e.g., no more flows to be processed, the network flow features are used to generate the input *Danger* and *Safe* signals and return the updated *Antigen* object.

After the *Antigen* and input signal information are updated, the *iDCA* object orchestrates the execution of the algorithm's main stages. The method *incremental_process* consists in the sampling of the available *Antigen* by all the immature *Dcs*, followed by exposing all cells to the *Antigen*, i.e., update *csm* and *k* cumulative signals and test if each immature *Dc* is ready to migrate.

On the other hand, the method *incremental_classify* consists in the classification of the *Ags*³², i.e., calculating the *mcav* CDF and compare it with the *At*. The classification result (*class*) concludes the process iteration for the currently available *Ag*. Finally, when a stop criterion is met, the algorithm calculates by default several performance metrics, considering the classification of the *Antigens* sampled and the availability of a class label. Metric examples are the detection *Acc*³³, *ROC*³⁴ curve and *AUC-ROC*³⁵, *TPR*³⁶ and *TNR*³⁷. Note that this is only possible in mode 3, where the true labels of each flow are available.

6.3.2 incremental Dendritic Cell Algorithm

Incremental learning [301] refers to *AI*³⁸ that learn incrementally over data. These dynamic algorithms are helpful when: i) data becomes supplied progressively over time (stream data); ii) its size exceeds system memory restrictions; or iii) training an algorithm across the entire dataset is computationally infeasible. A different and more straightforward way is to do periodic model retraining with a batch algorithm, which involves retraining the model over and over again based on the data included in each batch. Even though the model is being adapted to new patterns in the data, it is critical to retain some recall of the previous data. This is only achievable with incremental learning, not batch learning (unless the whole database is loaded each time into the memory and the algorithm learns fresh from the beginning). Model retraining may be unfeasible in *DCA*-based classifiers since they do not have a training stage. Furthermore, incremental learning makes it feasible to deliver a result immediately after a data point becomes available, which is highly beneficial in data streams and problems with demanding classification speed. This is the case with

³²*Antigens*

³³*Accuracy*

³⁴*Receiver Operating Characteristic*

³⁵*Area Under the Receiver Operating Characteristic Curve*

³⁶*True Positive Rate*

³⁷*True Negative Rate*

³⁸*Artificial Intelligence*

industrial intrusion detection. On the other hand, even when using small micro-batches in batch learning, there is always a dataset gathering phase before outputting the results.

In the domain of anomaly detection [302], incremental learning provides several advantages over batch learning. First, it works best in autonomous systems, such as CPPSs, since there is continuous model adaptation based on a continually arriving data stream. What is considered a typical profile now may not be actual in the future. As a result, adjusting the normal baseline over time is crucial. Furthermore, many datasets might grow so large that dealing with them as a data stream is an ideal strategy. Finally, in industrial applications where detection speed is critical to minimizing the impact of an intruder in the system, incremental learning may produce detection results significantly faster than batch techniques.

The *iDCA* is characterized by the same four major stages as the *dDCA* (see Section 3.4), but with distinct procedures in each stage. The *iDCA* pseudocode is available in Algorithm 2. In the following sections, we will describe the similarities and differences between *dDCA* and *iDCA*.

In the *Pre-processing & Initialization* stage, concretely in the generation of *Ags*, *Ds*³⁹ and *Ss*⁴⁰, since data is available in a streaming format instead of a batch format, incremental data pre-processing methods are used for data normalization and input signal mapping. The *iDCA* uses a two signal model (*Ds* and *Ss*) instead of three, leaving out the *PAMP*⁴¹. An *Ag* is the ID of a network flow collected. This can be a unique ID generated for this purpose, in the case of online data processing or the row index, if a dataset is being used as data input. Regarding input signals, there are many ways to form *Ds* and *Ss* by considering the values of the features of each *Ag*. Detail about this can be found in Section 6.3.2.3. At each new data entry, the normalized (range of 0 - 100%) value and current mean for each feature selected are likewise updated to form the input signals. Note that, in the *iDCA*, *Ds* and *Ss* are properties of *Ags* and are not different entities to avoid issues in the time-based correlation between signals and *Antigen*. This way, it is possible to reduce the correlation between unsynchronized signals and *Antigen* in the *Detection* phase. However, if an attack impact takes longer than expected to become observable, this time correlation limitation still holds.

During the *Detection* and *Context Assessment* phases, all *Dcs* sample the unique *Ag* available at that time, which was already combined with the computed input signals. This allows the *Dc* profile to be changed, as the cumulative output signals are computed to determine the *Dc* migration status (Mature or Semi-Mature). Instead of collecting a batch of *Ags* for subsequent sampling, the *Dcs* sample only one *Ag* at a time since it is the only one available at the moment. This is done several times (equal to the *#Iter*⁴²). This means that for the time being, the updating of the *cs**m* and *k* output signals and migration of the *Dcs* are dependent only on a single sampled *Ag* and respective input signals. Note that, in the *iDCA* there is not any random *Ag* sampling. All *Dcs* sample multiple times the available *Ag*. This way, it is possible to reduce the negative effect of having insufficient *Ags* when sampling, such as non-classifications. Theoretically, processing all

³⁹*Danger Signal*

⁴⁰*Safe Signal*

⁴¹*Pathogen-Associated Molecular Pattern*

⁴²*Number of Iterations*

Algorithm 2: *Incremental Dendritic Cell Algorithm* pseudocode.

```

input   : #Dc, #Iter, At,  $t_m$ 
generate: Ag, Ds, Ss

1 Ag  $\leftarrow$  generate: Ds, Ss
2 for  $i \leq \#Dc$  do
3    $Dc \leftarrow \text{InitializeDC}(t_m)$ 
4 for  $j \leq \#Iter$  do
5   forall Dc do
6     ExposeCell(Ag)
7      $csm \leftarrow Ds + Ss$ 
8      $k \leftarrow Ds - 2Ss$ 
9     UpdateProfile(Dc)
10    if  $Dc.lifespan \leq 0$  then
11       $M \leftarrow \text{Migrate}(Dc)$ 
12      Reincarnate( $Dc, t_m$ )
13      if  $k \geq 0$  then
14         $Dc.state \leftarrow mat$ 
15      else
16         $Dc.state \leftarrow semi$ 
17 forall M do
18    $\#Ag \leftarrow \text{Count}(Ag)$ 
19   if  $Dc.state == mat$  then
20      $\#mAg \leftarrow \text{Count}(M)$ 
21      $mcav \leftarrow \#mAg / \#Ag$ 
22      $gau \leftarrow \text{UpdateGau}(mcav)$ 
23      $cdf \leftarrow gau.getCDF(mcav)$ 
24     if  $cdf \geq At$  then
25       context  $\leftarrow 1$ 
26       Ag  $\leftarrow$  'Anomaly'
27   else
28     context  $\leftarrow 0$ 
29     Ag  $\leftarrow$  'Normal'

```

the incoming *Ag*s during the correlation process is more effective than picking random *Dcs* to sample the *Ag*. Moreover, this may be more lightweight than the antigen multiplier work-around proposed in the original *DCA*.

Finally, the *mcav* calculation procedure is the same in the *Classification* step, but only one sampling *Ag* is examined at a time. However, on the contrary of the *dDCA*, where the *mcav* is compared with the anomaly threshold *At*, in the *iDCA*, every *mcav* calculation is used to update a normal distribution, where the *CDF* is compared with the anomaly threshold *At*.

In summary, in a typical *dDCA* batch model, the *Pre-processing & Initialization* stage focuses on constructing a list of all *Ag*s and signal inputs based on the data in the dataset. The *Detec-*

tion and *Context Assessment* steps will then use all of the input data to 'train' the **dDCA**, i.e., to evaluate the sample contexts of *Dc*. Finally, in the *Classification* step, all of the sample contexts will be utilized to categorize the *Ag*s as normal or anomalous, with a context of 1 indicating an anomalous *Ag* classification and a context of 0 indicating a normal *Ag* classification. On the other hand, with the **iDCA**, each time a new *Ag* becomes available, all four phases are processed as a unique execution block. This allows for a classification result far faster than needing to analyze all sampled *Ag*s in a dataset.

6.3.2.1 Data Processing

In the **dDCA**, the input of data for classification is in a batch format. Typically, assuming the usage of a dataset, all data entries are first converted to *Ag*s. All *Ag*s must be processed to identify which *Dc*s in the population migrate and in what state (mature or semi-mature). The classification of each *Ag* occurs only after the entire set of migrated *Dc*s is derived. So, to classify, e.g., initial entries in the dataset, the **dDCA** processes the entire dataset first for the migration process (during the *Context Assessment* phase), and then the migrated *Dc*s are used to classify each *Ag*.

On the contrary, in the **iDCA**, the data is provided in a streaming format rather than a batch format during the *Pre-processing & Initialization* stage. In this case, the *River Python* library [300], a package for online ML⁴³, is used for incremental data pre-processing. Every time a new data entry is available, the **iDCA** converts it to an *Ag* format and derives signals based on the information available at that moment, i.e., the sampled *Ag* and all the history of samples before. So, because the migration process depends on the only available *Ag* at the moment and historical data, the classification of this *Ag* considers the information available in the subset of *Dc*s migrated until that stage only.

Thus, at the end of the *Context Assessment*, it is expected to have less migrated *Dc*s in comparison with the **dDCA**. Also, in the *Classification* stage, there is only one *Ag* to be considered for classification. In consequence, it is expected the **iDCA** to be much faster to process and classify *Ag*s, since, during the *Dc* migration and *Ag* classification, the number of agents involved in the processes is pretty smaller when compared to the **dDCA**. However, since the *Dc* migration process is based on less information, the process may lack a complete statistical overview of the dataset, resulting in less effective classification.

6.3.2.2 Feature Selection

Feature selection is performed before the *Pre-processing & Initialization*. Feature selection corresponds to selecting the attributes that can make the predicted variable more accurate or eliminate irrelevant features that can decrease the model accuracy and quality. In this case, both **dDCA** and **iDCA** share a similar process to select features. Consulting state of the art, this is done manually, considering a human expert on the problem context, or use correlation methods to identify the features with more useful information, as proposed by Gu [256]. Gu shows that in several test cases,

⁴³Machine Learning

the feature correlation results in better classification performance when compared to the manual process. The *Pearson correlation* [303] is used by calculating the correlation coefficients and mutual information between all the features and the dependent variable, i.e., the label. This way, it is possible to get insights into the dependency of the correct classification with different features or even identify the causal relationship between features and the class label (true classification).

The *Pearson correlation* results range from -1 to 1. A positive cross-correlation means that a given feature and the class label are linear. In contrast, a negative cross-correlation means an inverse linear relationship between the feature and the class label. On the other hand, a null correlation means that both features are not correlated. Thus, for the *iDCA* and *dDCA* in the context of this study, the absolute value of the *Pearson correlation* between all (numerical and continuous) features and the class label is calculated, and only the highest correlated features are selected. A perfect (positive or negative) correlation equals 1 or -1.

6.3.2.3 Input Signal Mapping & Data Processing

In the *Pre-processing & Initialization* stage, it is assumed that the features to be used in that specific problem are already selected. After deriving conclusions from empirical results, different authors use multiple approaches for signal mapping, according to the features available and what works best in a specific problem context. In this study, we consider the original proposal of the *DCA* [265], and Gu's proposal to be used in the *dDCA* [256] for the initial classification of input signals into various categories, namely *PAMP*, *Danger* and *Safe* signals.

On the one hand, Greensmith proposal (we will refer to it as *One-Feature Approach*) considers one feature to form both *PAMP* and *Ss* signals, which are the most representative of anomalous and safe situations, respectively. Based on a predefined threshold, feature values greater than the threshold are classed as *Safe* signals, while values below the threshold would be used as *PAMP* signals. The procedure is:

1. Select a suitable feature;
2. Calculate the mean of all the selected attribute's values;
3. For each attribute value, determine if it is a *PAMP* or *Safe* signal;
4. If *PAMP*, the signal is attributed with the absolute difference between the original feature value and the calculated mean, while *Ss* is attributed with 0. On the other hand, if *Safe* signal, the procedure is the other way around, i.e., *Ss* is equal to the absolute difference between the value and mean, while *PAMP* is equal to 0.
5. Finally, to derive the *Danger Signal*, the rest of the features are used, by calculating and relating mean values of each attribute across the values of class 1 (anomaly) only, without including class 2 (normal) values.

On the other hand, Gu proposal regarding feature reduction (we will refer to it as *Two-Feature Approach*) considers one feature for each signal *PAMP*, *Ds*, and *Ss*. He assumes that *PAMP*

and Ds signals correlate with anomalous situations, while the magnitude of Ss is correlated with normal situations (inverse of anomalous situations). Thus, the feature that primarily represents anomalous situations can be mapped to the $PAMP$ signal. In contrast, the second-best feature that represents anomaly should be mapped to the Ds . Finally, the best feature representing normal situations should be mapped to Ss . This means that, using a correlation coefficient approach to cross-correlate all features with the class label:

1. The feature with the highest positive correlation should correspond to the $PAMP$;
2. Ds should correspond to the second highest positive correlation;
3. The high negative correlation can form the Ss signal.

Note that similar approaches may use a subset of highly correlated features for each signal category instead of only one feature per signal. Also, other work follows the same strategy but does not consider the correlation calculation. Instead, they use intuition (based on expert knowledge) about similarity and feature correlation with the class label.

In the $iDCA$, we assume that two signals (Ds and Ss) are enough to represent all the anomalous and safe situations. Thus, we assume the same procedures described before, but to form both Ds and Ss while not considering the $PAMP$. Suppose Greensmith's proposal is considered, i.e., a *One-Feature Approach*. In that case, we replace $PAMP$ with Ds when mapping with the only feature selected and do not consider the remaining features of the dataset. This presents advantages regarding the speed of the *Pre-processing & Initialization* stage. However, relying on one unique feature may not hold all the information needed to represent all the anomalous and safe situations adequately. On the other hand, using a *Two-Feature Approach* may enable more information. Again, we are using only Ds and Ss , i.e., we only consider the two highest correlated features. However, it is still unclear if a simple assessment of the correlation is enough to encode the characteristics of each signal category. This approach is not guaranteed to be scalable to a real-world detection scenario.

6.3.2.4 Algorithm Variables

Despite eliminating most of the random-based elements, there is still the need to specify some input parameters in the $dDCA$. Authors in previous work propose some guidelines, but it seems that similar to the feature selection and input signal mapping, the specification of these parameters is dependent on the problem context. These variables are:

- Size of Dc population. The most significant variation is found between 1 and 100 cells based on past sensitivity analyses of the cell numbers found in the literature.
- Number of iterations. The number of iterations is a parameter introduced in the $iDCA$, but it can also be used in the $dDCA$. In the $dDCA$, while data is available, a Dc samples Ags or input signals. This will be later used to update the Dc profile and calculate output

signals to assess the migration of the *Dcs*. Thus, in the *dDCA*, the notion of iterations is the process referred to as antigen multiplier work-around, i.e., clone and multiply new data entries every time they are available. Without the work-around, this occurs once, but we can force to repeat sampling of the same *Ag*s multiple times. Also, the sampling process target a random *Dc*, and it can be an *Ag* or signal input.

In the *iDCA*, every *Dc* in the population samples all *Ag*s and signal available in each iteration. Imagine that the *#Iter* is 10. In the *iDCA*, this means that each *Ag* will be sampled by all *Dcs* during 10 iterations. The *Ag* and input signals temporal correlation is performed before, i.e., when sampled, each *Ag* already has a value of *Ds* and *Ss* associated. Thus, in the *iDCA*, the number of iterations is a way to expose multiple times the same *Ag* to all *Dc* population and increase the speed of migration. If we choose to replicate the same behavior in the *dDCA*, then all the *Ag*s must be sampled by all *Dcs*. In contrast, the output signal derivation and *Dc* profile update are repeated during the number of iterations specified.

- *Dc* Migration threshold. Each cell in the population has a lifespan, which defines the number of input signals and *Ag*s the cell can process before migrating. Each cell is assigned a different migration threshold to ensure that the *Dc* population observes the data (*Ag*s and input signals) over different time scales. This threshold can be random or generated from a uniform distribution. Thus, each *Dc* may be seen as a moving window when sampling *Ag*s and signals. According to Gu [256], the factor of generating migration thresholds may be set to the max value of *csm* or 100 times the max value of *csm*.
- Anomaly threshold. The anomaly threshold *At* is used for the binary classification of each *Ag*, i.e., to understand if the *Ag* is normal or anomalous. After calculating the *mcav* (ratio of mature *Ag*s with all *Ag*s presented), if the *mcav* is higher than the *At*, the *Ag* is anomalous. On the other hand, if lower, the *Ag* is treated as normal. Thus, and assuming the *mcav* ranges between 0 and 1, the default value of the *At* is 0.5. However, it can be derived by considering the original proposal of the *DCA* [265]. Greensmith suggested that the *At* could be derived based on the distribution between the two classes (normal and anomaly) in the dataset. Thus, the *At* can be calculated by finding the ratio between anomalous data entries and the total number of data items.

Regarding the derivation of the *mcav*, the *mcav* calculation procedure is the same in the *Classification* step for both *dDCA* and *iDCA*. The difference is that, in the *iDCA*, only one sampled *Ag* is examined at a time. This means that the information represented by the *mcav* is short since only one *Ag* is considered. Thus, and assuming the classification must consider not only the current *Ag* but the entire history of sampled *Ag*s, the *iDCA* uses every calculated *mcav* to update a normal distribution. In this case, the *CDF* of the calculated *mcav* is compared with the anomaly threshold *At*, instead of the *mcav* value. Thus, the returned *CDF* is the probability that the variable takes a value less than or equal to *mcav*. This value is then compared with the *At*.

6.3.2.5 Runtime Properties

In both the **dDCA** and **iDCA**, there are runtime variables that can be used as quantitative indicators of the behavior of the algorithm. Using different values for the input parameters described previously may influence these runtime variables and, consequently, the algorithm behavior. Thus, the runtime variables can be good indicators of whether the current setup of the algorithms is suitable or should be updated given a specific problem context. They are:

- Number of migrated **Dcs** and processed **Ags**. Originally, the number of migrated **Dcs**, or the M^{44} , indicates the amount of processed information (based on the input signals). In contrast, the number of processed **Ags** indicates the amount of processed information (based on the sampled **Ag**). Increasing the size of the **Dc** population will guarantee good coverage in using sampled signals for further migration. Each **Dc** in the population can sample and correlate both **Ag** and input signals. This way, the **Dc** can obtain sufficient information to migrate and further present the **Ag** collected. The problem is that considering that an **Ag** corresponds to a data entry, there will be only one **Ag** per **Ag** type. In consequence, sampling **Ags** may not be effective due to the absence of sufficient *Antigens*. The **Ag** multiplier workaround is an approach to mitigate this limitation. Each input **Ag** instance is multiplied by a factor, resulting in having multiple **Ags** that are the same and belong to a specific **Ag** type. In the *Classification* stage, the *mcav* is calculated considering **Ag** types, where each type corresponds to a group of cloned **Ags**.

In the **iDCA**, instead of the **Ag** multiplier approach, we use the number of iterations input parameter (*#Iter*) to define how many times a specific **Ag** is sampled. This way, instead of increasing the number of **Ags** to be processed, we increase the number of **Dcs** migrations and, consequently, the rate of reincarnations, i.e., migrated **Dc** to be reinstated in the population as immature. These **Dcs** are reinstated with a different migration threshold than the original **Dcs**, thus contributing to a wider variety of **Dcs** in the population without increasing the number of **Dcs**. Thus, we can enable continuous detection and online adaptation to the changes encountered while different **Ags** arrive with this approach.

- Duration of both migration and classification. When processing large-size datasets, the duration of the **Dc** migration and **Ag** classification processes greatly influence the algorithm's time to process and classify data entries. In the **dDCA**, despite the number of **Ags**, i.e., the size of the dataset, the size of the **Dc** population, and eventually the number of iterations also influence directly this duration. An increased number of **Dcs**, together with the number of **Ags**, will contribute to an increased migration of **Dcs**. Thus the migration process takes longer. Moreover, because there is an increased number of migrated **Dcs**, the classification process will also present a long runtime duration.

In the **iDCA**, there is a similar dependency between the size of the **Dc** population and the *#Iter*. They directly influence the duration of both migration and classification procedures.

⁴⁴Number of Cell Migrations

However, because we do not have a group of *Ag*s to be processed, the algorithm execution speed is substantially faster.

- Detection duration. The detection duration and the $R_{\Delta D}$ ⁴⁵ are related to the duration of both migration and classification. In the *dDCA*, because an *Ag* is only classified after all dataset is processed, the duration of detection naturally increases. In the *iDCA*, since both these procedures are performed sequentially every time an *Ag* is sampled, the detection duration decreased significantly and, in consequence, the $R_{\Delta D}$ is much higher.
- $R_{\emptyset}$ ⁴⁶. Non-classifications refer to data entries that are not further classified. This can occur when the amount of processed information sampled by the population of *Dc* is not enough. During the *Classification* stage, the *mcav* is calculated by relating the number of sampled *Ag*s in migrated *Dcs* in the mature state. Thus, if no mature *Dc* has migrated at that stage, the *mcav* cannot be derived, and the data entry is not classified. Also, even if there are mature *Dcs*, they may have failed to sample the *Ag* corresponding to the data entry to be classified. This means the *mcav* cannot be derived, and no classification will become available.

6.4 Security by Design

This section focuses on the creation and deployment of the *A-HIDS* in the Edge layer of a *CPPS* based on the *iDCA* approach. This *IDS* examines *OPC UA* network data to identify cyber threats that use this communication protocol. In this case, specific *FBs*⁴⁷ conforming with the *IEC* 61499 standard are developed utilizing the *DINASORE* open-source technology. These *FBs* may be utilized in many application contexts for embedded intrusion detection in Edge devices. In this sense, this study engages in the development and deployment of the *iDCA* approach in a real-world Edge layer of an *IIoT* application, taking into account the *IEC* 61499 standard. The authors investigate a new technique to allow data-driven anomaly detection by design, according to the *MBE*⁴⁸ approach, for system design and implementation by developing the *iDCA* in an *FB* compatible format with the *IEC* 61499 standard. The direct result is the *A-HIDS* Pipeline, which is shown in Figure. 6.2.

There are two primary *FBs* based on this architecture: *OPCUA_Flow_Sniffer* and the *iDCA*. The first *FB* corresponds to the *Network Traffic Capture* stage (see Figure 5.1 in Section 5.5), and it is used to scan the network for *OPC UA* packets. The second *FB*, on the other hand, corresponds to *Pre-processing & Initialization*, *Detection*, *Context Assessment* and *Classification* activities. This *FB* is responsible for the data analysis and intrusion detection using the *iDCA* approach.

Building *FBs* using *DINASORE* requires *Python* 3.6 or above, as well as numerous *Python* packages, including *psutil*, *cryptography*, *opcua*, *numpy*, *scikit-learn* and *pandas*. Furthermore, in order to work particularly with the *OPCUA_Flow_Sniffer* and the *iDCA* *FB*, there

⁴⁵ Detection rate

⁴⁶ Classification rate

⁴⁷ Function Blocks

⁴⁸ Model-based Engineering

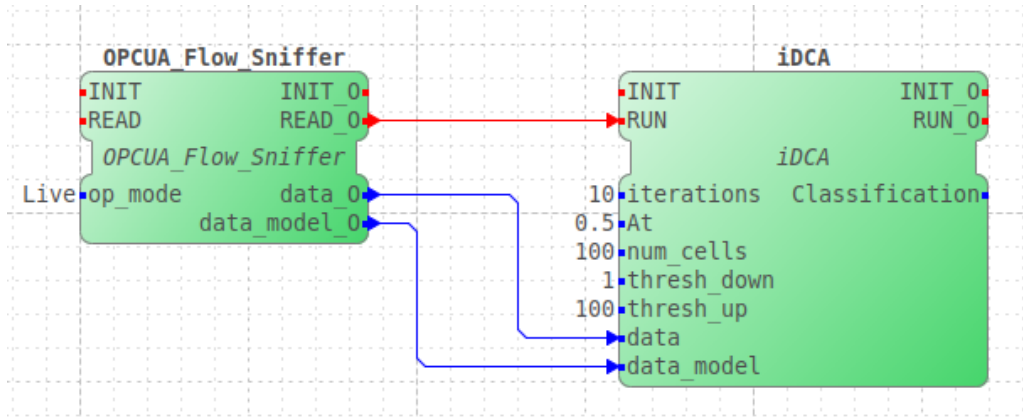


Figure 6.2: Anomaly Host Intrusion Detection System Pipeline.

are additional *Python* requirements, in addition to the ones already stated, namely `pyshark` [296] and `river` [300]. `pyshark` is a *Python* wrapper for the *Tshark* library, which is used for network package sniffing in the *OPCUA_Flow_Sniffer* FB. `river`, on the other hand, is a *Python* package for ML applications in streaming data or online ML. This is utilized for incremental analysis and real-time intrusion detection in the *iDCA* FB. The ECC⁴⁹ of the A-HIDS Pipeline is depicted in Figure 6.3.

6.4.1 OPCUA Flow Sniffer

The *OPCUA_Flow_Sniffer* FB can operate in three different modes, which can be defined in the FB's `op_mode` input:

- **Live:** The live packet capture mode relates to the gathering of OPC UA packets from the Edge device in real-time. From a live network interface, all OPC UA traffic is sniffed using the `pyshark` package.
- **File:** The file packet capture mode relates to the gathering of local OPC UA packets from a PCAP file holding historical OPC UA communication from an Edge device. As a result, all OPC UA traffic is read from the file using `pyshark` again.
- **The dataset mode** is a backdoor mode in which the user may test the implementation using an existing dataset in CSV file format that comprises OPC UA traffic data and network characteristics that have been configured. The FB was optimized in this mode, in terms of considered network features, to accept as input the *M2M using OPC UA* dataset [34] (see Section 6.2.1).

The *OPCUA_Flow_Sniffer* FB has two data outputs: `data_0` and `data_model_0`. Both of these outputs correspond to data inputs in the *iDCA* FB, which are related with the *RUN* input event. This input event triggers the *iDCA* FB execution. The `data_0` represents the OPC UA

⁴⁹Execution Control Chart

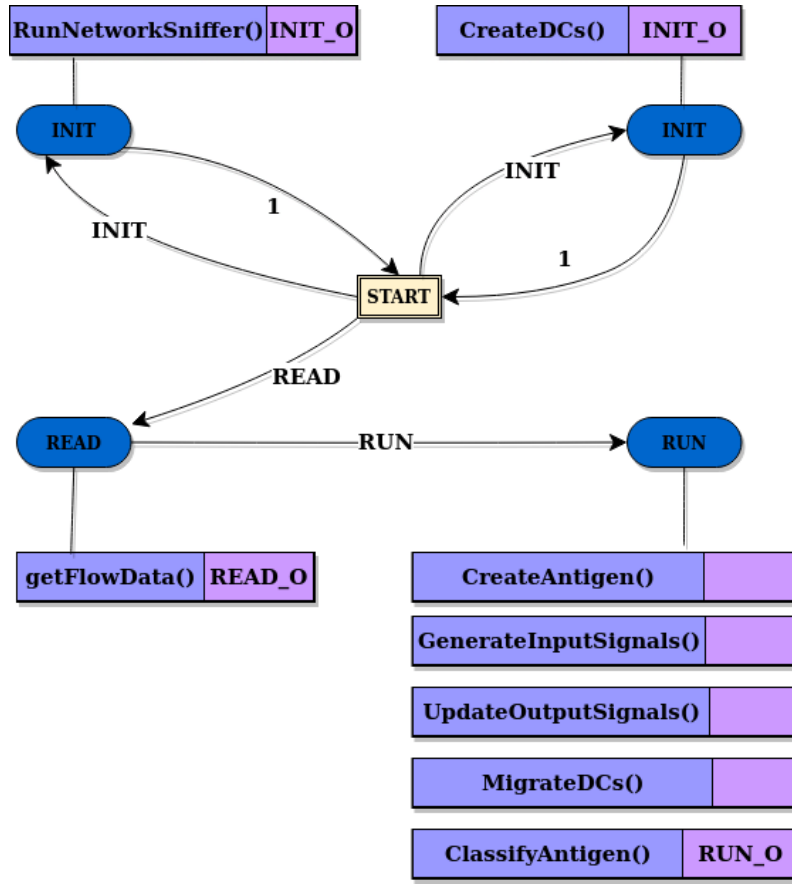


Figure 6.3: Execution Control Chart of the Anomaly Host Intrusion Detection System Pipeline.

bidirectional flow that will be analyzed by the **iDCA** approach. The **OPCUA_Flow_Sniffer FB** sends **OPC UA** flows to the **iDCA** as soon as they are available in a sequential format. The **data_model_O**, on the other hand, corresponds to the network features extracted from the network data inputs, which as default, correspond to the *M2M Using OPC UA* features (see Section 6.2.1).

6.4.2 iDCA

Regarding the **iDCA FB**, besides the dynamic data input from the **OPCUA_Flow_Sniffer FB**, such as the **data** and the **data_model**, there are other important data inputs, related to the **iDCA** initial parametrization. In this case they are:

- **iterations**: Equivalent to the **#Iter** parameter of the **iDCA**.
- **At**: **At** used in the *Classification* phase of the **iDCA**.
- **num_cells**: Size of **Dc** population.
- **thresh_up**: Higher threshold value used to generate **Dc** lifespan (expected time to live of the **Dc**). This is equivalent to t_m^{50} .

⁵⁰Migration Threshold

- `thresh_down`: Lower threshold value used to generate *Dc* lifespan. This value is 0 as default.

Intrusion detection is achievable thanks to the *iDCA* technique. Because the input data (network flows) to be analyzed become available progressively over time, the *iDCA* may analyze incrementally across the data (stream data). Finally, as data output, the *iDCA* **FB** makes the classification result of each **OPC UA** flow (received in the `data` input) available in the `classification` output. Suppose `classification` is zero in this situation. In that case, the **OPC UA** flow being examined is a normal network instance. Otherwise, a value of 1 indicates that the **OPC UA** flow is an anomalous network instance and probably an assault on the network's Edge device is active. Finally, if `classification` has a value other than 0 or 1, then it signifies that the *iDCA* was unable to classify the **OPC UA** flow owing to inadequate information.

6.5 Conclusion

This chapter is organized into three main topics, namely: the **OPC UA** data collection, the *iDCA* as detection approach, and the **A-HIDS** Pipeline, compliant with the **IEC** 61499 standard. First, we describe a new dataset, designated *M2M using OPC UA*, which can be used to assess **IDS** techniques' classification performance. This work was motivated by the need to have publicly available a dataset containing industrial network data. On the one hand, the dataset presents network flows collected in a laboratory **CPPS** testbed while different network nodes (Edge devices) were exchanging data using **OPC UA**. On the other hand, several attacks were injected into the network by exploring **OPC UA** vulnerabilities, targeting one or more network nodes.

Second, we describe in detail a new **DCA**-based approach, designated *iDCA*. This algorithm tries to embed incremental capabilities and adapt the original **dDCA** to enable online classification, which is a usual scenario in intrusion detection problems. The main advantage of the algorithm is the low detection duration of intrusions since all algorithms stages are executed in sequence every time a new **Ag** is available. A binary classification output is ready moments after the **Ag** is available as algorithm input. This is not the case with the **dDCA**, where the algorithm processes the entire dataset before starting the classification stage. Since no complete statistical analysis of the dataset is available upfront, the most significant limitation of the *iDCA* is the increased difficulty to find out what is the expected behavior considered to be the baseline and how to model that behavior as *Danger* vs. *Safe* signals.

There are also other limitations, which are shared with the **dDCA**. Since the technique is highly dependent on the problem context for initial parametrization, the validation of the *iDCA* should follow a comprehensive empirical analysis by combining multiple parameters in different contexts to understand what works best or not. More detail about this analysis can be found in Chapter 7. There are some guidelines regarding parameter specification in the literature, but it is not clear if they work in the *iDCA* since the algorithm functionality is different. Moreover, at this stage, it is not clear if the **dDCA** itself is appropriate for different network intrusion detection problems, especially considering the industrial domain.

Third, we investigate the security of devices in the CPPS Edge layer by providing an A-HIDS Pipeline that analyzes OPC UA network data in typical M2M communication scenarios. Also covered is the problem of providing security by design, i.e., introducing intrusion detection capabilities during the CPPS design process. Enabling CPPS Edge layer security by design is accomplished by the IEC 61499 standard, used to build FBs particularly for implementing the A-HIDS functionality in Edge devices. The DINASORE technology is used to generate and deploy the FB Pipeline.

It is feasible to infer that the proposed approach offers several advantages. To begin, it proposes a feasible IDS architecture for embedded Edge devices that takes into account detection characteristics from ICT⁵¹ as well as the particular aspects of the industrial domain. Second, the A-HIDS technique provides a level of protection that traditional ICT security solutions may not achieve. Third, by applying complex analysis using bio-inspired methodologies and coping with the system's dynamic behaviors throughout time, this strategy allows a comprehensive anomaly detection approach. Fourth, our technique can act on real-time data by gathering, integrating, evaluating, and identifying threats as they occur. Processing data streams is a common scenario in IIoT applications.

⁵¹Information and Communication Technology

Chapter 7

DCA-based IDS Evaluation

7.1 Introduction

In this chapter, the reader will learn about the primary experimental designs, testing methodologies, main results achieved and discussion. We performed two sets of experiments to validate different components of the proposed approach described in Chapter 6. They are:

E1. iDCA vs. dDCA

E2. A-HIDS Pipeline

In Section 7.2, the **E1. iDCA vs. dDCA** experiment is detailed. We validate both **dDCA**¹ and **iDCA**² performances as **IDSs**³. We use different datasets oriented for network intrusion detection. On the one hand, we intend to assess the quality of **dDCA** and **iDCA** when applied in network intrusion detection problems. On the other hand, we want to understand how the **iDCA** behaves when compared to the **dDCA** in terms of detection rate and accuracy. The goal is to find answers to the following research questions:

***RQ T.7** What impact in the Dendritic Cell Algorithm's detection rate and accuracy has an incremental learning approach?*

***RQ T.8** What is the cause-effect relationship between algorithm variables, runtime properties and detection performance in Dendritic Cell Algorithm-based intrusion detection?*

Considering the **E2. A-HIDS Pipeline** experiment, in Section 7.3, we intend to validate the **A-HIDS**⁴ Pipeline architecture to be deployed in the **CPPS**⁵ Edge layer. We use the **IEC**⁶ 61499

¹deterministic Dendritic Cell Algorithm

²Incremental Dendritic Cell Algorithm

³Intrusion Detection Systems

⁴Anomaly Host Intrusion Detection System

⁵Cyber-Physical Production System

⁶International Electrotechnical Commission

standard as a [MBE⁷](#) approach for [CPPS](#) design. We show the feasibility of enabling by design an [IDS](#) to protect the Edge layer of the [CPPS](#). In this case, the detection approach is based on the [iDCA](#). Thus, we attempt to answer the research question:

RQ T.9 *How effective is IEC 61499 in supporting the implementation of low complexity and high maintainability security by design Cyber-Physical Production Systems?*

Thus, we describe the experimental design, testing methodology, and experimental setup for each experiment. Also, we present in detail the achieved results and provide discussions about the findings.

Section [7.2.1](#) focuses on the experimental design regarding the [dDCA](#) and [iDCA](#) validation. We describe the validation methodology in terms of algorithm parametrization and assessment metrics to be used. Section [7.2.2](#) focuses on the algorithms' pre-processing approach, i.e., the procedures for feature selection and formation of input signals *Safe* and *Danger*. Then, Section [7.2.3](#) describes the experimental methodology by identifying multiple test case scenarios to run the experiments using datasets.

Finally, Section [7.2.4](#) reports results and describes in detail the main findings about the assessment of the [dDCA](#) and [iDCA](#) and detection techniques. This section is organized into four subsections, each reporting findings of the experiments in all the datasets considered, notably the proposed *M2M Using OPC UA* [\[34\]](#), *KDD Cup99* [8](#), *NSL-KDD* [9](#) and *UNSW-NB15* [10](#) datasets (more detail in Sections [6.2](#) and [2.4.5](#)).

On the other hand, Section [7.3.1](#) focuses on the *Experimental Design: A-HIDS Pipeline* regarding the [A-HIDS](#) Pipeline validation. We start by identifying the main testing needs of a [HIDS¹¹](#). We use these needs to design a methodology to validate the suggested approach's appropriateness in resource-constrained Edge devices. Section [7.3.2](#) describes the testing methodology and main performance metrics used. Section [7.3.3](#) presents the experimental setup, which consists in a lab [CPPS](#) testbed, inspired in the same setup used to generate the *M2M using OPC UA* dataset. Then, Section [7.3.4](#) reports results about the deployment of the [A-HIDS](#) Pipeline in the [CPPS](#) testbed. These results are reported in two different subsections. The first tackles the performance overhead of the [HIDS](#) when implemented in the industrial Edge layer. The second focuses on the design complexity and maintainability criteria to assess the [FB¹²](#) Pipeline and overall [MBE](#) approach.

To conclude the chapter, Section [7.4](#) states final remarks about the work presented and resumes the study findings used to provide answers to the research questions mentioned before.

Partial results of the presented work in this chapter have been submitted at:

⁷Model-based Engineering

⁸<https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

⁹<https://www.unb.ca/cic/datasets/nsl.html>

¹⁰<https://research.unsw.edu.au/projects/unswnb15-dataset>

¹¹Host-based Intrusion Detection Systems

¹²Function Block

- Pinto, R. & Gonçalves, G. (2022). Immune-based Industrial Network Intrusion Detection System. IEEE Access, (submitted).

7.2 E1. iDCA vs. dDCA

7.2.1 Experimental Design

Considering the problem context dependability, the application of the dDCA and iDCA imply the algorithm's fine-tuning, according to empirical results. By tuning the algorithm, we can play with different input parameters such as the $\#Dc^{13}$, $\#Iter^{14}$, t_m^{15} and At^{16} to test the quality of the algorithms empirically. Also, the procedures for data pre-processing are different since the algorithm depends greatly on the features used and the methodology to form each input signal. Thus, the performed experiments consider both dDCA and iDCA with different combinations of algorithm input parameters and pre-processing procedures. They are as follows:

1. Feature selection: We follow the cross-correlation procedure between features and the class label, as suggested by Gu *et al.* [278]. Only highly correlated features will be considered. As mentioned before, there are two approaches for signal mapping, i.e., the *One-Feature Approach* to form both signal categories or use the *Two-Feature Approach*, one for each signal category. The features selected are dependent on each dataset. More detail is provided in the sections reserved for each specific dataset.
2. Dc^{17} lifespan: The lifespan of a given Dc should be randomly generated within a range of values, dependent on the maximum value of the interval, i.e., the t_m . In this case, and following orientations from previous work, the maximum value of the interval can be equal to the maximum value of csm^{18} , and it can go as far as 100 times more. csm can be calculated using Eq. (3.3), so the max value of csm corresponds to the sum of the maximum values of both Ds^{19} and Ss^{20} . Since all values used to derive the input signals are normalized between 0-100%, and, depending on the feature selection and signal mapping procedure, the maximum csm can range from 100 to 10 000.
3. Number of iterations: We assume that the algorithm may not use iterations, i.e., $\#Iter = 1$, or there are iterations during the migration process. In this case, we are assuming $\#Iter = 10$.
4. Population size: The most popular choice of $\#Dc$ is 100. We also want to assess how the algorithm behaves with a low Dc variety ($\#Dc = 1$).

¹³Number of Dendritic Cells

¹⁴Number of Iterations

¹⁵Migration Threshold

¹⁶Anomaly Threshold

¹⁷Dendritic Cell

¹⁸Co-stimulatory Signal

¹⁹Danger Signal

²⁰Safe Signal

5. Anomaly threshold: The At can assume two values. The default is 0.5, assuming a balanced 50/50 dataset. The option is the derived version, considering the abovementioned guidelines for its calculus, i.e., the ratio between the number of anomalous data entries and the total number of data items.

Regarding metrics for performance evaluation, as one of the most commonly used approaches in intrusion detection, the ROC^{21} analysis and $AUC-ROC^{22}$ measure, which are performed to assess the classification performance of the algorithms. Also, we count the TP^{23} , FP^{24} , TN^{25} and FN^{26} to generate the TPR^{27} , TNR^{28} , and detection Acc^{29} . These are calculated at the end of the classification of the entire dataset. Moreover, we assess important non-functional requirements regarding the classification, which have a direct impact on the suitability of the algorithm as a detection approach in an industrial IDS. The non-functional metrics considered are the $R_{\emptyset}^{30}$ and the $R_{\Delta D}^{31}$.

The $R_{\emptyset}$ takes into consideration the number of non-classifications, which refer to data entries that the algorithm was not capable of classifying. In algorithmic terms, this means that, during the *Classification* stage, when calculating the $mcav^{32}$ for a given Ag^{33} , migrated Dcs may not be available at that time. Also, even if migrated Dcs are available, the Ag to be classified may not have been sampled by those Dcs at that stage. This may occur if the Dc migration process is not fine-tuned accordingly to the problem context. On the other hand, the $R_{\Delta D}$ is the metric used to assess the algorithm's time to classify a data entry after being available as input. For this, we calculate the detection duration of the classification output of the first TP available in the dataset, along with the total time of the migration process and the classification of the entire dataset.

7.2.2 Pre-Processing

A correlation-based method is used to assess the highest correlations between a given feature and the class label (see more in Section 6.3.2.2). Based on the cross-correlation results, Table 7.1 represents the relationship between selected features and signal categories used to perform the experiments in each dataset. For the curious reader, Appendix F contains the cross-correlation results, presenting for each dataset the highly correlated features. We assume that an absolute correlation higher than 0.5 can be considered a high correlation. The best "low" correlated feature may be considered if at least two high features are not found.

²¹Receiver Operating Characteristic

²²Area Under the Receiver Operating Characteristic Curve

²³True Positive

²⁴False Positive

²⁵True Negative

²⁶False Negative

²⁷True Positive Rate

²⁸True Negative Rate

²⁹Accuracy

³⁰Classification rate

³¹Detection rate

³²Mature Context Antigen Value

³³Antigen

Table 7.1: Relationship between selected features and signal categories.

Dataset	One-Feature Approach	Two-Feature Approach
<i>M2M Using OPC UA</i>	<i>b_pktTotalCount</i>	<i>Ss</i> $\Rightarrow$ <i>b_pktTotalCount</i> <i>Ds</i> $\Rightarrow$ <i>count</i>
<i>KDD Cup99</i>	<i>logged_in</i>	<i>Ss</i> $\Rightarrow$ <i>logged_in</i> <i>Ds</i> $\Rightarrow$ <i>count</i>
<i>NSL KDD</i>	<i>same_srv_rate</i>	<i>Ss</i> $\Rightarrow$ <i>dst_host_srv_count</i> <i>Ds</i> $\Rightarrow$ <i>same_srv_rate</i>
<i>UNSW</i>	<i>sttl</i>	<i>Ss</i> $\Rightarrow$ <i>swin</i> <i>Ds</i> $\Rightarrow$ <i>sttl</i>

We use both training and testing sets (when available and with class labels) to validate the algorithms. From the datasets considered, only the *M2M Using OPC UA* does not present a testing set version. Also, despite having one, the *KDD Cup99* testing set is not labeled. We split the dataset into two sets for training and testing purposes in these situations. The testing set corresponds to 30% of the entire dataset and contains random data entries while maintaining the same class distribution between normal and anomaly. The features presented in Table 7.1 were derived based on only the training set and are used for both training and testing set validations. Finally, as mentioned before, all features are normalized between 0-100% before being used for input signal calculation. Detail about the signal calculation strategy for both feature selection approaches can be found in Section 6.3.2.3.

7.2.3 Experimental Setup

Both dDCA and iDCA, including the *Network Traffic Capture*, incremental approaches, and *Pearson Correlation* for feature selection, were developed using *Python* programming language, version 3.7. All experiments were performed on a *Ubuntu 18.04 LTS* (64-bit)-based machine, with 16Gb RAM³⁴, 700Gb of disk space, and an *Intel Core I7 CPU*³⁵ 8x @ 2.8GHz. Regarding experimental methodology, Table 7.2 presents 16 test case scenarios, considering the combination of multiple algorithm parameters. A subset of these was used in different experiments, i.e., the validation of dDCA and iDCA in each dataset considered.

7.2.4 Methodology

For the validation of the dDCA and iDCA in each dataset, we select a subset of test case scenarios to be used in each algorithm and dataset. The results presented include the *AUC-ROC* and detection *Acc*, the *TPR*, the *TNR*, $\emptyset$ ³⁶, Δ ^{D37} and the *T*³⁸. *AUC-ROC* and *Acc* ranges between

³⁴Random Access Memory

³⁵Central Processing Unit

³⁶Number of Non-classifications

³⁷Intrusion Detection Duration

³⁸Total Classification Time

Table 7.2: Experimental methodology: test case scenarios.

Test Case	Pre-processing	t_m	#Iter	#Dc	At
TC01	One-Feature Approach	100	10	100	Default
TC02	One-Feature Approach	100	10	100	Derived
TC03	One-Feature Approach	10 000	1	1	Default
TC04	One-Feature Approach	10 000	1	1	Derived
TC05	Two-Feature Approach	100	10	100	Default
TC06	Two-Feature Approach	100	10	100	Derived
TC07	Two-Feature Approach	10 000	1	1	Default
TC08	Two-Feature Approach	10 000	1	1	Derived
TC09	One-Feature Approach	100	1	1	Default
TC10	One-Feature Approach	100	1	1	Derived
TC11	One-Feature Approach	10 000	10	100	Default
TC12	One-Feature Approach	10 000	10	100	Derived
TC13	Two-Feature Approach	100	1	1	Default
TC14	Two-Feature Approach	100	1	1	Derived
TC15	Two-Feature Approach	10 000	10	100	Default
TC16	Two-Feature Approach	10 000	10	100	Derived

0 and 100%. TPR is the metric to measure how often does the algorithm predicts the positive class correctly. TNR measures how often does the algorithm predicts the negative class correctly. They both range between 0 and 100%. On the other hand, $\varnothing$ is a non-negative integer, ranging between 0 and the dataset size, and represents the number of non-classified data entries. Δt^D and T represent a duration in seconds. Finally, we also include M^{39} to assess runtime properties.

In the end, we select the best test cases in each dataset and directly compare the performance of different test cases. To identify best test cases, we assume three main goals: i) *Area Under the Receiver Operating Characteristic Curve*; ii) *Detection rate*, measured by relating Δt^D and T ; iii) *Classification rate*, measured by relating $\varnothing$ and size of dataset. A weighted mean is used to help decide what test case is best, considering that some goals are more important than others. Thus, the metric to assess the P^{40} is represented in Eq. (7.1), and the following rating system is used:

- **AUC-ROC**: The **AUC-ROC** is the most expressive component for validating the algorithms. We use 60% for the W_{AUC}^{41} .
- **$R_{\Delta t^D}$** : The rate of detection, defined in Eq. (7.2), relates the duration of the first true positive classification and the total duration to process the dataset. We use 15% for the $W_{\Delta t^D}^{42}$. The detection rate represents the classification speed, so a high detection rate corresponds to a fast classification speed.

³⁹Number of Cell Migrations

⁴⁰Global System Performance

⁴¹Area Under Curve Weight

⁴²Detection Rate Weight

- $R_{\emptyset}$: The rate of classifications, defined in Eq. (7.3), relates the number of non-classifications with the number of Ag . We use 25% for the $W_{\emptyset}$ ⁴³. The classification rate represents the successful classifications among all dataset entries, so a high classification rate corresponds to reduced non-classified entries.

$$P = W_{AUC} \times AUCROC + W_{\Delta^D} \times R_{\Delta^D} + W_{\emptyset} \times R_{\emptyset} \quad (7.1)$$

$$R_{\Delta^D} = 1 - \frac{\Delta^D}{T} \quad (7.2)$$

$$R_{\emptyset} = 1 - \frac{\emptyset}{Ag} \quad (7.3)$$

7.2.5 M2M Using OPC UA

Considering the *M2M Using OPC UA*, the $b_pktTotalCount$ and $count$ are the most correlated features, so they are used for signal mapping. $b_pktTotalCount$ is used for the *One-Feature Approach*. If the *Two-Feature Approach*, $b_pktTotalCount$ corresponds to the Ss while $count$ corresponds to Ds . The dataset contains 107 633 data entries, which is equivalent to the number of Ag created. Also, the derived At corresponds to the attack ratio. In this case it is 0.688.

7.2.5.1 dDCA Validation

Table 7.3 presents the validation results for dDCA in the *M2M Using OPC UA* dataset. We highlight the best P achieved using the dDCA. In this case it was TC14.

Table 7.3: *deterministic Dendritic Cell Algorithm* validation results in the *M2M Using OPC UA*.

Test Case	AUC-ROC	Acc	TPR	TNR	$\emptyset$	Δ^D	T	M
TC03	100%	99.97%	100%	99.92%	233	3.251	10.141	907
TC04	100%	99.99%	100%	99.96%	78	3.251	9.867	922
TC07	100%	100%	99.99%	100%	56	3.194	9.714	915
TC08	100%	100%	100%	100%	55	3.302	10.051	929
TC09	100%	99.99%	100%	99.97%	0	174.584	560.158	60 246
TC10	100%	100%	100%	99.99%	0	184.811	589.347	60 222
TC11	100%	99.95%	99.92%	100%	0	8 791	28 005	9 193
TC12	100%	99.88%	99.83%	100%	0	9 018	28 157	9 188
TC13	100%	100%	100%	100%	56	229.163	726.120	74 779
TC14	100%	100%	100%	100%	18	222.189	716.215	74 776

Results presented on Table 7.3 show that the dDCA is very accurate in detecting anomalies and distinguish them from normal instances. It is clear, in this dataset, that the dDCA is insensitive

⁴³Classification Rate Weight

to input parameter adjustment, because both *AUC-ROC* and *Acc* are (near) to perfect. Also, the *dDCA* does not share the typical limitation of *A-NIDS*⁴⁴ detection algorithms regarding high false positives. We can see the overall reduced number of *FP* and *FN* in all test cases. Usually, high false positives and bad detection accuracy may be explained due to the big variation of network traffic. The *M2M Using OPC UA* traffic patterns are stable when compared to other datasets that were created using traditional *ICT*⁴⁵ networks. Since all the traffic in the network is generated through an automated process, this stability means that the original packet transmission rate is stable over large periods. This can explain both the number of highly correlated features (see Appendix F) and the excellent detection *Acc* and *AUC-ROC*.

The detection performance is enhanced when considering the non-functional metrics. First, the $\emptyset$ is reduced, which means the *dDCA* can classify the majority of data entries. Regarding Δ^D and *T*, the *dDCA* presents a very satisfactory detection duration and processing time. Considering the dataset has 107 633 data entries, according to the detection duration/overall processing time, test cases can be classified as:

- Low duration (average of 3.25 seconds) considering TC03, TC04, TC07 and TC08.
- Normal duration (average of 3.38 minutes or 202.69 seconds) considering TC09, TC10, TC13 and TC14.
- High duration (average of 2.47 hours or 8 904 seconds) considering TC11 and TC12.

The test cases with short detection duration use as algorithm inputs only one iteration, the size of the cell population is only one *Dc*, and the lifespan is large (max. 10 000). Consequently, the *M* is rather low, i.e., the existing *Dc* migrates on average 919 times. On the other hand, the test cases with longer detection duration use ten times more iterations and 100 times more *Dc*, which results in more migrated cells (average of 9 191 times per cell) and, consequently, increases the detection rate. The normal detection duration test cases present a good balance between reduced *#Dc*, *#Iter*, and *t_m*. A reduced *t_m* means the *Dc* lifespan decreases, resulting in more migrations. This way, the algorithm maximizes the migrations (average of 65 506 times per cell) while maintaining an acceptable detection duration. Also, more migrations usually correspond to a higher classification rate since the probability of having mature *Dc* migrated and all the *Ag* sampled is higher.

Finally, based on the overall analysis of the *dDCA* results, we can identify the best test case results, assuming the three different goals mentioned before. If the *AUC-ROC* and *Acc* are the most important, all test cases present excellent results. Looking closely at the *TPR* and *TNR*, TC07, TC08, TC13 and TC14 are the best test cases. If the priority is in the detection duration, TC03, TC04, TC07 and TC08 are equivalent (TC07 shows slightly better results). Finally, if the number of non-classifications is an issue, TC09, TC10, TC11 and TC12 present no non-classifications. Considering Eq. (7.1), the global system performance in each test case is represented next.

There is not a big different between test cases, each one presents a good global result. The highest result achieved corresponds to TC14, where we use a *Two-Feature Approach* (*b_pktTotalCount*

⁴⁴Anomaly Network Intrusion Detection System

⁴⁵Information and Communication Technology

TC03: 95.137%	TC08: 95.058%	TC11: 95.291%	TC13: 95.253%
TC04: 95.039%	TC09: 95.325%	TC12: 95.196%	TC14: 95.342%
TC07: 93.255%	TC10: 95.296%		

as *Ss* and *count* as *Ds*), lifespan between 0-100, 10 iterations, 100 *Dc* in the population and the derived *At* (0.688).

7.2.5.2 iDCA Validation

Table 7.4 presents the validation results for iDCA in the *M2M Using OPC UA* dataset. We highlight the best *P* achieved using the iDCA. In this case they were TC01 and TC05.

Table 7.4: *Incremental Dendritic Cell Algorithm* validation results in *M2M Using OPC UA*.

Test Case	AUC-ROC	Acc	TPR	TNR	$\emptyset$	Δt^D	<i>T</i>	<i>M</i>
TC01	100%	99.69%	99.17%	71.87%	798	34.2	521.916	521 648
TC02	78%	69.21%	18.27%	89.12%	799	0.349	518.605	521 640
TC03	92%	91.82%	98.31%	79.92%	106 789	10.595	12.208	844
TC04	66%	67.78%	5.41%	95.59%	106 823	10.569	12.18	810
TC05	100%	99.99%	99.17%	74.85%	99	53.208	611.121	741 519
TC06	50%	31.14%	19.46%	90.95%	106	53.33	606.293	741 457
TC07	97%	97.01%	98.75%	99.22%	106 364	11.564	13.451	1 285
TC08	97%	96.5%	26.66%	99.93%	106 401	11.462	13.259	1 285
TC09	99%	99.15%	99.02%	75.62%	55 643	11.109	13.783	51 982
TC10	79%	77.54%	7.62%	94.71%	55 641	10.881	13.575	51 982
TC11	90%	87.72%	98.95%	75.41%	3 042	15.666	100.274	8 121
TC12	73%	63.55%	17.67%	91.96%	3 130	15.322	100.262	8 126
TC13	100%	100%	99.03%	99.07%	33 515	11.804	14.836	74 118
TC14	93%	92.71%	25.77%	99.31%	33 603	11.917	15.04	74 030
TC15	95%	94.87%	99%	97.67%	7 648	17.28	105.481	12 781
TC16	76%	68.22%	21.91%	98.5%	7 628	17.31	106.002	12 796

By inspecting the results, the iDCA is very sensitive regarding parameter's combination since performance results are very different in specific test cases. The first impression is regarding the $\emptyset$. There are several test cases, such as TC03, TC04, TC07 and TC08, where the iDCA dropped the majority of data entries, presenting an average non-classification of 106 594 in a total of 107 633. All these test cases share the same combination of $t_m = 10\,000$, $\#Dc$ and $\#Iter$ both equal to 1. We can conclude that, since the migration process is based on the sampling of only one *Ag*, a reduced number of $\#Dc$ and $\#Iter$ is not enough to have a significant number of migrations, leading to a higher number of non-classifications. In fact, test cases TC09, TC10, TC13 and TC14 share the same problem, since they present a non-acceptable number of non-classifications (average of 44 603 in a total of 107 633 data entries). In these test cases, despite reducing the t_m to foster cell

migration, the number of $\#Dc$ and $\#Iter$ is still low, and it does not enable satisfactory results in terms of classified data entries.

This means that the *iDCA* works best with a high cell population size and number of iterations, which correspond to test cases TC01, TC02, TC05, TC06, TC11, TC12, TC15 and TC16. When analysing both *AUC-ROC* and *Acc* of these test cases, we can conclude that only the ones using the default $At = 0.5$ present satisfactory detection performance. They are test cases TC01, TC05, TC11 and TC15 and present average *Acc* of 95.6%, on contrary to test cases TC02, TC06, TC12 and TC16, which present a most lower *Acc* (average of 58.03%). This means that the *iDCA* is sensitive to the At , which should be the default. This is an advantage because in online setups, we can't derive the At upfront, since we don't have access to the statistical properties of the entire dataset.

Despite the satisfactory detection performance of test cases TC01, TC05, TC11 and TC15, TC01 and TC05 stands out in comparison to TC11 and TC15. First, both the *AUC-ROC* and *Acc* are better (average of 99.84% vs. 91.3%). Second, the number of non-classifications is lower (average 449 vs. 5 345). This may be explained by the fact of using a reduced t_m , resulting in more cell migrations. Finally, between TC01 and TC05, the main difference is in the signal mapping approach, i.e., TC01 uses one signal while TC05 uses two signals. Between both, the *Two-Feature Approach* presents slightly better results in terms of classification rate. However, the *One-Feature Approach* achieves better detection duration. The global system performance in each test case is represented next.

TC01: 98.832%	TC05: 98.671%	TC09: 74.386%	TC13: 80.282%
TC02: 86.604%	TC06: 68.656%	TC10: 62.450%	TC14: 76.110%
TC03: 57.378%	TC07: 60.599%	TC11: 90.950%	TC15: 92.766%
TC04: 41.773%	TC08: 60.518%	TC12: 80.781%	TC16: 81.379%

Analysing the results from applying Eq. (7.1), we confirm that TC01 and TC05 are the best test cases, since they present the best global result. They both use a lifespan between 0-100, 10 iterations, 100 Dc in the population and the default At (0.5). The main difference is in the signal mapping approach. While TC01 follows a *One-Feature Approach* ($b_pktTotalCount$), TC05 uses a *Two-Feature Approach* ($b_pktTotalCount$ as Ss and $count$ as Ds).

7.2.5.3 Comparison and Discussion

In this section we discuss and compare findings about the application of *dDCA* and *iDCA* in the *M2M Using OPC UA*, including a testing set. The results that support the findings are presented in Table 7.5. We highlight the best P achieved in both *dDCA* and *iDCA*. In this case, the *iDCA* outperforms the *dDCA*.

Starting with the comparison of using the same technique in both training and testing sets, we use the *train test split* routine to export a 30% size of the original dataset, which is used as a

Table 7.5: *deterministic Dendritic Cell Algorithm* and *Incremental Dendritic Cell Algorithm* global performance comparison in *M2M Using OPC UA*.

Test Case	Technique	Dataset	Size	<i>AUC-ROC</i>	$R_{\Delta D}$	$R_{\emptyset}$	P
TC14	dDCA	Training set	107 633	100%	68.98%	99.98%	95.342%
TC14	dDCA	Testing set	32 290	100%	69.28%	100%	95.393%
TC01	iDCA	Training set	107 633	100%	93.45%	99.26%	98.832%
TC01	iDCA	Testing set	32 290	100%	98.72%	99.98%	99.802%
TC05	iDCA	Training set	107 633	100%	91.29%	99.91%	98.671%
TC05	iDCA	Testing set	32 290	100%	99.79%	99.96%	99.959%

testing set. Note that *DCA*⁴⁶-based approaches do not require a training phase to fit a model to the data. So, the *train test split* may be unnecessary since the model will not "see" any training data. However, *DCA*-based approaches rely heavily on linear transformations, based on the input signals, which are formed based on the training data. Thus, the *train test split* routine is helpful to assess whether a given algorithm is fitting the data nicely, or it may be presenting underfitting or overfitting problems.

Overall, the global performance of *dDCA* in the testing set is similar when compared with the training set (a difference of 0.051%). In the *iDCA*, the global performance increases slightly (an average difference of 1.129%) in the testing set. We can see that the detection performance is very high and similar in different test cases if we look closely. This means that both *dDCA* and *iDCA* models, considering their respective parametrization and pre-processing routines, present a good fit with the dataset. The same goes for the classification rate results. Considering the total dataset size, these algorithms tend to have high classification rates, i.e., insignificant non-classified data entries. Finally, distinctive differences are found in the detection rate. *iDCA* presents an improved rate of detection when compared with *dDCA*.

In conclusion, regarding the validation with *M2M Using OPC UA* dataset, results show that both *dDCA* and *iDCA* are good approaches for the detection problem. This is true for network environments with similar characteristics to the one used to build this dataset, i.e., automated *M2M*⁴⁷ communication between devices. This automation leads to regular communication in the network. Also, the attacks were injected into the network using automated tools, which maintains 'stability' in both normal and anomalous data. This may explain why most dataset features are largely correlated with the class label.

When looking closer to the results of *dDCA* and *iDCA*, we can conclude that the *iDCA*, in this dataset, is a better algorithm for intrusion detection. It enables an increased detection rate while preserving detection performance and classification rate. In terms of global performance, we can compare the results achieved by *dDCA* (TC14) and *iDCA* (average of TC01 and TC05). Results show an increase of 3.41% in the training set and 4.49% in the testing set global performance.

⁴⁶*Dendritic Cell Algorithm*⁴⁷*Machine-to-Machine*

7.2.6 KDD Cup99

Of all datasets, the *KDD Cup99* contains the most data entries, which results in the creation of 494 020 *Ags*. Among those, 80% of the entries represent a form of attack, resulting in a derived $At = 0.803$. Regarding signal mapping, *logged_in* is the most correlated feature, and it is used to form the input signals. If a *Two-Feature Approach* is used, *logged_in* is used to form the *Safe signal*, while *count* (the second-highest correlated feature) is used to form the *Danger signal*.

7.2.6.1 dDCA Validation

The *dDCA* was validated in the *KDD Cup99* training set. Results are presented in Table 7.6. We highlight the best *P* achieved using the *dDCA*. In this case they were TC07 and TC08.

Table 7.6: *deterministic Dendritic Cell Algorithm* validation results in *KDD Cup99* training set.

Test Case	<i>AUC-ROC</i>	<i>Acc</i>	<i>TPR</i>	<i>TNR</i>	$\emptyset$	Δt^D	<i>T</i>	<i>M</i>
TC03	93%	96.34%	98.84%	86.17%	176	26.665	131.237	2 469
TC04	92%	96.16%	98.68%	85.91%	53	26.899	132.719	2 489
TC07	99%	98.95%	98.72%	99.87%	154	119.012	589.209	7 792
TC08	99%	98.94%	98.76%	99.7%	119	119.262	587.926	7 784
TC09	87%	94.32%	99.15%	74.61%	1	1 578	8 008	171 497
TC10	87%	94.33%	99.15%	74.66%	1	1 495	7 629	171 844
TC13	92%	96.01%	98.81%	84.6%	10	4 132	21 055	409 677
TC14	92%	95.99%	98.85%	84.35%	15	3 330	16 876	409 416

Results on Table 7.6 show good detection *Acc* (average of 96.38%) while the number of non-classifications are kept to the minimum (average of 66 out of 494 020). However, the number of *FP* and *FN* is meaningful, and the overall *AUC-ROC* is not as good as it could be (average of 92%). The only test cases that present a low rate of false positives and highest *AUC-ROC* are TC07 and TC08. These test cases present a combination of high lifespan with a unique *Dc* and iteration. As explained before, this combination results in the the migration of less *Dc* (average *M* is 7 788), decreasing the overall processing time. TC07 and TC08 present a normal detection duration (average of 1.99 minutes or 119.137 seconds), on contrary to high detection duration of TC09, TC10, TC13 and TC14 (average of 43.9 minutes or 2 633 seconds).

TC03 and TC04 present the same parameter combination as TC07 and TC08 but fail to meet the same results (the unique advantage of the detection duration - average of 26.782 seconds). The main difference is that TC07 and TC08 use a two signal approach, which enables more information to the algorithm. Consequently, *Dc* can reach the migration threshold faster, so the number of migrations increases. By analyzing the results, the *dDCA* do not require the migration of a large number of cells to have good detection results. This turns out to be an advantage regarding the rate of detection, which is also reduced. Finally, between TC07 and TC08, the main

difference is with the At used, i.e., TC07 uses the default value of 0.5 while TC08 uses the derived version (0.803).

Based on the overall analysis of the dDCA results, we can conclude that TC09, TC10, TC13 and TC14 are not good options, since they can't reach highest detection results and the detection rate is poor. If the priority is to achieve the lowest detection duration, TC03 and TC04 are the best options since they deliver satisfactory results. On the other hand, if the detection Acc is the most important, then TC07 and TC08 are the best options. They deliver the best detection results and the detection duration is very reasonable. Considering Eq. (7.1), the global system performance in each test case is represented next.

TC03: 94.742%	TC09: 89.244%
TC04: 92.157%	TC10: 89.260%
TC07: 96.362%	TC13: 92.255%
TC08: 96.351%	TC14: 92.239%

Looking at the global system performance results, we conclude that TC07 and TC08 achieve the highest results (with a small tendency to TC07). These test cases use a *Two-Feature Approach* (*logged_in* as Ss and *count* as Ds), lifespan between 0-10 000, 1 iteration, 1 Dc in the population and the At can be both default or derived (0.803).

7.2.6.2 iDCA Validation

Table 7.7 presents the validation results for iDCA in the *KDD Cup99* training set. We highlight the best P achieved using the iDCA. In this case they were TC01, TC05 and TC15.

Similarly to the *M2M Using OPC UA*, the iDCA application in the *KDD Cup99* presents a high sensitivity regarding parameter's combination, since performance results are very different in distinct test cases. First, TC02, TC04, TC06, TC08, TC10, TC12, TC14 and TC16 present a non-acceptable detection accuracy (average Acc of 32.79%). The common input parameter among all these test cases is the $At = 0.803$. We can conclude, similarly to the *M2M Using OPC UA* evaluation, that the iDCA do not present good accuracy results when using derived At , especially when the threshold is this high.

Second, and considering only the remaining test cases, TC03, TC07, TC09 and TC13 present a prohibitive number of non-classifications (average of 335 402 out of 494 020). These test cases use a unique iteration and only one Dc in the population. This means that the migrations decrease and, consequently, the number of non-classification increases (due to the lack of migrated Dc). TC09 and TC13 present, in comparison with TC03 and TC07, slightly better results because of the decrease of t_m from 10 000 to 100. Reducing the migration threshold makes it easier for a Dc to migrate, fostering migration. However, the combination of these input parameters does not work.

Table 7.7: *Incremental Dendritic Cell Algorithm* validation results in *KDD Cup99*.

Test Case	<i>AUC-ROC</i>	<i>Acc</i>	<i>TPR</i>	<i>TNR</i>	$\emptyset$	Δt^D	<i>T</i>	<i>M</i>
TC01	86%	93.8%	99.17%	71.87%	210	32.542	2 462	2 144 462
TC02	54%	32.2%	18.27%	89.12%	210	38.884	2 269	2 144 356
TC03	89%	93.92%	98.31%	79.92%	490 993	32.81	64.152	3 027
TC04	50%	26.31%	5.41%	95.59%	490 987	30.771	63.532	3 033
TC05	87%	94.52%	99.17%	74.85%	5 049	61.724	3 331	4 094 170
TC06	55%	33.11%	19.46%	90.95%	5 055	60.602	3 328	4 094 191
TC07	99%	98.83%	98.75%	99.22%	486 209	28.935	69.779	7 811
TC08	63%	39.75%	26.66%	99.93%	486 222	25.594	70.639	7 798
TC09	87%	93.83%	99.02%	75.62%	279 827	30.974	68.486	214 193
TC10	51%	27%	7.62%	94.71%	279 584	27.528	67.921	214 436
TC11	87%	94.36%	98.95%	75.41%	10 059	31.052	484.088	30 059
TC12	55%	32.14%	17.67%	91.96%	9 940	29.563	488.462	30 075
TC13	99%	99.03%	99.03%	99.07%	84 582	28.945	75.864	409 438
TC14	62%	37.85%	25.77%	99.31%	84 512	30.807	77.091	409 508
TC15	98%	98.79%	99%	97.67%	28 704	30.903	559.619	78 262
TC16	60%	33.93%	21.91%	98.5%	28 685	30.084	554.116	78 280

The remaining test cases (TC01, TC05 and TC15) present overall satisfactory results. TC15 stands out with the best detection performance and duration, but in exchange, it has the most non-classifications, which is still rather low (28 704 in 494 020). To achieve this result, TC15 employs a combination of high t_m and uses the two signal approach. With this, it is possible to increase migrated Dc to a reasonable number that does not negatively impact the detection duration while keeping more information for the classification stage. The global system performance in each test case is represented next.

TC01: 91.391%	TC05: 91.667%	TC09: 71.255%	TC13: 89.397%
TC02: 72.132%	TC06: 72.471%	TC10: 50.372%	TC14: 66.929%
TC03: 60.881%	TC07: 68.575%	TC11: 90.729%	TC15: 96.519%
TC04: 37.889%	TC08: 47.76%	TC12: 71.589%	TC16: 73.734%

By analysing the results from applying Eq. (7.1), we confirm that TC01, TC05 and TC15 are the best test cases, since they present the best global result. Of those, TC15 stands out. It uses a *Two-Feature Approach* (*logged_in* as Ss and *count* as Ds), lifespan between 0-10 000, 10 iterations, 100 Dc in the population and the default At (0.5). Parameters of both TC01, TC05 are very similar. The main difference is in the use of a one signal approach (TC01) and using a max. lifespan of 100 (TC05).

7.2.6.3 Comparison and Discussion

In this section, we discuss and compare findings of the application of **dDCA** and **iDCA** in the *KDD Cup99*, including a testing set. The results that support the findings are presented in Table 7.8. We highlight the best P achieved in both **dDCA** and **iDCA**. In this case, the **dDCA** outperforms the **iDCA**.

Table 7.8: *deterministic Dendritic Cell Algorithm and Incremental Dendritic Cell Algorithm* global performance comparison in *KDD Cup99*.

Test Case	Technique	Dataset	Size	$AUC-ROC$	$R_{\Delta D}$	$R_{\emptyset}$	P
TC07	dDCA	Training set	494 020	99%	79.80%	99.97%	96.362%
TC07	dDCA	Testing set	148 206	100%	79.20%	99.96%	96.869%
TC08	dDCA	Training set	494 020	99%	79.71%	99.98%	96.351%
TC08	dDCA	Testing set	148 206	100%	78.77%	99.96%	96.807%
TC01	iDCA	Training set	494 020	86%	98.68%	99.96%	91.391%
TC01	iDCA	Testing set	148 206	85%	99.56%	99.99%	90.932%
TC05	iDCA	Training set	494 020	87%	98.15%	98.98%	91.667%
TC05	iDCA	Testing set	148 206	87%	99.80%	99.25%	91.983%
TC15	iDCA	Training set	494 020	98%	94.48%	94.19%	96.519%
TC15	iDCA	Testing set	148 206	66%	98.85%	94.04%	77.938%

Although *KDD Cup99* has available a dataset version for testing purposes, this dataset is specifically for unsupervised problems, i.e., it does not have any class labels or features. Since we need this information to assess the detection algorithms, we follow the same *train test split* strategy as in *M2M Using OPC UA*. Thus, we export a 30% size of the original dataset used as a testing set. Results are broadly similar when comparing global performance P between training and testing sets. The only test case that does not comply with the majority is TC15 (**iDCA**), which decreases performance by more than 18%.

It is usual to have slightly worst performances in testing sets, but this is a significant decrease in performance. This can be explained by the worst $AUC-ROC$, i.e., the capacity to distinguish correctly normal from anomaly. Since all the **iDCA** input parameters and runtime properties hold on both training and testing sets validation, it is unclear why the detection $AUC-ROC$ decreases. Also, this is a one-time case since TC01 and TC05 maintain competitive results while being in the same conditions.

Regarding the overall comparison between **dDCA** and **iDCA**, results show that **dDCA** and **iDCA** present a good fit with the *KDD Cup99* dataset. However, the **dDCA** stands-out. $AUC-ROC$ decreases (an average difference of 14.67%), despite still being very satisfactory (except on TC15). The $R_{\emptyset}$ is similar in both training and testing sets. However, the $R_{\Delta D}$ in **iDCA** is better when compared with **dDCA** (an average difference of 18.88%). Thus, despite being a good detection approach, there is an overall trade-off between the **iDCA** $R_{\Delta D}$ and detection Acc when compared to the **dDCA**, i.e., the $R_{\Delta D}$ increases greatly in exchange of reducing the detection Acc . If we

look closely to TC15 (training set), this trade-off is not evident and breaks the tendency of the other tests. In TC15 (training set), the *AUC-ROC* is similar to the *dDCA*. However, because of the extreme underperformance in the TC15 (testing set), there may be a change of this test case is overfitting to the training set.

In conclusion, regarding the validation with *KDD Cup99* dataset, overall results show the *dDCA* is a better algorithm for intrusion detection if the main goal is in detection *Acc* and *AUC-ROC*. Otherwise, if the $R_{\Delta D}$ is the most important, then the *iDCA* is the best choice. On average, the *iDCA* increases detection rate from 79.37% to 98.245% (79.755% to 97.22% in the training set, and 78.985% to 99.403% in the testing set). This significant detection rate increase has a trade-off of a poorer *AUC-ROC*, i.e., from an average of 99.5% to 84.83% (99% to 90.3% in the training set, and 100% to 79.3% in the testing set). Finally, in terms of *P*, results show a decrease of 3.16% from *dDCA* to *iDCA* in the training set, while in the testing set, the difference is up to 9.88%. If we exclude TC15 because it represents a parameter combination that does not generalize well, this difference is 5.38%.

7.2.7 NSL-KDD

Considering the *NSL-KDD* dataset, the *same_srv_rate* and *dst_host_srv_count* are the most correlated features, so they are used for signal mapping. *same_srv_rate* is used for the *One-Feature Approach*. If the *Two-Feature Approach*, *same_srv_rate* corresponds to the *Ds* while *dst_host_srv_count* corresponds to *Ss*. The dataset is shorter than the others considered in this evaluation methodology. It contains 25 192 data entries. Of those, almost half (46.6%) correspond to attacks, so the derived *At* corresponds to 0.466.

7.2.7.1 dDCA Validation

Table 7.9 presents the validation results for *dDCA* in the *NSL-KDD* training set. We highlight the best *P* achieved using the *dDCA*. In this case they were TC11 and TC12.

Results presented in Table 7.9 show that most of the time, the *dDCA* is very accurate in detecting anomalies and distinguishing them from normal instances. Test cases that don't present good *Acc* results are TC09, TC10, TC13 and TC14 (average *Acc* of 75%). Curiously, they present the highest migrated *Dc* (average of 15 984 per *Ag*) and a normal detection duration (average of 20.86 seconds), despite sharing minimum iterations and size of the cell population. This is because they have a decreased migration threshold, which results in a very active migration process.

From the remaining, TC11, TC12, TC15 and TC16 present a high detection duration (1 227.29 seconds or 20.45 minutes), but a very good detection *Acc* (average of 99.55%). They use $\#Dc = 100$ and $\#Iter = 10$, while the t_m is increased to 10 000. A large lifespan is great for detection results, but it also increases the detection time. Considering TC03, TC04, TC07 and TC08, these test cases present very good *Acc* results, along with an increased performance in the detection duration (average of 0.609 seconds). As downsize is the number of non-classifications (average of 90 out of 25 192), which is still good for the standards.

Table 7.9: *deterministic Dendritic Cell Algorithm* validation results in *NSL-KDD*.

Test Case	<i>AUC-ROC</i>	<i>Acc</i>	<i>TPR</i>	<i>TNR</i>	$\emptyset$	Δt^D	<i>T</i>	<i>M</i>
TC03	99%	99.49%	98.91%	100%	139	0.376	0.655	212
TC04	100%	99.66%	99.27%	100%	31	0.394	0.703	210
TC07	100%	99.74%	99.51%	99.94%	108	0.827	1.476	542
TC08	99%	98.86%	97.97%	99.63%	81	0.839	1.521	573
TC09	82%	83.58%	66.04%	98.89%	2	18.874	35.324	14 383
TC10	82%	83.61%	65.99%	98.99%	0	18.645	34.86	14 331
TC11	100%	99.76%	99.48%	100%	0	958.585	1 759	2 101
TC12	100%	99.80%	99.57%	100%	0	974.163	1 801	2 106
TC13	66%	66.21%	55.6%	75.48%	0	22.659	42.514	17 606
TC14	66%	66.6%	56.49%	75.43%	1	23.273	43.81	17 619
TC15	99%	99.28%	98.49%	99.97%	0	2 091	3 913	5 512
TC16	99%	99.37%	98.71%	99.96%	0	1 844	3 438	5 502

With a low detection duration, these test cases use the combination of only one iteration and the minimum size of the cell population as algorithm inputs. At the same time, the lifespan is large (max. 10 000). Consequently, the *M* is relatively low, i.e., the existing *Dc* migrates on average 385 times. Finally, based on the overall analysis of the dDCA results, we can identify the best test case results, assuming the three goals mentioned before.

If the priority is in the detection rate, TC03, TC04, TC07, TC08, present the best results. On the other hand, if detection performance is really important, TC03, TC04, TC07, TC08, TC11, TC12, TC015 and TC16 are good options. TC03, TC04, TC07 and TC08 are the best of both worlds, i.e., fast and accurate detection performance. Considering Eq. (7.1), the global system performance in each test case is represented next.

TC03: 90.652%	TC11: 91.828%
TC04: 91.564%	TC12: 91.887%
TC07: 91.487%	TC13: 71.605%
TC08: 91.043%	TC14: 71.631%
TC09: 81.183%	TC15: 91.382%
TC10: 81.177%	TC16: 91.352%

There is differences between test cases. Results confirm the previously analysis, i.e., the best test cases are TC03, TC04, TC07, TC08, TC11, TC12, TC015 and TC16. The tendency of highest results correspond to TC11 and TC12. These test cases use a *One-Feature Approach* (*same_srv_rate*), lifespan between 0-10 000, 10 iterations, 100 *Dc* in the population. The *At* can be either the default or derived (in this scenario, both are practically the same), but there is a slight tendency of being preferable the derived version (0.466).

7.2.7.2 iDCA Validation

Table 7.10 presents the validation results for iDCA in the NSL-KDD training set. We highlight the best P achieved using the iDCA. In this case they were TC01 and TC02.

Table 7.10: Incremental Dendritic Cell Algorithm validation results in NSL-KDD.

Test Case	AUC-ROC	Acc	TPR	TNR	$\emptyset$	Δ^D	T	T
TC01	86%	86.38%	76%	95.45%	3	0.104	156.519	143 034
TC02	86%	86.4%	76.06%	95.45%	3	0.114	154.044	143 036
TC03	51%	43.2%	2.5%	100%	24 986	0.109	2.997	206
TC04	51%	42.72%	1.67%	100%	24 986	0.103	2.945	206
TC05	60%	60.63%	44.9%	74.28%	89	0.116	156.471	178 794
TC06	60%	60.85%	45.5%	74.15%	86	0.11	155.73	178 811
TC07	52%	81.35%	6.32%	97.12%	24 645	0.115	3.411	547
TC08	51%	82.92%	4.6%	97.26%	24 630	0.111	3.268	562
TC09	74%	71.57%	50.03%	97.27%	10 985	0.102	3.29	14 207
TC10	74%	72.24%	51.26%	97.21%	10 904	0.103	3.376	14 288
TC11	57%	58.53%	38.9%	75.66%	88	0.117	26.225	2 109
TC12	58%	59.34%	41.53%	74.88%	93	0.107	26.214	2 108
TC13	69%	72.41%	62.21%	75.62%	7 891	0.112	3.649	17 301
TC14	69%	72.22%	61.66%	75.62%	7 807	0.105	3.575	17 385
TC15	54%	56.45%	35.05%	71.95%	2 058	0.108	30.001	5 510
TC16	53%	56.02%	36.79%	69.97%	2 041	0.104	30.425	5 505

By inspecting the results, we can draw two immediate conclusions. First, different parameter combinations on the iDCA do not have an impact on the detection duration (average 0.109 seconds and standard deviation of 0.005). Second, overall, the iDCA is not a good approach if the detection Acc is a priority (average of 66.45%). Only TC01 and TC02 present competitive detection Acc (average of 86.39%) while the other test cases present very poor Acc (average of 63.60%). The worst test cases are TC03 and TC04, which present an Acc below 0.5 (a threshold that corresponds to picking the classification randomly). So, these results, especially considering the good performance of dDCA, are very disappointing.

The results show that the test cases that follow the combination of one iteration and one sized cell population present a higher number of non-classifications. This is verifiable in test cases TC03, TC04, TC07, TC08, TC09, TC10, TC13 and TC14. Of those, the test cases that adhere to a higher t_m stands out for the worst by presenting on average 18 432 non-classifications out of 25 192. This is explained by the low number of migrations (average of 380 migrations per cell).

Considering the test cases using a combination of 10 iterations and cell population size of 100 (TC01, TC02, TC05, TC06, TC11, TC12, TC15 and TC16), the number of non-classifications are greatly reduced (average of 558 out of 25 192). The number of non-classifications is especially worse in TC15 and TC16, which uses a higher t_m and a two signal approach. On contrary, reduced number of non-classifications and high detection Acc results are found in TC01 and TC02, by

combining a lower t_m with a one signal approach. However, despite the best result, detection Acc is not competitive when comparing to dDCA results. The global system performance in each test case is represented next.

TC01: 91.587%	TC05: 75.901%	TC09: 73.031%	TC13: 73.11%
TC02: 91.586%	TC06: 75.904%	TC10: 73.12%	TC14: 73.213%
TC03: 45.258%	TC07: 46.235%	TC11: 74.046%	TC15: 70.304%
TC04: 45.280%	TC08: 45.644%	TC12: 74.647%	TC16: 69.723%

Analysing the results from applying Eq. (7.1), we confirm that TC01 and TC02 are the best test cases, since they present the best global result. They both use a *One-Feature Approach* (*same_srv_rate*), lifespan between 0-100, 10 iterations and 100 Dc in the population. Between TC01 and TC02, the difference is in the At . TC01 uses the default At while TC02 uses the derived version.

7.2.7.3 Comparison and Discussion

In this section, we discuss and compare findings of the application of dDCA and iDCA in the *NSL-KDD* dataset, including a testing set. The results that support the findings are presented in Table 7.11. We highlight the best P achieved in both dDCA and iDCA. In this case, the dDCA outperforms the iDCA.

Table 7.11: *deterministic Dendritic Cell Algorithm* and *Incremental Dendritic Cell Algorithm* global performance comparison in *NSL-KDD*.

Test Case	Technique	Dataset	Size	$AUC-ROC$	$R_{\Delta D}$	$R_{\emptyset}$	P
TC11	dDCA	Training set	25 192	100%	45.52%	100%	91.828%
TC11	dDCA	Testing set	22 544	97%	54.78%	100%	91.418%
TC12	dDCA	Training set	25 192	100%	45.91%	100%	91.887%
TC12	dDCA	Testing set	22 544	98%	55.38%	100%	92.106%
TC01	iDCA	Training set	25 192	86%	99.93%	99.99%	91.587%
TC01	iDCA	Testing set	22 544	73%	99.83%	99.99%	83.772%
TC02	iDCA	Training set	25 192	86%	99.93%	99.99%	91.586%
TC02	iDCA	Testing set	22 544	72%	99.93%	99.99%	83.187%

Overall, the $AUC-ROC$ of dDCA is better when compared with iDCA in both training and testing sets. Regarding the dDCA, there is a slight (normal) decrease (an average difference of 2.5%) between the $AUC-ROC$ of training and testing sets. On the other hand, the iDCA presents a significant underachievement in testing sets when compared to the results in the training sets (an average difference of 13.3%). Also, comparing the $AUC-ROC$ of dDCA and iDCA, even considering the training set for best results, there is an average difference of 19.5%. This means that the iDCA is underfitting the data since it is unable to encapsulate the relations among data

properly. The solution may be in increasing the model complexity by considering more features to form the signals or using different combinations for $\#Dc$, $\#Iter$, and t_m . We extended the $iDCA$ tests with multiple other combinations and could not find a major impact in the $AUC-ROC$ results. Thus, maybe the model complexity should take into account non-linear relationships among features.

Regarding P , the $dDCA$ achieves similar results in both training and testing sets (an average difference of 0.096%). On the other hand, the $iDCA$ presents in testing sets poorer P when compared with training sets (an average difference of 8.1%), mainly due to the underperformance in the $AUC-ROC$. When comparing P between $dDCA$ and $iDCA$, the difference is reduced when compared to the $AUC-ROC$ comparison. The $AUC-ROC$ under-performance of $iDCA$ is compensated by the great improvement of the $R_{\Delta D}$, leading to an average P difference of 4.2%). In conclusion, both $dDCA$ and $iDCA$ are sensitive to different combination of input parameters. $dDCA$ seem to be the best approach when detection $AUC-ROC$ is the priority. However, according to the P , the $iDCA$ is very competitive with $dDCA$ (except when used in the testing set).

7.2.8 UNSW-NB15

From the *UNSW-NB15* dataset, it is possible to create 82 332 Ag s. Among those, 55% represent an attack, resulting in a derived At of 0.551. Regarding signal mapping, $sttl$ is the most correlated feature, and it is used to form the input signals. If a *Two-Feature Approach* is used, $sttl$ is used to form the *Danger signal*, while $swin$ (the second-highest correlated feature) is used to form the *Safe signal*.

7.2.8.1 dDCA Validation

The $dDCA$ was validated in the *UNSW-NB15* training set. Results are presented in Table 7.12. We highlight the best P achieved using the $dDCA$. In this case they were TC07, TC08 and TC15.

Table 7.12: *deterministic Dendritic Cell Algorithm* validation results in *UNSW-NB15* training set.

Test Case	$AUC-ROC$	Acc	TPR	TNR	$\emptyset$	Δt^D	T	M
TC03	72%	72.24%	77.76%	65.48%	137	2.722	5.798	623
TC04	72%	72.53%	79.03%	64.57%	55	2.634	5.573	598
TC07	78%	76.94%	64.12%	92.64%	5	7.401	16.199	2 014
TC08	78%	76.67%	63.83%	92.39%	15	7.426	16.292	1 961
TC09	68%	68.74%	73.42%	63.01%	0	157.522	349.425	42 848
TC10	68%	68.78%	73.43%	63.08%	1	159.935	354.362	42 897
TC11	72%	72.98%	79.95%	64.45%	0	7 434	16 053	6 169
TC12	72%	72.73%	79.08%	64.94%	0	6 707	14 819	6 157
TC13	77%	75.88%	66.14%	87.81%	0	275.873	612.382	77 013
TC14	77%	75.78%	66.15%	87.59%	0	282.792	627.962	77 043
TC15	79%	77.21%	64.34%	92.98%	0	18 001	39 893	20 004

By analyzing the results, we can immediately draw three main conclusions. First, the overall number of non-detections is negligible through all the test cases (average of 19 out of 82 332 data entries). Second, we can conclude that the dDCA presents an overall poor detection *Acc* (average of 73.68%). Finally, despite low, the *Acc* is stable through all the test cases (the standard deviation is 0.03).

Looking closely at the results, there are three categories of detection duration:

- Low duration (average of 5.046 seconds) considering TC03, TC04, TC07 and TC08.
- Normal duration (average of 3.65 minutes or 219.03 seconds) considering TC09, TC10, TC13 and TC14.
- High duration (average of 2.97 hours or 10 714.7 seconds) considering TC11, TC12 and TC15.

Low detection duration cases are characterized by the combination of $\#Dc = 1$, $\#Iter = 1$, and $t_m = 10\,000$. This combination, as seen before, results in fewer migrations (average of 1 299) and a shorter duration of the classification. On the contrary, high detection duration test cases adhere to the combination of $\#Dc = 10$, $\#Iter = 100$ and $t_m = 10\,000$. Increasing the cell population and the number of iterations increases the number of migrations (average of 10 776), resulting in more time spent in the classification process. Normal rate test cases present a combination that works well ($\#Dc = 1$, $\#Iter = 1$, and $t_m = 100$), since the expected low number of migration, due to the small population size and one iteration only, is not verified thanks to the *Dc* lifespan. Consequently, the migration ratio is high (average of 59 950), but the duration of the classification is acceptable.

Despite being stable through all the test cases, the best detection *Acc* results tend mostly for test cases that use a *Two-Feature Approach*. Considering Eq. (7.1), the global system performance in each test case is represented next.

TC03: 76.117%	TC09: 74.038%	TC13: 79.443%
TC04: 76.093%	TC10: 74.03%	TC14: 79.445%
TC07: 79.946%	TC11: 76.253%	TC15: 80.631%
TC08: 79.958%	TC12: 76.410%	

Looking at the global system performance results, we conclude that TC07, TC08 and TC15 achieve the highest results (with a small tendency to TC15). These test cases use a *Two-Feature Approach* (sttl as *Ds* and swin as *Ss*) and lifespan between 0-10 000. Regarding $\#Dc$ and $\#Iter$, TC07 and TC08 are characterized by one iteration and one *Dc* in the population, while TC15 has $\#Dc = 100$ and $\#Iter = 10$. Finally, *At* can be both default or derived (0.551), but the tendency for best results is when using the default *At*.

7.2.8.2 iDCA Validation

Table 7.13 presents the validation results for iDCA in the UNSW-NB15 training set. We highlight the best P achieved using the iDCA. In this case they were TC15 and TC16.

Table 7.13: Incremental Dendritic Cell Algorithm validation results in UNSW-NB15.

Test Case	AUC-ROC	Acc	TPR	TNR	$\emptyset$	Δt^D	T	M
TC01	71%	71.51%	80.37%	60.64%	12	1.587	442.099	402 848
TC02	70%	70.85%	78.56%	61.4%	12	1.562	447.339	402 823
TC03	73%	73.12%	74.07%	72.43%	81 748	1.07	11.058	584
TC04	76%	75.67%	75.78%	75.59%	81 736	1.318	10.669	596
TC05	72%	71.22%	66.32%	77.38%	1 056	2.079	582.026	773 832
TC06	72%	71.22%	66.32%	77.39%	1 057	2.197	601.026	773 827
TC07	75%	73.44%	56.8%	92.98%	80 318	1.294	11.799	2 014
TC08	76%	73.51%	58.39%	92.68%	80 350	1.005	11.71	1 982
TC09	71%	70.45%	71.01%	70%	42 192	1.374	11.643	40 140
TC10	71%	70.5%	70.91%	70.16%	42 140	1.172	11.93	40 192
TC11	70%	70.36%	76.25%	63.42%	1 754	0.917	81.974	5 857
TC12	70%	70.14%	75.36%	63.98%	1 778	1.082	82.479	5 852
TC13	77%	74.94%	66.19%	87.3%	5 318	1.057	13.17	77 014
TC14	77%	74.92%	66.21%	87.24%	5 275	1.305	12.951	77 057
TC15	79%	77.02%	66.15%	90.87%	1 509	1.025	101.018	20 013
TC16	78%	76.46%	64.79%	91.3%	1 459	1.273	99.911	20 023

Similarly to the dDCA, the iDCA application in the UNSW-NB15 presents stable (but poor) Acc results (average of 72.83% and standard deviation of 0.023). Also, the number of non-classifications is significant in some cases, namely TC03, TC04, TC07 and TC08, with an average of 98.43%. This is mainly due to fewer iterations and lower cell population size. If we look closely to the other test cases with a similar combination, i.e., TC09, TC10, TC13 and TC14, the number of non-classification is not as high, because $t_m = 100$. This means that the cell lifespan decreases, and, in consequence, the migrations increase, and the number of non-classified Ag decreases. Moreover, because TC13 and TC14 follow a *Two-Feature Approach*, the number of migrations increases, which means that one of the main reasons for non-classification (lack of migrated cells) is mitigated.

When analyzing the detection duration, we can conclude that it is also very stable along with all test cases (average of 1.33 seconds and standard deviation of 0.37). Thus, we can conclude that the best results are achieved by test cases TC15 and TC16. The global system performance in each test case is represented next.

By analysing the results from applying Eq. (7.1), we confirm that TC15 and TC16 are the best test cases, since they present the best global result. Of those, TC15 stands out. It uses a *Two-Feature Approach* (sttl as Ds and swin as Ss), lifespan between 0-10 000, 10 iterations, 100 Dc in the population and the default At (0.5).

TC01: 82.542%	TC05: 82.826%	TC09: 68.018%	TC13: 83.381%
TC02: 81.944%	TC06: 82.824%	TC10: 68.33%	TC14: 83.087%
TC03: 57.526%	TC07: 58.967%	TC11: 81.3%	TC15: 86.79%
TC04: 58.928%	TC08: 59.915%	TC12: 81.263%	TC16: 86.116%

7.2.8.3 Comparison and Discussion

In this section, we discuss and compare findings of the application of **dDCA** and **iDCA** in the *UNSW-NB15*, including a testing set. The results that support the findings are presented in Table 7.14. We highlight the best P achieved in both **dDCA** and **iDCA**. In this case, the **iDCA** outperforms the **dDCA**.

Table 7.14: *deterministic Dendritic Cell Algorithm and Incremental Dendritic Cell Algorithm* global performance comparison in *UNSW-NB15*.

Test Case	Technique	Dataset	Size	$AUC-ROC$	$R_{\Delta D}$	$R_{\emptyset}$	P
TC07	dDCA	Training set	82 332	78%	54.31%	99.99%	79.946%
TC07	dDCA	Testing set	175 341	80%	68.41%	100%	83.262%
TC08	dDCA	Training set	82 332	78%	54.42%	99.98%	79.958%
TC08	dDCA	Testing set	175 341	80%	66.8%	99.99%	83.018%
TC15	dDCA	Training set	82 332	79%	54.88%	100%	80.631%
TC15	dDCA	Testing set	175 341	79%	64.01%	100%	82.001%
TC01	iDCA	Training set	82 332	71%	99.64%	99.99%	82.542%
TC01	iDCA	Testing set	175 341	81%	86.42%	100%	87.562%
TC15	iDCA	Training set	82 332	79%	98.99%	98.17%	86.79%
TC15	iDCA	Testing set	175 341	83%	68.76%	97.61%	84.518%
TC16	iDCA	Training set	82 332	78%	98.73%	98.23%	86.166%
TC16	iDCA	Testing set	175 341	82%	68.88%	97.61%	83.935%

Overall, the $AUC-ROC$ results and global performance P between training and testing sets, using both **dDCA** and **iDCA**, are poor along with several test cases (average of 79%). On the other hand, $R_{\Delta D}$ improves with the **iDCA**, presenting different results between training and testing sets (in testing sets, the $R_{\Delta D}$ is not as good as in training sets). Finally, the **iDCA** present a slight decrease in the $R_{\emptyset}$ in comparison with the **dDCA**, but overall this is stable among different test cases using the same technique. Thus, looking closely at the results, P is better in the **iDCA** when compared to the **dDCA**.

Considering the **dDCA**, when comparing P between training and testing sets, results are better in testing sets. There is a regular slight decrease in the $AUC-ROC$ (an average difference of 2%). On the other hand, the $R_{\Delta D}$ improves in the testing set (an average difference of 13.24%). The $R_{\emptyset}$ is similar. Thus P increases in the testing set (an average difference of 3.18%).

The scenario is different in the **iDCA**. The $AUC-ROC$ improves in the testing set (an average difference of 6%). However, the $R_{\Delta D}$ is worse in the testing sets (an average difference of 24.4%).

There is also a small variation in the $R_{\emptyset}$ among test sets using the *iDCA*. When considering TC01, P is better in the testing set (difference of 5.02%). On the other hand, regarding TC15 and TC16, P decreases slightly between training and testing sets using the *iDCA* (average of 2.25%), which is mainly due to the worse $R_{\Delta D}$.

In conclusion, regarding the validation with *UNSW-NB15* dataset, global performance results show the *iDCA* is a better algorithm for intrusion detection. Most of the time, the algorithm can improve the *AUC-ROC* while having a high detection rate. The detection rate is also greatly improved, especially in training sets. However, the overall *AUC-ROC* may not be competitive with other techniques since it is low. This means the solution may be underfitting the dataset, so increasing the model complexity is a solution. Also, despite low, the results are stable with different parameter combinations, which means that this complexity may consist of using non-linear approaches for the classification. Both *dDCA* and *iDCA* are not build to be used in non-linear problems.

Also, note that, despite TC01 did not achieved the highest result for P in the validation of the *iDCA* in the training set, this is not the case for the testing set. Despite, in general, the *iDCA* outperforms the *dDCA*, the test case TC01 stands out.

7.2.9 Overall Results

In terms of achieved results, we assess the *dDCA* and *iDCA* separately in all datasets and collect the results for further analysis. The primary assessment objectives are the *AUC-ROC*, the detection rate $R_{\Delta D}$, and the classification rate $R_{\emptyset}$. It is possible to derive a global system performance P in a unique metric considering the combination of the three objectives. Considering only the best achievements among these three domains, we can identify the best algorithm parameter combination in different datasets.

Regarding *dDCA*, the best parameter combination is TC07 and TC08. These test cases follow a *Two-Feature Approach*, $t_m = 10\ 000$, $\#Dc$ and $\#Iter = 1$. The At can be the default or the derived. As for the *iDCA*, the best parameter combination is TC05 or TC01. These test cases use $t_m = 100$, $\#Dc = 100$ and $\#Iter = 10$. The At should be the default 0.5. The only difference is in the signal mapping approach, since TC05 follows a *Two-Feature Approach* while TC01 follows a *One-Feature Approach*.

Since the *iDCA* processes a unique Ag in sequential order, instead of a Ag set, there is the need to compensate for the lack of statistical information available. Thus, increasing the cell population size and using the iterations parameter to extend the Ag sampling and Dc migration significantly is essential. Despite different Dcs sample the same Ag multiple times, the result is quite similar when compared to the equivalent in the *dDCA*, where one Dc samples multiple different Ag s one time. Also, to help with the migration process, in the *iDCA*, the t_m is reduced to foster cells to migrate faster.

In terms of runtime properties, if the number of migrations increases, non-classifications decrease. The only problem with a higher migration rate is the duration of the classification, which increases accordingly. Thus, in the *dDCA*, most of the time, a combination of $t_m = 100$, $\#Dc = 100$

and $\#Iter = 10$ is prohibited. However, in the **iDCA** is fine because the algorithm only has to classify one unique **Ag** at each time by relating it with a large set of migrated **Dcs**. In the **dDCA**, there is also a vast set of **Ag**, which will take much more time to process and output a classification. Moreover, there may be situations where the number of migrated cells is so high that even the **iDCA** is relatively slow. This is especially true when using a *Two-Feature Approach*. With this approach, the cell reaches the t_m much faster, which results in more migrations.

Thus, in the **iDCA**, using a *One-Feature Approach* is enough to reach good detection durations. The downside is that most of the time, the **AUC-ROC** is worse when compared with the **dDCA**. Nevertheless, when looking at the trade-off between accuracy, and detection time, using the **iDCA** will result in competitive global performances. Also, in terms of non-classifications, there is a small under-performance. Overall, both techniques behave similarly and are suitable for anomaly-based intrusion detection. Table 7.15 represents the average results of each technique among all the datasets used for validation.

Table 7.15: *deterministic Dendritic Cell Algorithm and Incremental Dendritic Cell Algorithm* average global performance comparison considering all datasets.

Test case (Average)	Technique	AUC-ROC	$R_{\Delta D}$	$R_{\emptyset}$	P
All	dDCA	93.5%	64.38%	99.99%	90.75%
	iDCA	85%	94.7%	98.9%	89.99%
Best	dDCA	94.5%	68.1%	99.99%	91.9%
	iDCA	91.3%	95.2%	98.5%	93.9%

Finally, despite these suggestions about the best parametrization with each technique, it is important to understand that both **dDCA** and **iDCA** are very sensitive when updating the parameter combination. These suggestions may not hold in different scenarios. In fact, in some cases, these suggestions may not hold when considering the same dataset but exchanging between testing set instead of the training set. Thus, each problem context should be considered carefully for the parameter tuning of each algorithm.

7.3 E2. A-HIDS Pipeline

7.3.1 Experimental Design

Vargas Martinez and Vogel-Heuser [70] contend that, due to restricted computing resources, integrating an **HIDS** in embedded industrial devices might be difficult, particularly under rigorous operating constraints. As a result, the authors abstracted a set of criteria while taking into account both **ICT HIDS** capabilities and industrial device considerations. In order to examine the feasibility of an **IDS** solution in such a scenario, industrial devices should adhere to industry surroundings and security requirements. According to the authors, the following are crucial needs for a generic **HIDS**:

- *Configurability*: It should be possible to configure the **HIDS** according to the requirements of the system to be protected.
- *Configuration and knowledge security*: The **HIDS** configuration and collected information for analysis should be protected.
- *Resiliency*: The **HIDS** should be available all the time, and so itself should be protected against attacks.
- *Low-performance overhead*: While executing in the host being protected, the **HIDS** should not negatively impact the system's performance.
- *Low detection time*: Quick detection and response to intrusions to avoid extra damage in the system being attacked.
- *Interoperability with other ICT*: Interaction with other security tools in the system for an overall security evaluation.
- *Centralized **HIDS** configuration & management*: Approach to manage distributed systems, such as **CPPSs**.
- *Log collection*: Enable log data for further processing and analysis.
- *System audit*: Collect host-related performance metrics.
- *Host network analysis*: Analyze network traffic in the device (input and output network packets).
- *Event-triggered or monitoring analysis*: Intrusion detection by monitoring host-related data.
- *Passive intrusion detection actions*: Logging and trigger alarms when facing important **IDS** events.

Taking the **A-HIDS** Pipeline into account, the suggested approach allows an **iDCA**-based **IDS** to identify real-time network assaults in **CPPSs** by analyzing incoming/outgoing **OPC UA**⁴⁸ network packets in the Edge device. As a result, it is aimed to validate the suggested approach's appropriateness in resource-constrained Edge devices. Even though it analyzes network packages, the solution may be categorized as an **HIDS** since it is run within Edge devices and only analyzes local network packets. As a result, integrating the **A-HIDS** into the **IIoT**⁴⁹ application might be difficult. Furthermore, some system operating limitations may not be met while running.

7.3.2 Testing Methodology

In this case, the **A-HIDS FB**'s evaluation may consider several metrics in various system characteristics, such as: i) Detection *Acc*; ii) *Resource Consumption*; and iii) *Performance Overhead*. Because **A-HIDS** analyzes network-related data, there is the benefit of doing offline assessments

⁴⁸*OPC Unified Architecture*

⁴⁹*Industrial Internet of Things*

in datasets, where typical system parameters assessed include *AUC-ROC*, $R_{\Delta D}$, and $R_{\emptyset}$. The effectiveness of the *iDCA*, in terms as *Acc*, as an online anomaly detection algorithm for network data has been considered in Section 7.2.

The two system properties suitable to be considered for assessment purposes are *Resource Consumption* and *Performance Overhead* system properties. *Resource Consumption* refers to the evaluation of the *A-HIDS*'s resource utilization. In contrast, the *Performance Overhead* refers to the evaluation of the overhead added by the *A-HIDS* to the Edge device where it is deployed. This means that, in order to be evaluated, the *A-HIDS* must be deployed in a physical test case scenario consisting of Edge devices in an *M2M* scenario. These techniques, however, provide significant problems for further comparing acquired results with similar work in the literature.

First, because the used testbed is not a benchmark (see more detail in Section 7.3.3), it is hard to analyze and compare the proposed solution to other similar solutions consistently. Even if the experimental setup consisted in building/simulating a physical test scenario identical to an existing one in the literature for comparison reasons, various Edge devices might not have the same baseline circumstances. As a result, the assessment findings of the same solution may change drastically when tested on different host systems. As a result, despite being considered for analyzing the feasibility of deploying the *A-HIDS* in an industrial Edge layer, this approach is not suitable to be used for result comparison purposes. Thus, a suitable assessment metric must be used to compare the proposal with previous work focusing on the development of security approaches based on the IEC 61499 standard. In this case, the assessment metric is the impact that a given *FB* application has in terms of design complexity and *Maintainability*, i.e., the effort required to develop, deploy, and refine secure *CPPS/ICS*⁵⁰ applications. According to *MBE* and IEC 61499 benefits for designing and sustaining *CPPS/ICS*s, design complexity and maintainability of such solutions are essential measures to assess the real effect in the entire industrial (see Section 2.3.3).

7.3.3 Experimental Setup

The feasibility of this approach was assessed by deploying the solution in a lab *CPPS* testbed, inspired in the same setup used to generate the *M2M using OPC UA* dataset (see Section 6.2.1). The lab *CPPS* testbed represents a typical *M2M* situation between Edge devices in a *CPPS*, as shown in Figure 7.1. Each Edge device consists of a *Raspberry Pi 4 Model B* with 2GB *RAM* that is linked to a cabled *LAN*⁵¹ supported by a standard network switch. Each Edge device runs a *DINASORE*⁵² instance with all of its dependencies. Finally, there is a malicious node (`pi@192.168.0.10`) that gains network access and launches various attacks such as *DoS*⁵³, *Message Spoofing*, and *MITM*⁵⁴.

Each Edge device has an *OPC UA* server (thanks to *DINASORE*) that allows it to publish to *OPC UA* variables updates about various sensor readings. For proof of concept, and since

⁵⁰*Industrial Control Systems*

⁵¹*Local-Area Network*

⁵²*Dynamic INtelligent Architecture for Software and MODular REconfiguration*

⁵³*Denial of Service*

⁵⁴*Man-in-the-Middle*

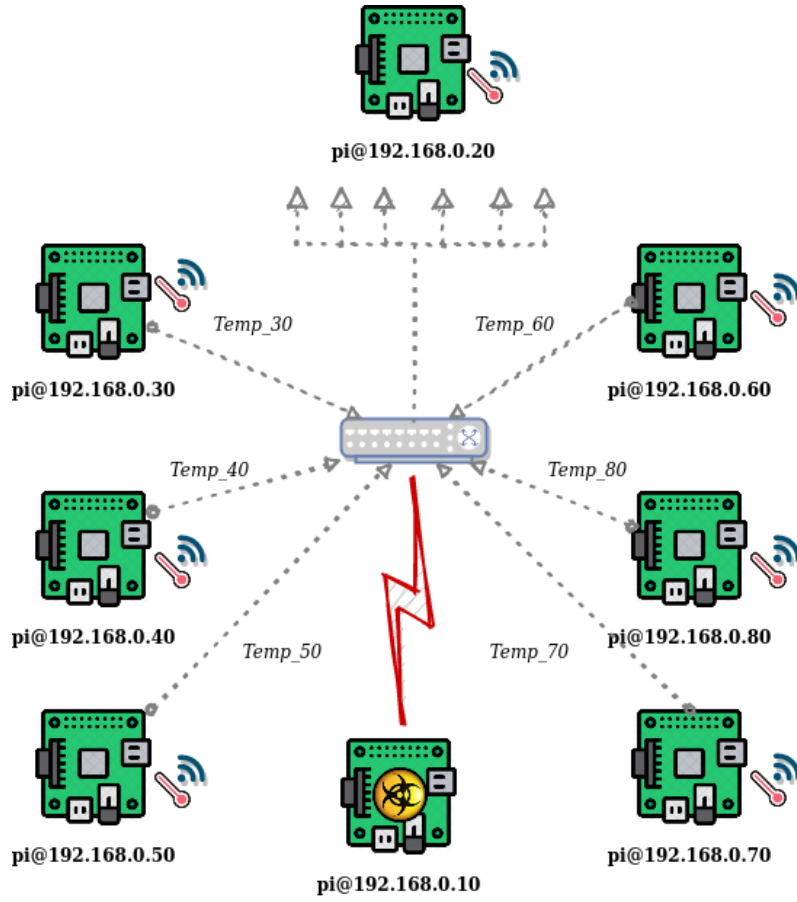


Figure 7.1: Experimental setup architecture.

OPC UA payload data is not used for detection purposes, we enable each Edge device with a temperature sensor. Additionally, one device (`pi@192.168.0.20`) implements an **OPC UA** client and subscribes to all **OPC UA** variables from all other devices on the same network. For testing reasons, the **A-HIDS** is deployed and operated solely in this device (`pi@192.168.0.20`), because it has the highest *Resource Consumption* and *Performance Overhead*, and it is the source of the majority of the **OPC UA** traffic in the network. Figure D.1 in Appendix D represents the **FB** Pipeline used to design and deploy the described functionalities among the Edge devices in the network. Besides the already mentioned *OPCUA_Flow_Sniffer* and the *iDCA FB*, responsible for the **A-HIDS** functionalities, there are other **FBs** to be considered in this experimental setup:

- **SENSOR_SIMULATOR** - This **FB** introduces the functionality of simulating sensor readings. There is one **SENSOR_SIMULATOR** for each Edge device, which contains three main data inputs: 1) **UA_NAME** - Name of the **OPC UA** variable that will contain the sensor reading updates; 2) **RANGE** - Range of possible values to be simulated, according to a Gaussian distribution; 3) **RATE** - Frequency of the sensor reading update/generation; Also, there is a data output (**VALUE**), which makes available the sensor readings.

- `CONNECT OPCUA` - This **FB** is responsible to create an **OPC UA** client. This client connects to an existing **OPC UA** server, so data input can be the address (`ADDRESS`) and port (`PORT`), in order to create and connect to the **OPC UA** server device. As data output, the `CONNECT OPCUA` makes available the connection object (`CLIENT`), which will be passed as input to the `OPCUA_SUBS`.
- `OPCUA_SUBS` - This **FB** enables the subscription of one **OPC UA** variable. As data inputs, `Node_ID`, and the `CLIENT` are used to specify the path for the **OPC UA** variable to be subscribed, considering the client object. `DATA_OUT` makes available the subscribed sensor readings.

There are 7 Edge devices in this experimental testbed, all of which are simulating temperature sensor data and one of which is subscribing to all sensor values. The **FBs** mapped to the subscribing device (`pi@192.168.0.20`) are highlighted in yellow, while the **FBs** that execute on other devices are highlighted in various colors for each device mapping on the bottom line. So, on each Edge device, a `SENSOR_SIMULATOR` **FB** is deployed, with the **FB** instances labeled *TEMP20* to *TEMP80*, depending on the device (`pi@192.168.0.20` to `pi@192.168.0.80`).

7.3.4 Results & Discussion

The proposed approach is evaluated in two ways: it evaluates the performance overhead of the **IDCA** technique when implemented in the industrial Edge layer, as well as crucial **FB** Pipeline criteria like *Design Complexity* and *Maintainability*.

7.3.4.1 Performance Overhead

The evaluation of the suggested solution takes into account different system parameters following the previously specified **HIDS** standards. In this situation, the assessment takes into account host and network characteristics. On the one hand, host-related attributes are utilized to measure the *Performance Overhead* of running the **A-HIDS** in the Edge device by measuring **CPU** and **RAM** consumption. On the other hand, network-based attributes are used to analyze the solution's suitability to handle network data by measuring network overhead.

This assessment technique takes into account the execution of the **A-HIDS** on one Edge device (`pi@192.168.0.20`), using the experimental setup given in Section 7.3.3. The purpose of this approach is to take into account the *Performance Overhead*, such as **CPU** demanding, memory-intensive, and network-intensive. The *Performance Overhead* data is gathered with the help of the *nmon* tool [304], a well-known monitor program for computer performance in Linux-based platforms. Thus, the test scenarios consider three different monitoring moments:

- (I) At system ramp-up and while no operation is being performed, which means little workload (processes, network connections, etc.);

- (II) System in operation, by executing the underlying tasks, such as sensor simulation, publishing and subscribing sensor data, as mentioned in Section 7.3.3, but without executing the A-HIDS;
- (III) Executing the A-HIDS while the Edge device is under normal operation, which also includes attack injection.

The monitoring procedure was carried out in all test scenarios for 60 minutes of system operation. Table 7.16 highlights the monitoring results obtained using the previously mentioned assessment approach.

Table 7.16: Performance overhead evaluation methodology results.

Test Scenario	(I)	(II)	(III)
Avg. CPU (%)	0.6%	3.2%	31.2%
Max. CPU (%)	9.1%	8.1%	34.9%
Avg. Free Memory (MB)	1 432.6	953.4	958.5
Min. Free Memory (MB)	1 380.6	887.1	845.5
Avg. Network I/O (KB/s)	0.19	13.75	297.6
Max. Network I/O (KB/s)	24.7	34.1	425.7

Based on the results, one can draw some conclusions about the applicability of the M2M scenario created for this unique experimental setup-test scenario (II), as well as the use of the A-HIDS in this distributed CPPS-test scenario (III). First, it is evident in test scenario (II) that adopting the M2M scenario will increase CPU use and network traffic while decreasing free system memory. However, the effect is minor, since just 3.2 percent of the CPU is dedicated to the underlying activities, leaving nearly 1GB of spare RAM (considering the 2GB of memory of the *Raspberry Pi 4 Model B*, which represents almost 77 percent of free memory). Also, the overall network traffic (read and write) in all the interfaces (local, wireless and wired) is under 14 KB/s.

On the other side, considering test scenario (III), the impact is more evident when the A-HIDS is used to process the network traffic in the M2M scenario considered. First, the CPU consumption increases by 10x to 31% CPU allocation. Despite this significant CPU consumption increase, there is no apparent negative impact on this system, considering almost 70% CPU free to be allocated to other processes. Also, a small impact is verified in the RAM since the free memory is reduced to 51%. Secondly, the overall network traffic increases significantly, to almost 300 KB/s, mainly due to the increase in attack injection traffic in the local interface. However, despite the increase in the network traffic, the rate of missing packet classification is relatively low. Less than 0.1% of the overall packets weren't classified.

7.3.4.2 FB Pipeline Measures

In this assessment methodology, we perform validation and analysis of both the FB Pipelines of the A-HIDS and the CPPS testbed. For this, we used adapted measures from software, presented

earlier by Zhabelova and Vyatkin [63] (see Section 2.3.3) and already used by Tanveer *et al.* [295] when assessing an industrial security solution proposal.

The deployment of the A-HIDS Pipeline in the CPPS testbed specified in Section 7.3.3 is considered in this assessment methodology. The major purpose of this methodology is to analyze the *Design Complexity* and *Maintainability* of the standalone A-HIDS Pipeline, as well as the FB Pipeline of the experimental setup (CPPS Testbed) with and without the A-HIDS FBs, i.e., executing or not the proposed solution. It is also proposed to compare the *Design Complexity* and *Maintainability* metrics obtained in this study to similar studies in the literature. Tanveer *et al.* [295] is the sole work in this case that addresses IEC 61499-based security solutions and uses these metrics for assessment. The authors suggest many security solutions based on TLS⁵⁵ and AEAD⁵⁶ techniques and demonstrate them in two case studies: an IMCS⁵⁷ case study and a major BHS⁵⁸ application. The following test scenarios are being considered:

- (I) Security Approaches Design: Assess the S^* ⁵⁹, DC^* ⁶⁰, C^* ⁶¹ and MI ⁶² of three different security approaches proposed, namely: a) A-HIDS (proposed approach in this study); b) Average results of the *Secure Links* approach [295], which is a set of security mechanisms; and c) *Secure Links* best result mechanism proposed in the same work.
- (II) CPPS Testbed Impact: Assess the impact in terms of complexity and *Maintainability* that the application of the A-HIDS has when deployed in the CPPS Testbed. This is done by comparing the assessment of S^* , DC^* , C^* and MI of both: a) CPPS Testbed standalone; and b) CPPS Testbed while executing the A-HIDS.
- (III) IMCS Impact: Assess the S^* , DC^* , C^* and MI impact of the *Secure Links* application to the IMCS testbed. Comparison is done between: a) IMCS; and b) IMCS with *Secure Links*.
- (IV) Same as the previous one, but this time the testbed considered is the BHS instead of the IMCS: a) BHS; and b) BHS with *Secure Links*.

Based on these test scenarios, Table 7.17 summarizes and contrasts the results obtained in this study and in the work of Tanveer *et al.* [295], taking into account the previously established assessment approach.

Considering the findings of test scenario (I), the A-HIDS performs well in terms of *Data Complexity* (0.65). When compared to the average results of the *Secure Links* method, it is evident that the A-HIDS is a better solution in terms of complexity and produces extremely competitive results when compared to the top result of the *Secure Links* approach. In addition, when it comes to the *Maintainability Index* metric, the A-HIDS performs admirably (100.12), outperforming both

⁵⁵Transport Layer Security

⁵⁶Authenticated Encryption with Associated Data

⁵⁷Industrial Mixer Control System

⁵⁸Baggage Handling System

⁵⁹Structural Complexity

⁶⁰Data Complexity

⁶¹System Complexity

⁶²Maintainability Index

Table 7.17: Evaluation methodology results using *Function Block* Pipeline measures.

Test Scenarios		Complexity						Maintainability	
		<i>S*</i>	%	<i>DC*</i>	%	<i>C*</i>	%	<i>MI</i>	%
(I)	A-HIDS	16	-	0.65	-	16.65	-	100.12	-
	<i>Secure Links</i>	23.8	-	20.6	-	44.4	-	75.68	-
	(Average)								
	<i>Secure Links</i>	8	-	3.8	-	11.8	-	85.76	-
	(Best)								
(II)	CPPS Testbed	1 444		0.04		1 444.4		124.93	
	CPPS Testbed with A-HIDS	1 764	22.2%	0.04	0%	1 764.04	22.2%	95.49	23.6%
(III)	IMCS	8		29.6		37.6		93.51	
	IMCS with <i>Secure Links</i>	8	0%	41.6	40.5%	49.6	31.91%	89.31	4.5%
(IV)	BHS	196		37		233		42.19	
	BHS with <i>Secure Links</i>	196	0%	121	227%	317	36.1%	42.07	0.3%

the average and best results of *Secure Links*. Overall, although not being the same type of security solution, the **A-HIDS** is a more manageable and low-complexity **FB** application solution when compared to existing work in literature.

In terms of the influence of the **A-HIDS** when deployed in the **CPPS** Testbed (II), the **IDS** has an undeniable impact on the testbed. In terms of system complexity and *Maintainability Index*, there is a slight rise in complexity (22.16 percent) and a reduction in *Maintainability Index* (23.57 percent). Using the **A-HIDS**, on the other hand, does not influence data complexity. As implementing the **A-HIDS** in the **CPPS** Testbed, the increase in system complexity by 22.16 percent is shorter in (III) and (IV) when compared to the test scenarios (III) and (IV). This suggests that the proposed approach has a lower negative impact in terms of complexity when compared to the adoption of *Secure Links* in both **IMCS** and **BHS**. However, the impact on system *Maintainability Index* is more significant (23.57 percent when *Secure Links* implementation is less than 5%).

In conclusion, as compared to *Secure Links*, **A-HIDS** is a more maintainable solution with lower complexity when compared to independent security solutions. Considering the influence of solution deployment in testbeds, the **A-HIDS** performs well in terms of complexity but poorly in terms of *Maintainability Index*. This is due mainly to the **A-HIDS**' increased program volume, as well as its control and data flow complexity.

7.3.5 Discussion

The majority of the requirements specified by Vargas Martinez and Vogel-Heuser [70] (see the beginning of Section 7.3.2) were taken into account in the design, implementation, and assessment of the suggested solution. Concerning the solution's *Configurability*, the suggested A-HIDS may be parametrized by modifying the data-input fields of the *iDCA* FB. The data model, i.e., the network properties examined, are, however, derived internally in the *OPCUA_Flow_Sniffer*. This implies that the user cannot define or add new features to be used unless the FB source code is updated. The A-HIDS *Configuration and knowledge security*, on the other hand, do not safeguard the data model, data-input fields, and gathered data for analysis. This need was not taken into account when developing the solution.

Given the solution's *Resiliency*, all of the threats studied in this study, notably DoS, MITM, and Message Spoofing, exploit the OPC UA communication channel in an M2M scenario. Because the A-HIDS is running at the Edge device level, it should be safe from OPC UA-related threats. However, there may be other sorts of attacks that target the Edge device, notably the A-HIDS, by exploiting various vulnerabilities. These are not taken into account or tested in this study.

In terms of *Performance Overhead* and *Detection Duration*, the results in Table 7.16 show that there is a typical rise in CPU consumption, RAM memory utilization, and network traffic during A-HIDS execution. However, there is no evidence of negative impact. The A-HIDS do not affect the performance of the Edge device's underlying tasks. Furthermore, the A-HIDS employs the *iDCA* as a detection approach, which gradually analyzes input data and allows real-time detection in typical stream data. In this manner, intrusions may be detected and responded to quickly.

The 4DIAC-IDE, which implements the IEC 61499 standard, may be used for the *configuration & management* of the CPPS and A-HIDS. It is feasible to monitor and control the distributed system by creating and installing the A-HIDS utilizing DINASORE. According to the results in Table 7.17, the A-HIDS Pipeline is a simple solution to maintain with low overall complexity for configuration and maintenance. Regarding *System audit*, on the other hand, the A-HIDS analyzes network traffic (input and output the OPC UA network traffic in the Edge device). It also collects host-related data to examine approach performance on the host device, such as CPU, RAM, and network traffic overhead. However, considering the *Monitoring analysis* need, host-related data generated for performance purposes is not treated as input in the A-HIDS for detection purposes. This is outside the scope of this work.

Concerning *passive intrusion detection actions*, the A-HIDS log and alert are both feasible. When an intrusion is identified, the A-HIDS logs the relevant network activity as a suspected intrusion, and the source machine's IP⁶³ address is added to a timestamped blacklist. In terms of *Log collection*, in addition to being analyzed for real-time detection, the gathered network data is saved in a log file for future offline analysis. In this scenario, the network data is recorded in a normal ML⁶⁴ tabular format dataset utilizing the data model as features evaluated in the A-HIDS

⁶³Internet Protocol

⁶⁴Machine Learning

using a [CSV⁶⁵](#) file type. This is highly useful since the dataset obtained may be utilized later for validation in other [IDS](#) that employ various [ML](#) approaches for detection. Finally, *Interoperability with other security tools* in the system for overall security evaluation is beyond the scope of this work.

7.4 Conclusion

This section describes two sets of experiments and achieved results.

In **E1. *iDCA* vs. *dDCA***, we describe a validation methodology for both [dDCA](#) and [iDCA](#). On the one hand, we want to understand if the [dDCA](#) is suitable to be used in intrusion detection problems, considering different contexts. On the other hand, we want to assess the performance of the [iDCA](#) when compared to the [dDCA](#). Since the [iDCA](#) was designed for online scenarios, and we want to use labeled datasets to infer performance results, all data point instances for each dataset become available in a sequential manner. In reality, this implies that all rows in the dataset are retrieved and made accessible sequentially until the end of the dataset file. This way, it is possible to mimic an online data processing scenario. This is the opposite of the [dDCA](#) where each dataset is provided as input to the algorithm in a typical batch format.

To answer the research question **RQ T.7** *What impact in the Dendritic Cell Algorithm's detection rate and accuracy has an incremental learning approach?*, results show improvements in the detection rate when using the [iDCA](#) since all the stages of the algorithm are performed every time a new data point is available to be classified. The [iDCA](#) also presents slightly worse classification accuracy when compared to the [dDCA](#), while dropping more data entries. However, the results are quite satisfactory. This trade-off between high detection accuracy while having fast detection rates and a low number of non-classifications can be represented in a unique metric, designated *Global System Performance*. When inspecting the *Global System Performance*, the [iDCA](#) tend to be a very similar (or even better in some cases) [IDS](#) detection approach when compared to [dDCA](#).

Moreover, to answer the research question **RQ T.8** *What is the cause-effect relationship between algorithm variables, runtime properties and detection performance in Dendritic Cell Algorithm-based intrusion detection?*, we conclude that increasing the cell population size and number of iterations will increase the number of migrations, which will result in a lower detection rate. On the other hand, the classification rate increases since there are more migrations. Moreover, if the cell has an increased lifespan, the migration process will take longer, reducing the number of migrations. Consequently, the detection rate increases, but the number of non-classifications goes up. We also find that a two feature approach for signal mapping also increases the number of migrated cells.

As for the **E2. *A-HIDS Pipeline***, we consider the premises of the [MBE](#) technique as an alternative strategy for building [CPPSs](#), since it employs models to create software and do component testing, which accommodates the complexity and dynamism of [CPPSs](#). Thus, we describe the methodology to validate an [iDCA](#)-based [IDS](#), used to identify real-time network assaults in [CPPSs](#)

⁶⁵Comma Separated Values

by analyzing incoming/outgoing **OPC UA** network packets in the Edge device. We consider several metrics in two system characteristics, namely *Performance Overhead*, *Design Complexity* and *Maintainability Index*.

To answer the research question **RQ T.9** *How effective is IEC 61499 in supporting the implementation of low complexity and high maintainability security-by-design Cyber-Physical Production System?*, our findings demonstrate that the **IEC 61499** standard is a reliable method to enabling the implementation of security by design **CPPSs**. The standard permits security by design, considering the low-complexity and high-maintenance outcomes of various security methods, including the **iDCA**. On the other hand, the **A-HIDS** Pipeline developed delivers improved maintainability and overall reduced complexity as compared to state-of-the-art work (see in Table 7.17). Thus, we can conclude that enabling bio-inspired network intrusion detection by design in **CPPSs** using the **IEC 61499** standard is not only possible but preferable when compared with classical software development methodologies.

Chapter 8

Conclusions

Historically, **ICS**¹ components were specific embedded devices connected with a proprietary communication protocol. Typically, these components were incompatible with competitor systems. Vendors would fail to include security by design since real-time processing and task execution robustness were the primary concern. Despite the lack of security features, manufacturing companies believed that their **ICS**s were protected, thanks to two assumptions. First, because these systems were physically connected to isolated field networks, access from other corporate **IP**² networks was not possible. Thus, an external attacker would not access these systems (or at least the job would be tough). Second, because the information publicly available about these systems was limited, an attacker would not understand how to explore vulnerabilities in these environments.

However, with the **I4.0**³ paradigm, these systems are being transformed into modern **CPPS**s⁴. The main trends in these new setups are: i) increased connectivity using **IP** oriented networks, which allows communication with corporate networks and remote access from engineers and administrators to the field devices through the internet; ii) use of Edge/**IoT**⁵ devices, which most of the times are low-cost **COTS**⁶ devices, characterized by limited computational resources.

Given the critical nature of infrastructures that employ **CPPS**s, criminal organizations and malicious individuals may find it valuable to target cyberattacks to these types of systems. Facing the scenario described, it is natural that more advanced security practices are necessary for **CPPS**s. These systems are still relatively isolated due to the use of firewalls. Manufacturing organizations make an effort to fix all the system's vulnerabilities while using active defense strategies, such as access control and authentication. However, it is technically infeasible to fix all flaws and vulnerabilities in the system. Also, vulnerabilities can be caused by configuration errors or insiders abuse. Thus, a second line of defense is needed, which may include an **IDS**⁷. Particularly, these

¹*Industrial Control Systems*

²*Internet Protocol*

³*Industry 4.0*

⁴*Cyber-Physical Production Systems*

⁵*Internet of Things*

⁶*Commercial Off-The-Shelf*

⁷*Intrusion Detection System*

approaches should comply with the autonomous behavior of CPPSs, capable of adapting their detection strategy.

8.1 Main Achievements

In this Thesis, we addressed the problem of bio-inspired intrusion detection in CPPSs. Thus, we formulated the following research question:

In what ways can the Dendritic Cell Algorithm be used for anomaly intrusion detection in Cyber-Physical Production Systems?

This research question is further decomposed into two subsets of research questions, where the first consists of qualitative and the last of quantitative research questions.

The qualitative research questions enabled the development of a theoretical baseline regarding the application of bio-inspired approaches, specifically AISs⁸ in SMSs⁹. They are:

RQ T.1 *What is the research trend of Artificial Immune System applications in Smart Manufacturing Systems?*

RQ T.2 *What is the status of Artificial Immune System applications in Smart Manufacturing Systems?*

RQ T.3 *What are the main motivations for using Artificial Immune Systems in Smart Manufacturing Systems?*

RQ T.4 *What are the research gaps and future areas within Smart Manufacturing Systems based on Artificial Immune System solutions?*

We showed that AIS applications in manufacturing scenarios are attracting more and more attention from the academic community, in research areas such as *Autonomic Systems, Control Systems, Islanding, Intrusion Detection System, Anomaly Detection, Fault Diagnosis and Operations Research*. Typically the fields of applications are *Electric Power Industry, Supervisory Control and Data Acquisition/Industrial Control Systems, Manufacturing, Robotics and Cyber-Physical Production System*.

The main motivation of using AISs in SMSs is related to the achievement of SMS requirements, such as *Data Analytics, Autonomy, Adaptability, Optimized Management, and Process Efficiency & Product Quality*. In this context, we found that AIS-oriented solutions are reported to be good approaches to:

- Handle big amounts of data and perform well-defined analysis
- Generate Self-behaving systems and system adaptability to changes

⁸Artificial Immune Systems

⁹Smart Manufacturing Systems

- Management of very complex systems
- Agent-oriented architecture for dynamic, heterogeneous and distributed environments

Overall, the usage of immune models in SMSs is motivated by the flexible decision-making mechanism provided by such solutions, which suits the dynamic and complex nature of SMS environments. However, we also found research gaps, related with important SMS requirements, such as *Safety & Security, Resource Efficiency, Communication Network Infrastructure, Standardization and Human Management, Regulatory Framework and Upskilling*.

On the other hand, the quantitative research questions are framed in the *Safety & Security* research gap. We intended to study the relationships among different variables by addressing two different perspectives. First, we study the autonomous nature of an IDS solution by exploring bio-inspired solutions, such as AIS. Second, we study the feasibility of employing such solutions by design in Edge devices. To the best of our knowledge, this Thesis provides the first study about developing strategies to embed in industrial Edge devices bio-inspired intrusion detection approaches. The research questions are:

RQ T.5 *What are the primary strengths and weaknesses when developing Dendritic Cell Algorithm-based solutions for classification problems?*

RQ T.6 *What research strategies can be adopted for addressing opportunities and threats in industrial anomaly detection?*

RQ T.7 *What impact in the Dendritic Cell Algorithm's detection rate and accuracy has an incremental learning approach?*

RQ T.8 *What is the cause-effect relationship between algorithm variables, runtime properties and detection performance in Dendritic Cell Algorithm-based intrusion detection?*

RQ T.9 *How effective is IEC 61499 in supporting the implementation of low complexity and high maintainability security by design Cyber-Physical Production Systems?*

Regarding **RQ T.5**, we provide a comprehensive analysis of the DCA¹⁰ and identify the main beneficial features and weaknesses of using the DCA in supervised anomaly detection or binary classification problems. We found out the significant motivations and the main weaknesses of using DCA in such problems are:

- ✓ The algorithmic linear nature of the DCA makes it very lightweight.
- ✓ No training phase is required.
- ✓ Capable to operate in the presence of dynamically changing environments.

¹⁰Dendritic Cell Algorithm

- ✓ The **DCA** provides temporal correlation, achieving cause-effect relationships between anomalies and potential causes.
- ✓ The algorithm enables Self-Immunity to the protected systems, i.e., long-run adaptive protection based on continuous learning.
- ✗ The **DCA** has a strong dependency on the problem domain, thus relying greatly on a manual pre-processing phase.
- ✗ The algorithm is stochastic, i.e., several parameter variations can lead to different results.
- ✗ The **DCA** is not suitable to perform online classification in typical **IoT** data streams.
- ✗ The linear nature of the algorithm may restrict its application in complex problems.
- ✗ The **DCA** lacks a formal analysis and definition for **OOP**¹¹.

Based on the **DCA** strengths and weaknesses, and considering the context of the intrusion detection problem in industrial scenarios in terms of opportunities and threats, we identify the research gaps and characterize the strategies to take advantage of the opportunities and avoid threats. To answer the **RQ T.6**, we provide a **SWOT**¹² analysis to understand precisely what is the state of the art and what are the opportunities for further scientific contribution. The main research gaps/strategies found are:

- a) **DCA**-based **IDS** in **IIoT**¹³ environments;
- b) **CPPS** security by design using **IEC**¹⁴ 61499;
- c) **DCA** formal analysis and definition;
- d) **DCA** online detection;
- e) **NIDS**¹⁵ **OPC UA**¹⁶-oriented dataset;
- f) **DCA** automatic pre-processing stage.

Regarding **RQ T.7**, we develop a novel variant of the **DCA**, designated **iDCA**¹⁷, that includes incremental learning approaches for the collection and processing of **OPC UA** network data in an online fashion. We showed, after validating the approach in four different datasets, that the **iDCA** improves the *Detection Rate* since all the stages of the algorithm are performed every time a new data point is available to be classified. Overall, despite satisfactory, the **iDCA** presents slightly worse *Classification Accuracy* when compared to the **dDCA**¹⁸, while dropping more data entries.

¹¹ *Object-Oriented Programming*

¹² *Strengths, Weaknesses, Opportunities and Threats*

¹³ *Industrial Internet of Things*

¹⁴ *International Electrotechnical Commission*

¹⁵ *Network-based Intrusion Detection System*

¹⁶ *OPC Unified Architecture*

¹⁷ *Incremental Dendritic Cell Algorithm*

¹⁸ *deterministic Dendritic Cell Algorithm*

To capture in a unique metric the *Global System Performance*, we relate the *AUC-ROC*¹⁹, *Detection Rate* and *Classification Rate* by using a weighted mean. When inspecting the *Global System Performance*, the *iDCA* tend to be a very similar (or even better in some cases) *IDS* detection approach when compared to *dDCA*.

As for **RQ T.8**, we provide a comprehensive analysis regarding the application of both *dDCA* and *iDCA* to the intrusion detection problem. For this, we defined several test cases, consisting of the combination of the different input parameters and pre-processing procedures, such as feature selection, *Dendritic Cell* lifespan, number of iterations, population size, and the anomaly threshold. Initial tuning of these parameters significantly affect the algorithm runtime properties (e.g., number of migrated cells), which in turn, will have an impact on the *Global System Performance*.

Overall, we conclude that increasing the cell population size and number of iterations will increase the number of migrations, which will result in a slower detection speed. On the other hand, the classification rate increases since there are more migrations. Moreover, if the cell has an increased lifespan, the migration process will take longer, reducing the number of migrations. Consequently, the detection rate increases, but the number of non-classifications goes up. We also find that a two feature approach for signal mapping also increases the number of migrated cells.

Moreover, we concluded that the best parameter combination is not the same for *dDCA* and *iDCA*. The behavior of the algorithms in different datasets is not the same either, i.e., a good combination parameter for one problem context may not hold for a different one. Regarding *dDCA*, the best parameter combination follows a *Two-Feature Approach*, $t_m^{20} = 10\ 000$, $\#Dc^{21}$ and $\#Iter^{22} = 1$. The At^{23} can be the default or the derived. As for the *iDCA*, the best parameter combination uses $t_m = 100$, $\#Dc = 100$ and $\#Iter = 10$. The At should be the default 0.5. Any signal mapping approach works well.

Finally, considering **RQ T.9**, we develop a *FB*²⁴ architecture, based on the *IEC 61499* standard, for the *iDCA* deployment in a *CPPS* in early design stages. We showed that the *IEC 61499* standard is a reliable method to enabling the implementation of security by design *CPPS*s. Also, by following a *MBE*²⁵ development approach, we proved that the *A-HIDS*²⁶ Pipeline, using the *iDCA* for detection purposes, is a good software implementation in terms of *Complexity* and *Maintainability*. Compared with similar approaches, the proposed *A-HIDS* Pipeline presents overall lower *Complexity* and higher *Maintainability*.

¹⁹Area Under the Receiver Operating Characteristic Curve

²⁰Migration Threshold

²¹Number of Dendritic Cells

²²Number of Iterations

²³Anomaly Threshold

²⁴Function Block

²⁵Model-based Engineering

²⁶Anomaly Host Intrusion Detection System

8.2 Main Outputs & Impact

Given the work developed in this Thesis, in an effort to find answers to the previous research questions, we identify the following outcomes of the Thesis:

- Conceptual framework about the implementation of AIS in SMS (Appendix B). Based on a SLR²⁷ on AISs applications used in SMSs (Chapter 4), this conceptual framework offers to manufacturers guidance for using AISs in their manufacturing applications to tackle various challenges. The main impact will be the speed-up of industrial adoption of the AIS technology, thus accomplishing the complete autonomy goal in SMSs faster.
- SWOT analysis of DCA-based approaches for intrusion detection in industrial scenarios (Appendix E). This analysis enables the identification of research gaps and strategies to contribute to the SMS and AIS fields. With this analysis, Scientists and Engineers can understand what the Strengths and Weaknesses of DCA are and how to use this immune approach to explore the Opportunities and avoid potential Threats considering the industrial intrusion detection problem.
- *M2M using OPC UA* [34] dataset (Section 6.2 in Chapter 6). This dataset emerged by the need to have publicly available industrial dataset and benchmark to validate NIDSs applied in the industrial context. Scientists and Engineers can evaluate the effectiveness of different intrusion detection techniques and overall classifiers in a manufacturing context.
- iDCA (Section 6.3 in Chapter 6). This novel variant of the dDCA is oriented for online network-based intrusion detection. Thus, the bio-inspired algorithm is a fast and cheap approach to detecting network intrusions, suitable for executing in low-computing resourced devices. The main advantages are the capability of online detection and the intrusion detection speed, which is a critical need in industrial scenarios (faster the detection, faster the intrusion mitigation and system recovery, to minimize the attack impact on the system).
- Comprehensive analysis of dDCA and iDCA in intrusion detection problems (Section 7.2 in Chapter 7). This empirical analysis considers multiple combinations of different parameters to assess the runtime properties of the algorithms and the detection/classification quality. This contribution to the AIS research field may foster Scientists and Engineers' involvement in the field as an emergent alternative to typical ML²⁸, and *Hard Computing* approaches.
- A-HIDS Pipeline, based on the iDCA detection technique (Section 6.4 in Chapter 6). This MBE methodology, compliant with the IEC 61499, can be used by Engineers and I4.0 Professionals to enable bio-inspired intrusion detection by design during the early stages of CPPS development. This low-complexity approach is much easier to maintain and avoids costly and error-prone code development.

²⁷Systematic Literature Review

²⁸Machine Learning

8.3 Limitations

Regarding the qualitative research questions, the study resulted in a comprehensive characterization of AIS applications in SMSs by following the PRISMA²⁹ guidelines. However, there are some limitations when conducting the review, mostly related to the exclusion criteria used, which resulted in possible relevant publications being left out of the study. Some of these limitations include excluding non-English documents and the unavailability to access the documents' full text. Moreover, several publications were excluded because they did not present an industrial application, but the work was framed within related research fields. Finally, several papers were found presenting AISs work with industrial applications, but no results were shown to support the claims, resulting in their exclusion.

On the other hand, considering the quantitative questions, the study resulted in the development of the iDCA compliant with the IEC 61499 standard and its application in CPPSs. The report on the findings of the feasibility of the iDCA for online intrusion detection shows that an incremental learning approach enables high detection speed in exchange for a slight underperformance in the detection accuracy. The main limitation is the strong dependency on the pre-processing and initialization stage. We provided a comprehensive testing methodology but failed to cover all the parametrizations possible for the iDCA regarding feature selection, signal input mapping, *Dendritic Cell* population size, number of iterations, migration, and anomaly thresholds. Also, the technique was validated solely in datasets, most of them not industry-oriented. So, the lack of validation in a real-world test case by processing real-time network data may leave out essential characteristics of the behavior of such networks and non-linear dynamics, possibly resulting in poorer detection results.

The report on the findings of the impact of the iDCA on the computational resources of the Edge layer of the CPPS shows no evidence of the negative effect of the iDCA when executing at the host level in constraint Edge devices. Results show that the iDCA can detect intrusions in an online fashion with a reduced overhead in CPU³⁰ consumption and RAM³¹ usage. The main limitation was the failure of considering the distributed nature of the CPPS and the close relationship with process-driven data. Considering only network data individually in each Edge device may limit the detection performance results. A fully distributed system would support detection decisions based on social engineering among all device nodes in the CPPS. Also, while not considering other industrial devices such as PLCs³², testing the approach only in resource constraining COTS devices may result in neglecting important performance characteristics related to resource consumption and process overhead in industrial tasks-related.

Finally, the report on the findings of the IEC 61499 design effectiveness for implementation of an A-HIDS shows successful implementation of security by design CPPS. The FB-oriented detection approach achieves better *Maintainability* and overall low *Complexity* when compared with

²⁹Preferred Reporting Items for Systematic Review and Meta-Analysis

³⁰Central Processing Unit

³¹Random Access Memory

³²Programmable Logic Controllers

similar work in the literature. As the main limitation, we failed to consider in the proposed solution necessary **A-HIDS** requirements, such as full configurability, knowledge security, resiliency to other non-**OPC UA**-driven formats of attack and interoperability with other **ICT**³³ security tools.

8.4 Future Work

Inspired by the contributions provided in this Thesis, there are interesting open issues worth investigating in the future.

First, future work may consider the distributed nature of the **CPPS** and the close relationship between the physical and virtual worlds to enhance detection performance. As stated before, one of the most neglected security threats in the industrial domain is related to inside attacks and malicious users who already have privileged access to field networks. Considering that attacks in **CPPS**s aim to impact the production process and restrict (fully or partially) the functionality of physical assets, a **CPPS** would benefit to process host-related data and merge it with network-based data for a meaningful detection approach. Also, a genuinely decentralized detection approach enables a global self-awareness of the entire network to be protected instead of only local perspectives in each device.

Second, the **A-HIDS** should be deployed and validated in real-world industrial scenarios, where Edge devices may have different constraints related to the production process. In a lab **CPPS** testbed, we can simulate task execution and automatic communication among different Edge devices. However, we cannot meet all the possible dynamics during a real-world production process. Also, despite presenting good results for *Complexity* and *Maintainability*, this **A-HIDS** may not be suitable to be used in every use case in a real-world industrial scenario.

Finally, the proposed solution may include in the future a more active reaction to a detected attack/intrusion, which may consist of a dynamic reconfiguration and deployment of the **CPPS**, following the **IEC** 61499 standard. This includes an autonomous reconfiguration of the Edge devices involved in the production process to deal with the logical isolation of the devices impacted by an attack. This way, due to the logical reconfiguration, it is possible to keep part of the system unaltered and avoid total system collapse. This way, by enabling detection and recovery, we could achieve a genuinely Self-Immune System since we are continuously shaping the system's protection and defense. Also, since the **A-HIDS** learns over time, the lessons acquired from previous attack detection and recovery strategy could be used for faster prevention next time the same malicious action occurs.

The contributions and main outcomes of this Thesis are currently being used in an R&D project with an industrial partner, namely *Continental Advanced Antenna Portugal*. The project designation is *Continental FoF* - Continental AA's Factory of the Future (code: POCI-01-0247-FEDER-047512).

Continental Advanced Antenna Portugal, located in Vila Real, is one of Europe's leading vehicle antenna specialists and manufacturers. It produces around 18 million antennas a year,

³³Information and Communication Technology

exported all over the world. With around 550 employees, *Continental Advanced Antenna Portugal* is one of the largest private employers in the district of Vila Real. The company belongs to the *Vehicle Networking Information* business area, is part of the *Connected Car Networking* business unit, and works essentially with the premium segment of brands in the automotive industry. The Continental group is present in 60 countries and employs more than 240 000 people.

Framed in the *Continental FoF* activities, we aim to contribute to the advancement of the industry and the technical-scientific community (at national and international levels) by developing new solutions for CPPS and the fully digital transformation of the factory (IIoE³⁴). We are fostering the study and implementation of all the current limitations mentioned within a *Continental FoF* test case scenario. With this, we want to achieve dynamic real-time data collection and classification (network and host) while enabling autonomous and intelligent decision-making for an active immune-based response when anomalies are found. Ultimately, we want to deploy an improvement of our proposed approach to enable Self-Immunity in Continental AA's Factory of the Future.

³⁴*Industrial Internet of Everything*

Part III

Appendix & Bibliography

Appendix A

Database Search Inputs

1. IEEE Xplore:

```
("Document Title":"Artificial Immune Systems"  
OR "Document Title":"AIS" OR "Document Title":  
"Dendritic Cell Algorithm" OR "Document Title":  
"DCA") AND ("Document Title":"Cyber-Physical System"  
OR "Document Title":"Cyber-Physical Systems" OR  
"Document Title":"CPS" OR "Document Title":  
"Cyber-Physical Production System" OR "Document Title":  
"Cyber-Physical Production Systems" OR  
"Document Title":"CPPS" OR "Document Title":  
"Industrial Cyber-Physical System" OR "Document Title":  
"Industrial Cyber-Physical Systems" OR "Document Title":  
"Supervisory Control and Data Acquisition" OR  
"Document Title":"SCADA" OR "Document Title":  
"Distributed Control System" OR "Document Title":  
"Distributed Control Systems" OR "Document Title":  
"Industrial Internet of Things" OR "Document Title":  
"IIoT" OR "Document Title":  
"Internet of Industrial Things") OR ("Abstract":  
"Artificial Immune Systems" OR "Abstract":"AIS" OR  
"Abstract":"Dendritic Cell Algorithm" OR "Abstract":  
"DCA") AND ("Abstract":"Cyber-Physical System" OR  
"Abstract":"Cyber-Physical Systems" OR  
"Abstract":"CPS" OR "Abstract":  
"Cyber-Physical Production System" OR "Abstract":  
"Cyber-Physical Production Systems" OR  
"Abstract":"CPPS" OR "Abstract":  
"Industrial Cyber-Physical System" OR "Abstract":
```

"Industrial Cyber-Physical Systems" OR "Abstract":
 "Supervisory Control and Data Acquisition" OR
 "Abstract": "SCADA" OR
 "Abstract": "Distributed Control System" OR "Abstract":
 "Distributed Control Systems" OR "Abstract":
 "Industrial Internet of Things" OR "Abstract":
 "IIoT" OR "Abstract": "Internet of Industrial Things")
 OR ("Index Terms": "Artificial Immune Systems" OR
 "Index Terms": "AIS" OR "Index Terms":
 "Dendritic Cell Algorithm" OR "Index Terms":
 "DCA") AND ("Index Terms": "Cyber-Physical System"
 OR "Index Terms": "Cyber-Physical Systems" OR
 "Index Terms": "CPS" OR "Index Terms":
 "Cyber-Physical Production System" OR
 "Index Terms": "Cyber-Physical Production Systems"
 OR "Index Terms": "CPPS" OR "Index Terms":
 "Industrial Cyber-Physical System" OR
 "Index Terms": "Industrial Cyber-Physical Systems"
 OR "Index Terms": "Supervisory Control and Data Acquisition"
 OR "Index Terms": "SCADA" OR "Index Terms":
 "Distributed Control System" OR "Index Terms":
 "Distributed Control Systems" OR "Index Terms":
 "Industrial Internet of Things" OR
 "Index Terms": "IIoT" OR "Index Terms":
 "Internet of Industrial Things")

2. Scopus:

(Artificial Immune Systems OR AIS OR
 Dendritic Cell Algorithm OR DCA) AND
 (Cyber-Physical System OR Cyber-Physical Systems OR
 CPS OR Cyber-Physical Production System OR
 Cyber-Physical Production Systems OR CPPS OR
 Industrial Cyber-Physical System OR
 Industrial Cyber-Physical Systems OR
 Supervisory Control and Data Acquisition OR SCADA OR
 Distributed Control System OR
 Distributed Control Systems OR
 Industrial Internet of Things OR IIoT OR
 Internet of Industrial Things)

3. ScienceDirect:

```
(“Artificial Immune Systems” OR “Dendritic Cell Algorithm”)
AND (“Cyber-Physical System” OR
“Cyber-Physical Production System” OR
“Industrial Cyber-Physical System” OR SCADA OR
“Distributed Control System” OR
“Industrial Internet of Things” OR IIoT)
```

4. Springer:

```
(Artificial Immune Systems OR AIS OR
Dendritic Cell Algorithm OR DCA) AND
(Cyber-Physical System OR Cyber-Physical Systems OR
CPS OR Cyber-Physical Production System OR
Cyber-Physical Production Systems OR CPPS OR
Industrial Cyber-Physical System OR
Industrial Cyber-Physical Systems OR
Supervisory Control and Data Acquisition OR
SCADA OR Distributed Control System OR
Distributed Control Systems OR
Industrial Internet of Things OR IIoT OR
Internet of Industrial Things)
```

5. ACM:

```
[[[Publication Title: "artificial immune systems"]
OR [Publication Title: "ais"] OR [Publication Title:
"dendritic cell algorithm"] OR [Publication Title: "dca"]]
AND [[Publication Title: "cyber-physical system"] OR
[Publication Title: "cyber-physical systems"] OR
[Publication Title: "cps"] OR [Publication Title:
"cyber-physical production system"] OR [Publication Title:
"cyber-physical production systems"] OR
[Publication Title: "cpps"] OR [Publication Title:
"industrial cyber-physical system"] OR
[Publication Title: "industrial cyber-physical systems"]
OR [Publication Title:
"supervisory control and data acquisition"] OR
[Publication Title: "scada"] OR [Publication Title:
"distributed control system"] OR [Publication Title:
```

```

"distributed control systems"] OR [Publication Title:
"industrial internet of things"] OR [Publication Title:
"iiot"] OR [Publication Title:
"internet of industrial things"]]] OR [[[Abstract:
"artificial immune systems"] OR [Abstract: "ais"] OR
[Abstract: "dendritic cell algorithm"] OR [Abstract:
"dca"]] AND [[Abstract: "cyber-physical system"] OR
[Abstract: "cyber-physical systems"] OR [Abstract:
"cps"] OR [Abstract: "cyber-physical production system"]
OR [Abstract: "cyber-physical production systems"] OR
[Abstract: "cpps"] OR [Abstract:
"industrial cyber-physical system"] OR [Abstract:
"industrial cyber-physical systems"] OR [Abstract:
"supervisory control and data acquisition"] OR
[Abstract: "scada"] OR [Abstract:
"distributed control system"] OR [Abstract:
"distributed control systems"] OR [Abstract:
"industrial internet of things"] OR [Abstract:
"iiot"] OR [Abstract: "internet of industrial things"]]]]
OR [[[Keywords: "artificial immune systems"] OR
[Keywords: "ais"] OR [Keywords:
"dendritic cell algorithm"] OR [Keywords: "dca"]]
AND [[Keywords: "cyber-physical system"] OR
[Keywords: "cyber-physical systems"] OR [Keywords:
"cps"] OR [Keywords: "cyber-physical production system"]
OR [Keywords: "cyber-physical production systems"]
OR [Keywords: "cpps"] OR [Keywords:
"industrial cyber-physical system"] OR [Keywords:
"industrial cyber-physical systems"] OR [Keywords:
"supervisory control and data acquisition"] OR
[Keywords: "scada"] OR [Keywords:
"distributed control system"] OR [Keywords:
"distributed control systems"] OR [Keywords:
"industrial internet of things"] OR [Keywords:
"iiot"] OR [Keywords: "internet of industrial things"]]]]

```

6. Engineering Village:

```

("Artificial Immune Systems" OR "AIS" OR
"Dendritic Cell Algorithm" OR "DCA") AND
("Cyber-Physical System" OR "Cyber-Physical Systems"

```

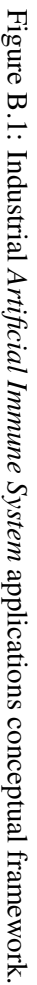
OR "CPS" OR "Cyber-Physical Production System" OR
 "Cyber-Physical Production Systems" OR "CPPS" OR
 "Industrial Cyber-Physical System" OR
 "Industrial Cyber-Physical Systems" OR
 "Supervisory Control and Data Acquisition" OR
 "SCADA" OR "Distributed Control System" OR
 "Distributed Control Systems" OR
 "Industrial Internet of Things" OR "IIoT" OR
 "Internet of Industrial Things")

7. ISI Web of Knowledge:

(ALL=("Artificial Immune Systems" OR "AIS" OR
 "Dendritic Cell Algorithm" OR "DCA")) AND
 ALL=("Cyber-Physical System" OR
 "Cyber-Physical Systems" OR "CPS" OR
 "Cyber-Physical Production System" OR
 "Cyber-Physical Production Systems" OR "CPPS"
 OR "Industrial Cyber-Physical System" OR
 "Industrial Cyber-Physical Systems" OR
 "Supervisory Control and Data Acquisition"
 OR "SCADA" OR "Distributed Control System"
 OR "Distributed Control Systems" OR
 "Industrial Internet of Things" OR "IIoT"
 OR "Internet of Industrial Things")

Appendix B

Industrial AIS Conceptual Framework



Appendix C

iDCA - System Modeling

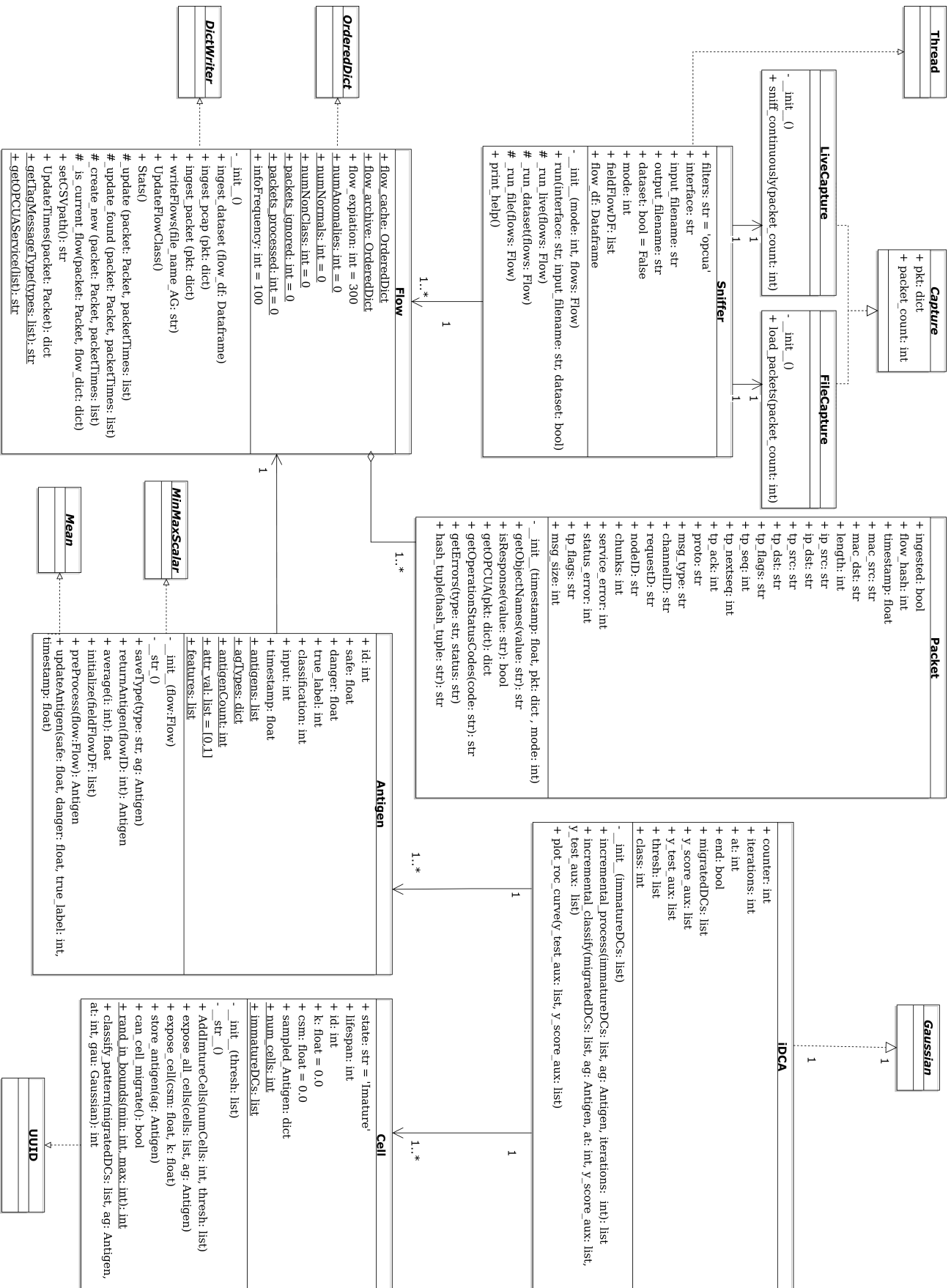
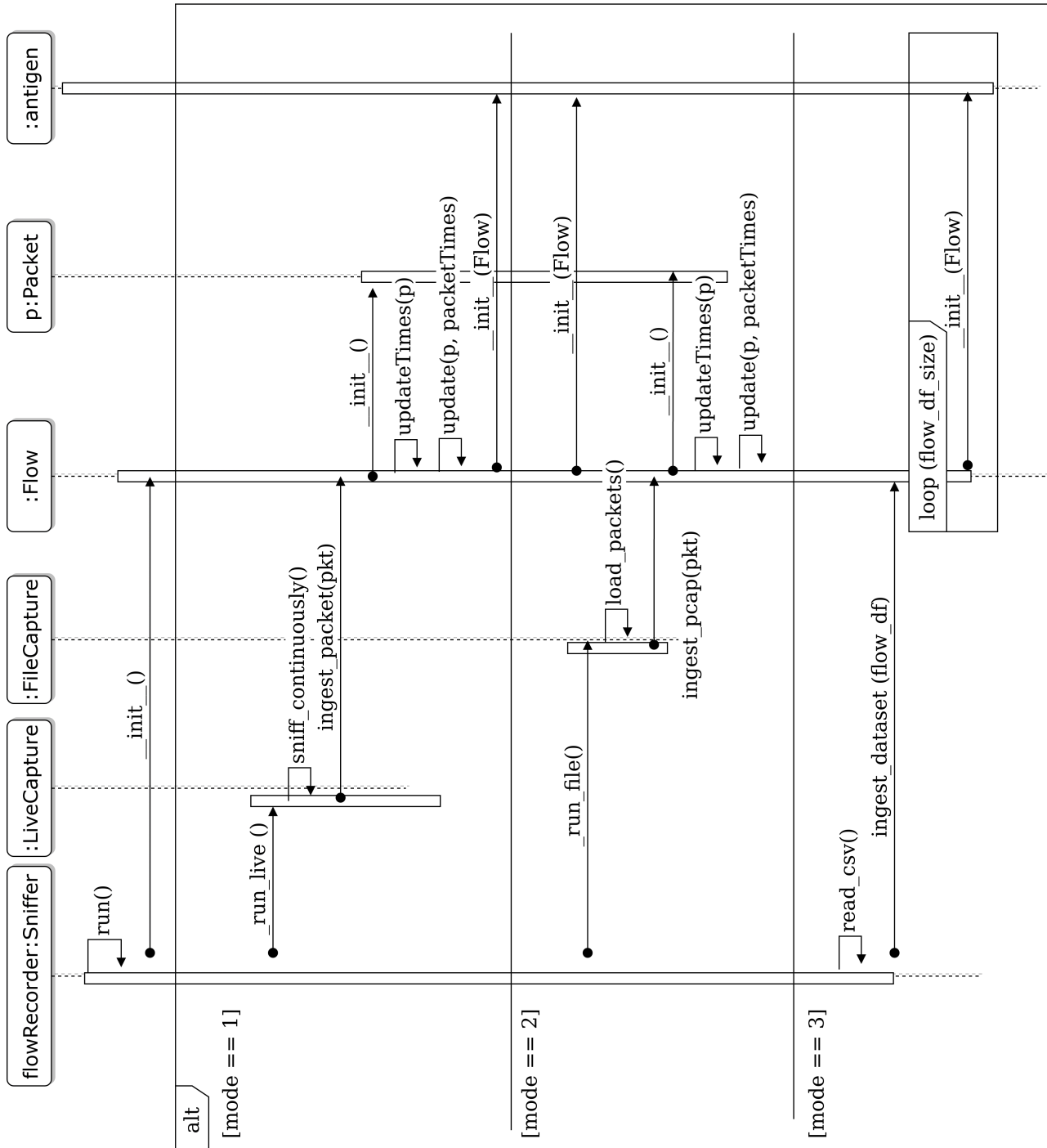
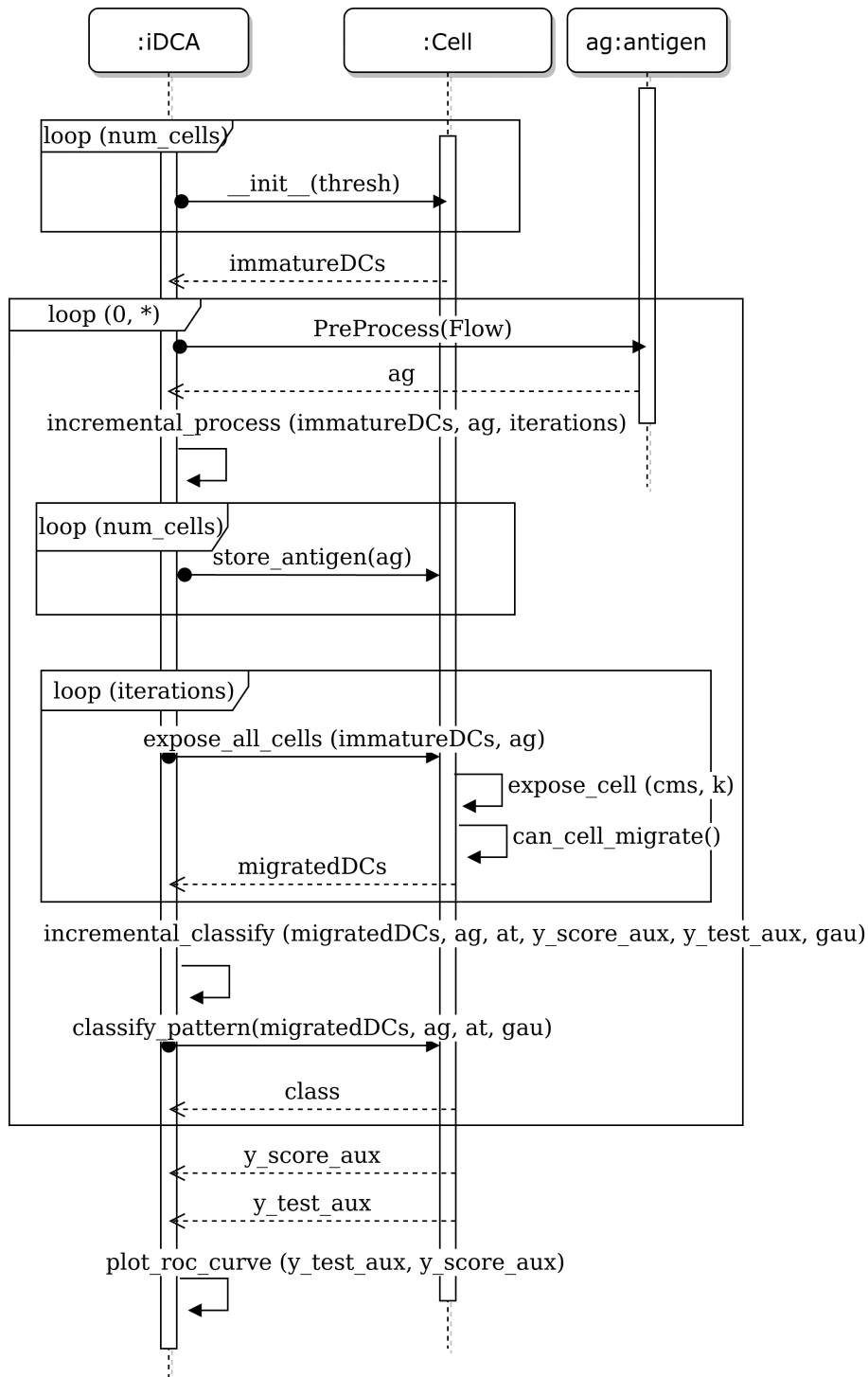


Figure C.1: Incremental Dendritic Cell Algorithm Unified Modelling Language class diagram.

Figure C.2: Network data collection *Unified Modelling Language* sequence diagram.

Figure C.3: *Incremental Dendritic Cell Algorithm Unified Modelling Language* sequence diagram.

Appendix D

A-HIDS Pipeline

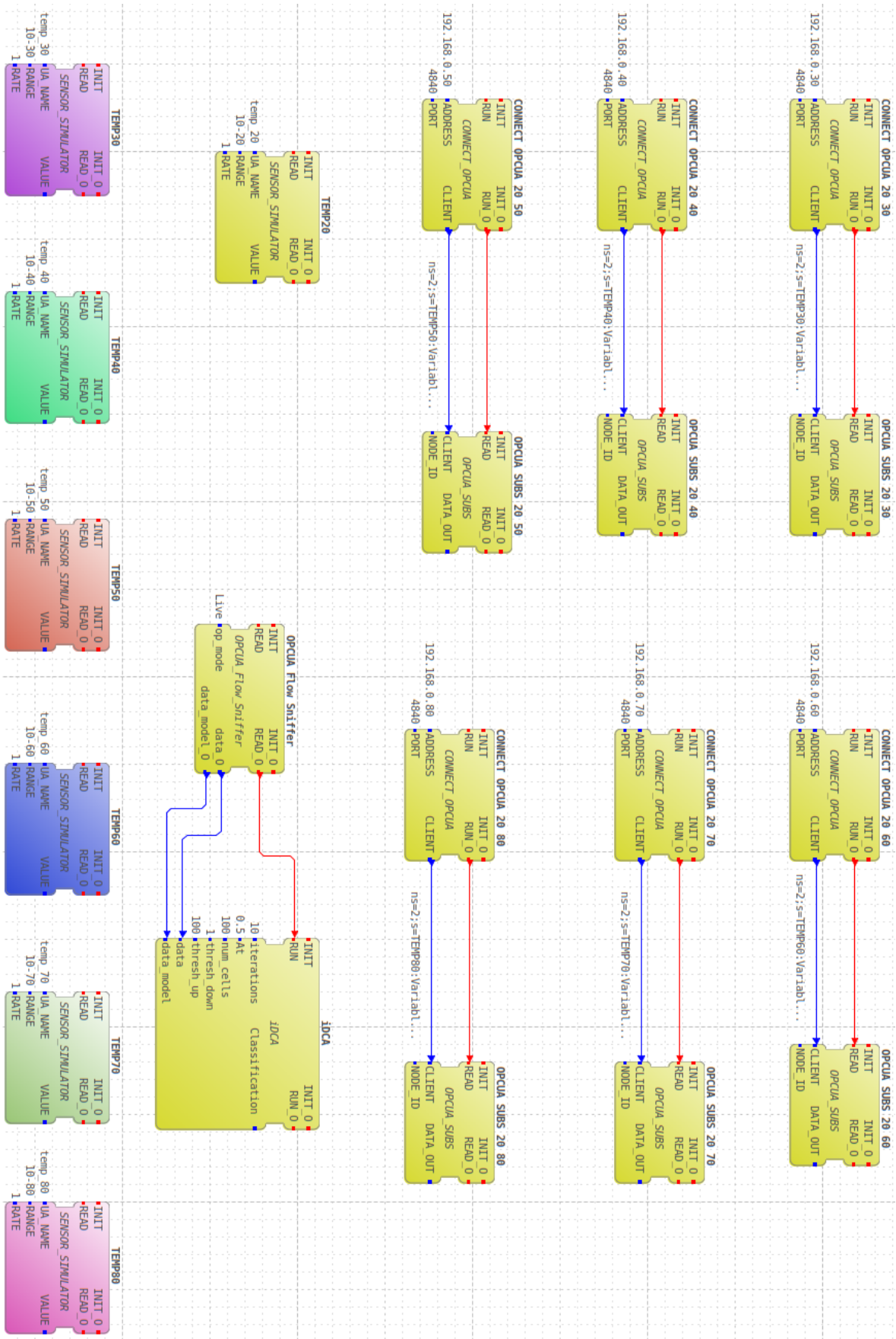


Figure D.1: Experimental setup - Function Block Pipeline.

Appendix E

SWOT Analysis

Table E.1: *Dendritic Cell Algorithm* SWOT analysis in its application in binary classification problems.

Strengths		Weaknesses	
- As an anomaly-based IDS, the DCA is suitable to detect novel attacks. DCA presents advantages over conventional anomaly detection techniques when large sized, noisy and timely ordered datasets are considered. - No-dependency of a training stage makes the DCA suitable to be applied in industrial contexts, where training periods may be prohibit and 'anomaly-free' normal profiles can be difficult to provide for training purposes. - The temporal correlation (signals and <i>Antigens</i>) facilitates the processing of data from disparate sources or sensors, so critical damage to the system can be measured and associated with a root cause. - Deployment of the DCA in the Edge layer of a CPPS will provide smart and adaptive intrusion detection. DCA may be adapted to be compliant with industrial network protocols, such a OPC UA and used as a NIDS to process network packets and flows.		- A theoretical investigation and runtime analysis of the algorithm will enable the production of a formal definition and understand if it is scalable to big datasets. All the DCA properties may be supported in Python, enabling the integration with the most advanced data science and ML technologies. - Both automated feature-selection and strategies for real-time analysis and classification exist in the data science field and can be applied to the DCA. Automated feature-selection and data preprocessing for signal mapping will help on reducing manual expert characterization of input signals and unnecessary empirical parameter tuning. - Considering the stochastic nature of the algorithm, structural modifications to the algorithm are needed to avoid false positives and unexpected classification behavior, while enabling the algorithm to fully adapt to a given problem domain. - Incremental learning capabilities may be embedded in the DCA for online detection, avoiding keeping all data instance classification on hold, while the entire dataset is not processed, and the limitation of input data order. - A suitable DCA architecture may be designed for the successfully deployment of the algorithm in IIoT scenarios (distributed and compliant with industry standards, such as IEC 61499).	
Opportunities		Threats	
- The DCA deployment to resource constrained devices, such as Edge devices in IoT scenarios or sensor nodes in WSN, is suitable when compared to classical ML techniques. - When considering industrial scenarios, DCA won't require prohibit training phases. - An OPC UA network traffic dataset may be used to validate a DCA-based NIDS implementation.		- Avoid multi-class anomaly detection, since the algorithm is application dependent and limited to binary classification problems. - Automated feature selection/reduction using non-linear techniques may increase the performance of the DCA in complex non-linear problems. - A MBE to enable security by design can reduce greatly the cost and effort of designing and deploying a DCA-based IDS in SMS scenarios.	

Appendix F

Cross-Correlation Results

Table F.1: *M2M Using OPC UA*: feature cross-correlation with class label.

Feature	Correlation
<i>flags</i>	-0.99987
<i>pktTotalCount</i>	-0.99491
<i>octetTotalCount</i>	-0.995046
<i>avg_ps</i>	-0.993052
<i>msg_size</i>	-0.99318
<i>min_msg_size</i>	-0.990805
<i>count</i>	0.822887
<i>srv_count</i>	0.822843
<i>f_octetTotalCount</i>	-0.982475
<i>f_rate</i>	-0.997346
<i>b_pktTotalCount</i>	-0.99974
<i>b_octetTotalCount</i>	-0.996367
<i>b_flowStart</i>	-0.99974
<i>label</i>	1

Table F.2: *KDD Cup99*: feature cross-correlation with class label.

Feature	Correlation
<i>logged_in</i>	-0.79528
<i>count</i>	0.752977
<i>srv_count</i>	0.566828
<i>dst_host_count</i>	0.642107
<i>label</i>	1
<i>multilabel</i>	0.942208

Table F.3: *NSL-KDD*: feature cross-correlation with class label.

Feature	Correlation
<i>logged_in</i>	-0.688084
<i>count</i>	0.57879
<i>serror_rate</i>	0.649952
<i>srv_error_rate</i>	0.647817
<i>same_srv_rate</i>	-0.749237
<i>dst_host_srv_count [SS]</i>	-0.719292
<i>dst_host_same_srv_rate</i>	-0.692212
<i>dst_host_serror_rate</i>	0.65105
<i>dst_host_srv_error_rate</i>	0.653759
<i>class</i>	1

Table F.4: *UNSW-NB15*: feature cross-correlation with class label.

Feature	Correlation
<i>sttl</i>	0.504159
<i>swin</i>	-0.414504

Appendix G

IEC 61499-based Security Solutions

Table G.1: *International Electrotechnical Commission 61499*-based Security Solutions.

Work	Methods	Application Scenario	Data Analyzed	Tools	Attack/Fault Spectrum	Assessment
[289]	Security Risk Analysis	Electric Grid FLISR (Simulated)	Network (MODBUS & DNP3)	Matlab; Simulink.	DoS; Attempted break-ins; Message Spoofing.	N.A.
[290]	Encryption of Data Communications	PLC in ICS Lab Testbed (Raspberry Pi)	Network (TCP & UDP)	AES; KE.	N.A.	2-10 ms Latency of Encryption Process
[291]	Rule-based IDPS	PLC in ICS Lab Testbed (Wago PFC200 PLC)	Network (TCP & UDP)	Snort; hping3.	DoS/DDoS; Masquerade; Flooding.	Packets Drop (N.A. results)
[293, 294]	MAS-based Fault Diagnostic	CPPS	Sensor (Temperature)	GORITE; SysML.	Sensor Fault; Software Fault; Actuator Electrical & Mechanical Faults.	Performance & Timeliness (N.A. results)
[295]	Secure Links (Protection of Communications)	PLC in ICS Lab Testbed (Wago PFC200 PLC)	Network (TCP & UDP)	TLS; AEAD; KE; TORUS.	N.A.	2-5 ms Latency of AEAD; 370-500 ms Latency of KE; MI = 75.68; S* = 23.8; DC* = 20.6; C* = 44.4.
Proposed approach	A-HIDS	CPPS Lab Testbed (Raspberry Pi)	Network (OPC UA)	iDCA; DINASORE; Pyshark; River.	DoS; MITM; Message Spoofing.	30% CPU; 50% RAM; 300 KB/s Network I/O; 0.1% Packet Drop; MI = 100.12; S* = 16; DC* = 0.65; C* = 16.65.

References

- [1] Rui Pinto and Gil Gonçalves. Application of artificial immune systems in advanced manufacturing. *Array*, page 100238, 2022. URL: <https://www.sciencedirect.com/science/article/pii/S2590005622000741>, doi:<https://doi.org/10.1016/j.array.2022.100238>. Cited on pages ix and 84.
- [2] Rui Pinto, Gil Gonçalves, Jerker Delsing, and Eduardo Tovar. Enabling data-driven anomaly detection by design in cyber-physical production systems. *Cybersecurity*, 5(1):9, 2022. URL: <https://doi.org/10.1186/s42400-022-00114-z>, doi: [10.1186/s42400-022-00114-z](https://doi.org/10.1186/s42400-022-00114-z). Cited on pages ix, 90, 100, 107, and 138.
- [3] Carlos Pinto, Rui Pinto, and Gil Gonçalves. Towards bio-inspired anomaly detection using the cursory dendritic cell algorithm. *Algorithms*, 15(1), 2022. URL: <https://www.mdpi.com/1999-4893/15/1/1>, doi:[10.3390/a15010001](https://doi.org/10.3390/a15010001). Cited on pages ix, xvii, xviii, 55, 65, 67, 70, 71, 90, 99, 106, 111, 124, 127, 130, and 133.
- [4] Rui Pinto, Gil Gonçalves, Jerker Delsing, and Eduardo Tovar. Incremental dendritic cell algorithm for intrusion detection in cyber-physical production systems. In Kohei Arai, editor, *Intelligent Computing*, pages 664–680, Cham, 2021. Springer International Publishing. Cited on pages ix, 90, 100, 106, 107, 110, 124, 127, and 138.
- [5] Rui Pinto, Gil Gonçalves, Eduardo Tovar, and Jerker Delsing. Attack detection in cyber-physical production systems using the deterministic dendritic cell algorithm. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1552–1559, 2020. doi:[10.1109/ETFA46521.2020.9212021](https://doi.org/10.1109/ETFA46521.2020.9212021). Cited on pages x, 90, 100, 106, 107, 110, 124, and 138.
- [6] Liliana Antão, Rui Pinto, João Reis, and Gil Gonçalves. Requirements for testing and validating the industrial internet of things. In *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 110–115, 2018. doi:[10.1109/ICSTW.2018.00036](https://doi.org/10.1109/ICSTW.2018.00036). Cited on pages x and 16.
- [7] Rui Pinto, Rosaldo J. F. Rossetti, and Gil Gonçalves. Wireless sensor network simulation for fault detection in industrial processes. In *2016 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)*, pages 1–6, 2016. Cited on page x.
- [8] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun, and Hui-Ying Du. Research on the architecture of internet of things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 5, pages V5–484–V5–487, 2010. doi:[10.1109/ICACTE.2010.5579493](https://doi.org/10.1109/ICACTE.2010.5579493). Cited on pages xvii and 18.

- [9] DIGI2-FEUP. DINASORE. <https://digi2-feup.github.io/dinasore/>, 2021. [Online; accessed July 2022]. Cited on pages xvii, 28, 29, and 31.
- [10] Sevil Sen. Chapter 4 - a survey of intrusion detection systems using evolutionary computation. In Xin-She Yang, Su Fong Chien, and Tiew On Ting, editors, *Bio-Inspired Computation in Telecommunications*, pages 73–94. Morgan Kaufmann, Boston, 2015. URL: <https://www.sciencedirect.com/science/article/pii/B9780128015384000045>, doi:<https://doi.org/10.1016/B978-0-12-801538-4.00004-5>. Cited on pages xvii and 36.
- [11] Dhruva Kumar Bhattacharyya and Jugal Kumar Kalita. *Network anomaly detection: A machine learning perspective*. Chapman and Hall/CRC, 2019. Cited on pages xvii, 43, 44, and 48.
- [12] OPC UA. Interoperability for Industrie 4.0 and the Internet of Things. <https://opcfoundation.org/wp-content/uploads/2017/11/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN.pdf>, 2017. [Online; accessed July 2022]. Cited on pages xvii and 47.
- [13] J.O. Kephart and D.M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003. doi:[10.1109/MC.2003.1160055](https://doi.org/10.1109/MC.2003.1160055). Cited on pages xvii, 56, and 57.
- [14] P. H. Richter. A network theory of the immune system. *European Journal of Immunology*, 5(5):350–354, 1975. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/eji.1830050511>, arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1002/eji.1830050511>, doi:<https://doi.org/10.1002/eji.1830050511>. Cited on pages xviii, 69, and 70.
- [15] AutomationDirect. History of the PLC. <https://library.automationdirect.com/history-of-the-plc/>, 2020. [Online; accessed July 2022]. Cited on page 5.
- [16] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & Information Systems Engineering*, 6(4):239–242, 2014. URL: <https://doi.org/10.1007/s12599-014-0334-4>, doi:[10.1007/s12599-014-0334-4](https://doi.org/10.1007/s12599-014-0334-4). Cited on page 5.
- [17] Henning Kagermann, Johannes Helbig, Ariane Hellinger, and Wolfgang Wahlster. *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry; Final Report of the Industrie 4.0 Working Group*. Forschungsunion, 2013. Cited on page 6.
- [18] Julie Tran. SMLC - Smart Manufacturing Leadership Coalition. <https://smlconsortium.org/>, 2016. [Online; accessed July 2022]. Cited on page 6.
- [19] Ling Li. China’s manufacturing locus in 2025: With a comparison of “made-in-china 2025” and “industry 4.0”. *Technological Forecasting and Social Change*, 135:66–74, 2018. URL: <https://www.sciencedirect.com/science/article/pii/S0040162517307254>, doi:<https://doi.org/10.1016/j.techfore.2017.05.028>. Cited on page 6.
- [20] Michel Athimon. Usine du Futur. <http://industriedufutur.fim.net/>, 2016. [Online; accessed July 2022]. Cited on page 6.

- [21] Ruth McKernan. How Catapults can help your business innovate. <https://www.yumpu.com/en/document/view/55348302/how-catapults-can-help-your-business-innovate/17>, 2016. [Online; accessed July 2022]. Cited on page 6.
- [22] Marco. Innovation in Manufacturing 3.0 Strategy Needs Better Focus with Clearer Direction. <https://2bnews.wordpress.com/2015/12/09/innovation-in-manufacturing-3-0-strategy-needs-better-focus-with-clearer-direction/>, December 2015. [Online; accessed July 2022]. Cited on page 6.
- [23] Sara Zaske. Germany's vision for Industrie 4.0: The revolution will be digitised. <http://www.zdnet.com/article/germanys-vision-for-industrie-4-0-the-revolution-will-be-digitised/>, February 2015. [Online; accessed July 2022]. Cited on page 6.
- [24] Rainer Drath and Alexander Horch. Industrie 4.0: Hit or hype? [industry forum]. *IEEE Industrial Electronics Magazine*, 8(2):56–58, 2014. doi:10.1109/MIE.2014.2312079. Cited on page 6.
- [25] Ercan Oztemel and Samet Gursev. Literature review of industry 4.0 and related technologies. *Journal of Intelligent Manufacturing*, 31(1):127–182, 2020. URL: <https://doi.org/10.1007/s10845-018-1433-8>, doi:10.1007/s10845-018-1433-8. Cited on pages 6, 84, 85, and 103.
- [26] Dong Chen, Guiran Chang, Lizhong Jin, Xiaodong Ren, Jiajia Li, and Fengyun Li. A novel secure architecture for the internet of things. In *2011 Fifth International Conference on Genetic and Evolutionary Computing*, pages 311–314, 2011. doi:10.1109/ICGEC.2011.77. Cited on page 8.
- [27] R. Sekar and Thomas Bowen. On preventing intrusions by process behavior monitoring. In *1st Workshop on Intrusion Detection and Network Monitoring (ID 99)*, Santa Clara, CA, April 1999. USENIX Association. URL: <https://www.usenix.org/conference/id-99/preventing-intrusions-process-behavior-monitoring>. Cited on page 8.
- [28] Andrei Costin, Jonas Zaddach, Aurélien Francillon, and Davide Balzarotti. A large-scale analysis of the security of embedded firmwares. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 95–110, San Diego, CA, August 2014. USENIX Association. URL: <https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/costin>. Cited on page 8.
- [29] Ang Cui and Salvatore J. Stolfo. A quantitative analysis of the insecurity of embedded network devices: Results of a wide-area scan. In *Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC '10*, page 97–106, New York, NY, USA, 2010. Association for Computing Machinery. URL: <https://doi.org/10.1145/1920261.1920276>, doi:10.1145/1920261.1920276. Cited on page 8.
- [30] George Loukas. *Cyber-physical attacks: A growing invisible threat*. Butterworth-Heinemann, 2015. Cited on page 8.
- [31] Jill Slay and Michael Miller. Lessons learned from the maroochy water breach. In Eric Goetz and Sujeet Sheno, editors, *Critical Infrastructure Protection*, pages 73–82, Boston, MA, 2008. Springer US. Cited on page 8.

- [32] Marie Baezner and Patrice Robin. Stuxnet. Technical report, ETH Zurich, 2017. URL: https://www.researchgate.net/publication/323199431_Stuxnet. Cited on page 8.
- [33] Robert M Lee, Michael J Assante, and Tim Conway. German steel mill cyber attack. Technical report, Industrial Control Systems, 2014. URL: https://assets.contentstack.io/v3/assets/blt36c2e63521272fdc/bltc79a41dbf7d1441e/607f235775873e466bcc539c/ICS-CPPE-case-Study-2-German-Steelworks_Facility.pdf. Cited on page 8.
- [34] Rui Pinto. M2m using opc ua, 2020. URL: <https://dx.doi.org/10.21227/ychv-6c68>, doi:10.21227/ychv-6c68. Cited on pages 11, 110, 124, 128, 137, 140, 141, 155, 160, and 200.
- [35] National Research Council. *Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*. The National Academies Press, Washington, DC, 2001. URL: <http://www.nap.edu/catalog/10193/embedded-everywhere-a-research-agenda-for-networked-systems-of-embedded>, doi:10.17226/10193. Cited on page 16.
- [36] Peter Marwedel. *Embedded system design: Embedded systems foundations of cyber-physical systems*. Springer Science & Business Media, 2010. Cited on page 16.
- [37] Edward A. Lee. Computing foundations and practice for cyber-physical systems: A preliminary report. Technical Report UCB/EECS-2007-72, EECS Department, University of California, Berkeley, May 2007. URL: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-72.html>. Cited on page 16.
- [38] Edward A. Lee. Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369, 2008. doi:10.1109/ISORC.2008.25. Cited on page 16.
- [39] Ariane Hellinger and Heinrich Seeger. Cyber-physical systems. driving force for innovation in mobility, health, energy and production. *Acatech Position Paper, National Academy of Science and Engineering*, 2011. Cited on page 17.
- [40] Sabina Jeschke, Christian Brecher, Tobias Meisen, Denis Özdemir, and Tim Eschert. *Industrial Internet of Things and Cyber Manufacturing Systems*, pages 3–19. Springer International Publishing, Cham, 2017. URL: https://doi.org/10.1007/978-3-319-42559-7_1, doi:10.1007/978-3-319-42559-7_1. Cited on page 18.
- [41] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, 2014. doi:10.1109/TII.2014.2300753. Cited on page 18.
- [42] Martin Hankel and Bosch Rexroth. The reference architectural model industrie 4.0 (rami 4.0). Technical report, ZVEI, 2015. URL: <https://przemysl-40.pl/wp-content/uploads/2010-The-Reference-Architectural-Model-Industrie-40.pdf>. Cited on page 18.
- [43] Shi-Wan Lin, B Miller, J Durand, R Joshi, P Didier, A Chigani, R Torenbeek, D Duggal, R Martin, G Bleakley, et al. Industrial internet reference

- architecture. Technical report, Industrial Internet Consortium (IIC), Tech. Rep, 2015. URL: https://www.iiconsortium.org/pdf/SHI-WAN%20LIN_IIRA-v1%208-release-20170125.pdf. Cited on page 18.
- [44] Michael Weyrich and Christof Ebert. Reference architectures for the internet of things. *IEEE Software*, 33(1):112–116, 2016. doi:10.1109/MS.2016.20. Cited on page 18.
- [45] Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan. Future internet: The internet of things architecture, possible applications and key challenges. In *2012 10th International Conference on Frontiers of Information Technology*, pages 257–260, 2012. doi:10.1109/FIT.2012.53. Cited on page 18.
- [46] Song Han, Miao Xie, Hsiao-Hwa Chen, and Yun Ling. Intrusion detection in cyber-physical systems: Techniques and challenges. *IEEE Systems Journal*, 8(4):1052–1062, 2014. doi:10.1109/JSYST.2013.2257594. Cited on pages 19, 115, and 116.
- [47] Eric Ke Wang, Yunming Ye, Xiaofei Xu, S. M. Yiu, L. C. K. Hui, and K. P. Chow. Security issues and challenges for cyber physical system. In *2010 IEEE/ACM Int’l Conference on Green Computing and Communications Int’l Conference on Cyber, Physical and Social Computing*, pages 733–738, 2010. doi:10.1109/GreenCom-CPSCoM.2010.36. Cited on page 19.
- [48] Christopher Kaestle, Hans Fleischmann, Michael Scholz, Stefan Haerter, and Joerg Franke. *Cyber-Physical Electronics Production*, pages 47–78. Springer International Publishing, Cham, 2017. URL: https://doi.org/10.1007/978-3-319-42559-7_3, doi:10.1007/978-3-319-42559-7_3. Cited on page 20.
- [49] Tharam Dillon, Vidyasagar Potdar, Jaipal Singh, and Alex Talevski. Cyber-physical systems: Providing quality of service (qos) in a heterogeneous systems-of-systems environment. In *5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011)*, pages 330–335, 2011. doi:10.1109/DEST.2011.5936595. Cited on page 20.
- [50] Israa Alqassem. Privacy and security requirements framework for the internet of things (iot). In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, page 739–741, New York, NY, USA, 2014. Association for Computing Machinery. URL: <https://doi.org/10.1145/2591062.2591201>, doi:10.1145/2591062.2591201. Cited on page 20.
- [51] Se-Ra Oh and Young-Gab Kim. Security requirements analysis for the iot. In *2017 International Conference on Platform Technology and Service (PlatCon)*, pages 1–6, 2017. doi:10.1109/PlatCon.2017.7883727. Cited on page 20.
- [52] Ovidiu Vermesan and Peter Friess. *Internet of things-from research and innovation to market deployment*, volume 29. River Publishers Aalborg, 2014. Cited on page 20.
- [53] NK Fantana, Till Riedel, Jochen Schlick, Stefan Ferber, Jürgen Hupp, Stephen Miles, Florian Michahelles, and Stefan Svensson. Iot applications—value creation for industry. *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*, page 153, 2013. Cited on page 20.

- [54] Hongyu Pei Breivold and Kristian Sandström. Internet of things for industrial automation – challenges and technical solutions. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pages 532–539, 2015. doi:10.1109/DSDIS.2015.11. Cited on pages 20 and 21.
- [55] K. Thramboulidis. Model-integrated mechatronics - toward a new paradigm in the development of manufacturing systems. *IEEE Transactions on Industrial Informatics*, 1(1):54–61, 2005. doi:10.1109/TII.2005.844427. Cited on page 26.
- [56] N. Jazdi. Cyber physical systems in the context of industry 4.0. In *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, pages 1–4, 2014. doi:10.1109/AQTR.2014.6857843. Cited on page 27.
- [57] Esteban Querol, Julio Ariel Romero, Julio Serrano, and Roberto Sanchis. Evaluation of closed loop control applications using different event management strategies under iec 61499. In *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–8, 2016. doi:10.1109/EBCCSP.2016.7605263. Cited on page 27.
- [58] Eclipse Foundation. Eclipse 4diac. <https://www.eclipse.org/4diac/>, 2021. [Online; accessed July 2022]. Cited on page 27.
- [59] NxtControl. NxT Technology IDE. <https://www.nxtcontrol.com/en/engineering/>, 2021. [Online; accessed July 2022]. Cited on page 27.
- [60] Holobloc In. HOLOBLOC, Inc. <https://www.holobloc.com/>, 2020. [Online; accessed July 2022]. Cited on page 27.
- [61] Laurin Prenzel, Alois Zoitl, and Julien Provost. Iec 61499 runtime environments: A state of the art comparison. In Roberto Moreno-Díaz, Franz Pichler, and Alexis Quesada-Arencibia, editors, *Computer Aided Systems Theory – EUROCAST 2019*, pages 453–460, Cham, 2020. Springer International Publishing. Cited on page 28.
- [62] Eliseu Moura Pereira, João Pedro Correia dos Reis, and Gil Gonçalves. Dinasore: A dynamic intelligent reconfiguration tool for cyber-physical production systems. In *SAM IoT*, pages 63–71, 2020. Cited on page 28.
- [63] Gulnara Zhabelova and Valeriy Vyatkin. Towards software metrics for evaluating quality of iec 61499 automation software. In *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–8, 2015. doi:10.1109/ETFA.2015.7301502. Cited on pages 30 and 189.
- [64] Rodrigo Roman, Pablo Najera, and Javier Lopez. Securing the internet of things. *Computer*, 44(9):51–58, 2011. doi:10.1109/MC.2011.291. Cited on page 34.
- [65] Ben Rossi. From prevention to detection: why businesses must shift their security focus . <http://www.information-age.com/technology/security/123459859/prevention-detection-why-businesses-must-shift-their-security-focus>, 2015. [Online; accessed July 2022]. Cited on page 35.
- [66] Jim Brennan. Detection Is NOT the New Prevention . <https://securityintelligence.com/detection-not-new-prevention-advanced-threat-protection/>, 2014. [Online; accessed July 2022]. Cited on page 35.

- [67] D.E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, 1987. doi:[10.1109/TSE.1987.232894](https://doi.org/10.1109/TSE.1987.232894). Cited on page 35.
- [68] Zakira Inayat, Abdullah Gani, Nor Badrul Anuar, Muhammad Khurram Khan, and Shahid Anwar. Intrusion response systems: Foundations, design, and challenges. *Journal of Network and Computer Applications*, 62:53–74, 2016. URL: <https://www.sciencedirect.com/science/article/pii/S1084804515002994>, doi:<https://doi.org/10.1016/j.jnca.2015.12.006>. Cited on page 36.
- [69] Monowar H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys Tutorials*, 16(1):303–336, 2014. doi:[10.1109/SURV.2013.052213.00046](https://doi.org/10.1109/SURV.2013.052213.00046). Cited on page 36.
- [70] Cynthia Vargas Martinez and Birgit Vogel-Heuser. A host intrusion detection system architecture for embedded industrial devices. *Journal of the Franklin Institute*, 358(1):210–236, 2021. URL: <https://www.sciencedirect.com/science/article/pii/S0016003219305307>, doi:<https://doi.org/10.1016/j.jfranklin.2019.03.037>. Cited on pages 36, 131, 183, and 191.
- [71] Dewan Md. Farid and Mohammad Zahidur Rahman. Learning intrusion detection based on adaptive bayesian algorithm. In *2008 11th International Conference on Computer and Information Technology*, pages 652–656, 2008. doi:[10.1109/ICCITECHN.2008.4803036](https://doi.org/10.1109/ICCITECHN.2008.4803036). Cited on page 37.
- [72] Shi Zhong, T.M. Khoshgoftaar, and S.V. Nath. A clustering approach to wireless network intrusion detection. In *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, pages 7 pp.–196, 2005. doi:[10.1109/ICTAI.2005.5](https://doi.org/10.1109/ICTAI.2005.5). Cited on page 37.
- [73] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), July 2009. URL: <https://doi.org/10.1145/1541880.1541882>, doi:[10.1145/1541880.1541882](https://doi.org/10.1145/1541880.1541882). Cited on page 37.
- [74] Ismail Butun, Salvatore D. Morgera, and Ravi Sankar. A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 16(1):266–282, 2014. doi:[10.1109/SURV.2013.050113.00191](https://doi.org/10.1109/SURV.2013.050113.00191). Cited on page 38.
- [75] P. Brutch and C. Ko. Challenges in intrusion detection for wireless ad-hoc networks. In *2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings.*, pages 368–373, 2003. doi:[10.1109/SAINTW.2003.1210188](https://doi.org/10.1109/SAINTW.2003.1210188). Cited on page 39.
- [76] Jinsong Zhou, Zhuo Chen, and Wei Jiang. Probability based ids towards secure wmn. In *2010 2nd International Workshop on Intelligent Systems and Applications*, pages 1–4, 2010. doi:[10.1109/IWISA.2010.5473597](https://doi.org/10.1109/IWISA.2010.5473597). Cited on page 39.
- [77] G. Bella, G. Costantino, and S. Riccobene. Managing reputation over manets. In *2008 The Fourth International Conference on Information Assurance and Security*, pages 255–260, 2008. doi:[10.1109/IAS.2008.24](https://doi.org/10.1109/IAS.2008.24). Cited on page 39.
- [78] Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection techniques for cyber-physical systems. *ACM Comput. Surv.*, 46(4), March 2014. URL: <https://doi.org/10.1145/2542049>, doi:[10.1145/2542049](https://doi.org/10.1145/2542049). Cited on page 40.

- [79] Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection in wireless network applications. *Computer Communications*, 42:1–23, 2014. URL: <https://www.sciencedirect.com/science/article/pii/S0140366414000280>, doi:<https://doi.org/10.1016/j.comcom.2014.01.012>. Cited on page 40.
- [80] Yizhong Ma, Hui Cao, and Jun Ma. The intrusion detection method based on game theory in wireless sensor network. In *2008 First IEEE International Conference on Ubi-Media Computing*, pages 326–331, 2008. doi:[10.1109/UMEDIA.2008.4570911](https://doi.org/10.1109/UMEDIA.2008.4570911). Cited on page 42.
- [81] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, and E. Spafford. Adepts: adaptive intrusion response using attack graphs in an e-commerce environment. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 508–517, 2005. doi:[10.1109/DSN.2005.17](https://doi.org/10.1109/DSN.2005.17). Cited on page 42.
- [82] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), July 2009. URL: <https://doi.org/10.1145/1541880.1541882>, doi:[10.1145/1541880.1541882](https://doi.org/10.1145/1541880.1541882). Cited on page 43.
- [83] Marcel Müller, Elmar Wings, and Lars Bergmann. Developing open source cyber-physical systems for service-oriented architectures using opc ua. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 83–88, 2017. doi:[10.1109/INDIN.2017.8104751](https://doi.org/10.1109/INDIN.2017.8104751). Cited on page 47.
- [84] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham, and M.A. Zissman. Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 12–26 vol.2, 2000. doi:[10.1109/DISCEX.2000.821506](https://doi.org/10.1109/DISCEX.2000.821506). Cited on page 49.
- [85] Julien Polge, Jeremy Robert, and Yves Le Traon. Assessing the impact of attacks on opc-ua applications in the industry 4.0 era. In *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6, 2019. doi:[10.1109/CCNC.2019.8651671](https://doi.org/10.1109/CCNC.2019.8651671). Cited on page 50.
- [86] Salvatore Cavalieri, Giovanni Cutuli, and Salvatore Monteleone. Evaluating impact of security on opc ua performance. In *3rd International Conference on Human System Interaction*, pages 687–694, 2010. doi:[10.1109/HSI.2010.5514495](https://doi.org/10.1109/HSI.2010.5514495). Cited on page 50.
- [87] Charles Varlei Neu, Ina Schiering, and Avelino Zorzo. Simulating and detecting attacks of untrusted clients in opc ua networks. In *Proceedings of the Third Central European Cybersecurity Conference*, CECC 2019, New York, NY, USA, 2019. Association for Computing Machinery. URL: <https://doi.org/10.1145/3360664.3360675>, doi:[10.1145/3360664.3360675](https://doi.org/10.1145/3360664.3360675). Cited on page 50.
- [88] Stephen D. Bay, Dennis Kibler, Michael J. Pazzani, and Padhraic Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *SIGKDD Explor. Newsl.*, 2(2):81–85, December 2000. URL: <https://doi.org/10.1145/380995.381030>, doi:[10.1145/380995.381030](https://doi.org/10.1145/380995.381030). Cited on pages 51, 110, and 127.
- [89] S.J. Stolfo, Wei Fan, Wenke Lee, A. Prodromidis, and P.K. Chan. Cost-based modeling for fraud and intrusion detection: results from the jam project. In *Proceedings DARPA*

- Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 130–144 vol.2, 2000. doi:10.1109/DISCEX.2000.821515. Cited on page 51.
- [90] Mahbod Tavallaei, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6, 2009. doi:10.1109/CISDA.2009.5356528. Cited on page 52.
- [91] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6, 2015. doi:10.1109/MilCIS.2015.7348942. Cited on pages 52 and 127.
- [92] IBM. Autonomic Computing: IBM's Perspective on the State of Information Technology. http://people.scs.carleton.ca/~soma/biosec/readings/autonomic_computing.pdf, October 2001. [Online; accessed July 2022]. Cited on page 56.
- [93] Rolf P Würtz. *Organic computing*. Springer Science & Business Media, 2008. Cited on page 56.
- [94] Manish Parashar and Salim Hariri. Autonomic computing: An overview. In Jean-Pierre Banâtre, Pascal Fradet, Jean-Louis Giavitto, and Olivier Michel, editors, *Unconventional Programming Paradigms*, pages 257–269, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. Cited on page 56.
- [95] A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, 2003. doi:10.1147/sj.421.0005. Cited on page 56.
- [96] Andrew Berns and Sukumar Ghosh. Dissecting self-* properties. In *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 10–19, 2009. doi:10.1109/SASO.2009.25. Cited on page 57.
- [97] L.A. Zadeh. On the definition of adaptivity. *Proceedings of the IEEE*, 51(3):469–470, 1963. doi:10.1109/PROC.1963.1852. Cited on page 57.
- [98] Umut A. Acar. Self-adjusting computation: (an overview). In *Proceedings of the 2009 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM '09*, page 1–6, New York, NY, USA, 2009. Association for Computing Machinery. URL: <https://doi.org/10.1145/1480945.1480946>, doi:10.1145/1480945.1480946. Cited on page 57.
- [99] Edsger W. Dijkstra. *Self-Stabilization in Spite of Distributed Control*, pages 41–46. Springer New York, New York, NY, 1982. URL: https://doi.org/10.1007/978-1-4612-5695-3_7, doi:10.1007/978-1-4612-5695-3_7. Cited on page 57.
- [100] Shlomi Dolev. *Self-stabilization*. MIT press, 2000. Cited on page 57.
- [101] P.K. McKinley, S.M. Sadjadi, E.P. Kasten, and B.H.C. Cheng. Composing adaptive software. *Computer*, 37(7):56–64, 2004. doi:10.1109/MC.2004.48. Cited on page 57.
- [102] Philip Koopman. Elements of the self-healing system problem space, Jun 2018. URL: https://kilthub.cmu.edu/articles/journal_contribution/

- [Elements_of_the_Self-Healing_System_Problem_Space/6622073/1](#), doi:10.1184/R1/6622073.v1. Cited on page 58.
- [103] Debanjan Ghosh, Raj Sharman, H. Raghav Rao, and Shambhu Upadhyaya. Self-healing systems — survey and synthesis. *Decision Support Systems*, 42(4):2164–2185, 2007. Decision Support Systems in Emerging Economies. URL: <https://www.sciencedirect.com/science/article/pii/S0167923606000807>, doi:<https://doi.org/10.1016/j.dss.2006.06.011>. Cited on page 58.
- [104] Gabi Dreö Rodosek, Kurt Geihs, Hartmut Schmeck, and Stiller Burkhard. Self-healing systems: Foundations and challenges. In Artur Andrzejak, Kurt Geihs, Onn Shehory, and John Wilkes, editors, *Self-Healing and Self-Adaptive Systems*, number 09201 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany. URL: <http://drops.dagstuhl.de/opus/volltexte/2009/2110>. Cited on page 58.
- [105] Fabian Kuhn, Stefan Schmid, and Roger Wattenhofer. A self-repairing peer-to-peer system resilient to dynamic adversarial churn. In Miguel Castro and Robbert van Renesse, editors, *Peer-to-Peer Systems IV*, pages 13–23, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. Cited on page 58.
- [106] N. Kandasamy, S. Abdelwahed, and J.P. Hayes. Self-optimization in computer systems via on-line control: application to power management. In *International Conference on Autonomic Computing, 2004. Proceedings.*, pages 54–61, 2004. doi:10.1109/ICAC.2004.1301347. Cited on page 58.
- [107] Bhaskar Krishnamachari, Stephen Wicker, Ramón Béjar, and Cèsar Fernández. On the complexity of distributed self-configuration in wireless networks. *Telecommunication Systems*, 22(1):33–59, 2003. URL: <https://doi.org/10.1023/A:1023426501170>, doi:10.1023/A:1023426501170. Cited on page 58.
- [108] J. Wildstrom, P. Stone, E. Witchel, R.J. Mooney, and M. Dahlin. Towards self-configuring hardware for distributed computer systems. In *Second International Conference on Autonomic Computing (ICAC'05)*, pages 241–249, 2005. doi:10.1109/ICAC.2005.63. Cited on page 58.
- [109] G.G. Lendaris. On the definition of self-organizing systems. *Proceedings of the IEEE*, 52(3):324–325, 1964. doi:10.1109/PROC.1964.2905. Cited on page 58.
- [110] Giovanna Di Marzo Serugendo, Anthony Karageorgos, Omer F Rana, and Franco Zambonelli. *Engineering self-organising systems: nature-inspired approaches to software engineering*, volume 2977. Springer, 2004. Cited on page 58.
- [111] Klaus Herrmann and Matthias Werner. A methodology for classifying selforganizing software systems. *International Transactions on Systems Science and Applications*, pages 41–50, 2006. Cited on page 58.
- [112] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. *Self-organising Systems*, pages 7–32. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. URL: https://doi.org/10.1007/978-3-642-17348-6_2, doi:10.1007/978-3-642-17348-6_2. Cited on page 58.

- [113] Andrew J. McMurtry, Clint A. Gilbert, Ben Y. Reis, Henry C. Chueh, Isaac S. Kohane, and Kenneth D. Mandl. A Self-scaling, Distributed Information Architecture for Public Health, Research, and Clinical Care. *Journal of the American Medical Informatics Association*, 14(4):527–533, 07 2007. URL: <https://doi.org/10.1197/jamia.M2371>, arXiv:<https://academic.oup.com/jamia/article-pdf/14/4/527/2366646/14-4-527.pdf>, doi:10.1197/jamia.M2371. Cited on page 58.
- [114] Pascal A. Felber and Ernst W. Biersack. Self-scaling networks for content distribution. In *In Proc. International Workshop on Self-* Properties in Complex Information Systems*, 2004. Cited on page 58.
- [115] Miranda Mowbray and Alexandre Bronstein. What kind of self-aware systems does the Grid need? <https://www.hpl.hp.com/techreports/2002/HPL-2002-266R1.pdf>, 2005. [Online; accessed July 2022]. Cited on page 58.
- [116] Marco D. Santambrogio, Henry Hoffmann, Jonathan Eastep, and Anant Agarwal. Enabling technologies for self-aware adaptive systems. In *2010 NASA/ESA Conference on Adaptive Hardware and Systems*, pages 149–156, 2010. doi:10.1109/AHS.2010.5546266. Cited on page 58.
- [117] M. Seltzer and C. Small. Self-monitoring and self-adapting operating systems. In *Proceedings. The Sixth Workshop on Hot Topics in Operating Systems (Cat. No. 97TB100133)*, pages 124–129, 1997. doi:10.1109/HOTOS.1997.595194. Cited on page 58.
- [118] Steven A. Hofmeyr and Stephanie Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000. doi:10.1162/106365600568257. Cited on pages 58 and 65.
- [119] D. Dasgupta and N. Atttoh-Okine. Immunity-based systems: a survey. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 1, pages 369–374 vol.1, 1997. doi:10.1109/ICSMC.1997.625778. Cited on page 58.
- [120] Jungwon Kim, Peter J. Bentley, Uwe Aickelin, Julie Greensmith, Gianni Tedesco, and Jamie Twycross. Immune system approaches to intrusion detection - a review. *Natural Computing*, 6(4):413–466, 2007. URL: <https://doi.org/10.1007/s11047-006-9026-4>, doi:10.1007/s11047-006-9026-4. Cited on pages 58 and 116.
- [121] Sukumar Ghosh and Arobinda Gupta. An exercise in fault-containment: Self-stabilizing leader election. *Information Processing Letters*, 59(5):281–288, 1996. URL: <https://www.sciencedirect.com/science/article/pii/0020019096001214>, doi:[https://doi.org/10.1016/0020-0190\(96\)00121-4](https://doi.org/10.1016/0020-0190(96)00121-4). Cited on page 58.
- [122] Colin English, Sotirios Terzis, and Paddy Nixon. Towards self-protecting ubiquitous systems: monitoring trust-based interactions. *Personal and Ubiquitous Computing*, 10(1):50–54, 2006. URL: <https://doi.org/10.1007/s00779-005-0030-y>, doi:10.1007/s00779-005-0030-y. Cited on page 59.
- [123] R. Ananthanarayanan, M. Mohania, and A. Gupta. Management of conflicting obligations in self-protecting policy-based systems. In *Second International Conference on Autonomic Computing (ICAC'05)*, pages 274–285, 2005. doi:10.1109/ICAC.2005.38. Cited on page 59.

- [124] Konrad Wegener, Thomas Gittler, and Lukas Weiss. Dawn of new machining concepts:: Compensated, intelligent, bioinspired. *Procedia CIRP*, 77:1–17, 2018. 8th CIRP Conference on High Performance Cutting (HPC 2018). URL: <https://www.sciencedirect.com/science/article/pii/S2212827118310436>, doi:<https://doi.org/10.1016/j.procir.2018.08.194>. Cited on page 59.
- [125] Ricardo Silva Peres, Xiaodong Jia, Jay Lee, Keyi Sun, Armando Walter Colombo, and Jose Barata. Industrial artificial intelligence in industry 4.0 - systematic review, challenges and outlook. *IEEE Access*, 8:220121–220139, 2020. doi:[10.1109/ACCESS.2020.3042874](https://doi.org/10.1109/ACCESS.2020.3042874). Cited on pages 59, 83, 84, and 85.
- [126] Jason Brownlee. *Clever algorithms: nature-inspired programming recipes*. Jason Brownlee, 2011. Cited on page 59.
- [127] Irun R Cohen. *Tending Adam’s Garden: evolving the cognitive immune self*. Academic Press, 2000. Cited on page 60.
- [128] Irun R. Cohen. Immune system computation and the immunological homunculus. In Oscar Nierstrasz, Jon Whittle, David Harel, and Gianna Reggio, editors, *Model Driven Engineering Languages and Systems*, pages 499–512, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. Cited on page 60.
- [129] Dipankar Dasgupta and Fernando Nino. *Immunological computation: theory and applications*. CRC Press, 2008. Cited on page 60.
- [130] Charles A. Janeway. How the immune system works to protect the host from infection: A personal view. *Proceedings of the National Academy of Sciences*, 98(13):7461–7468, 2001. URL: <https://www.pnas.org/content/98/13/7461>, arXiv:<https://www.pnas.org/content/98/13/7461.full.pdf>, doi:[10.1073/pnas.131202998](https://doi.org/10.1073/pnas.131202998). Cited on page 60.
- [131] Dipankar Dasgupta, Senhua Yu, and Fernando Nino. Recent advances in artificial immune systems: Models and applications. *Applied Soft Computing*, 11(2):1574–1587, 2011. The Impact of Soft Computing for the Progress of Artificial Intelligence. URL: <https://www.sciencedirect.com/science/article/pii/S1568494610002723>, doi:<https://doi.org/10.1016/j.asoc.2010.08.024>. Cited on pages 63, 65, and 85.
- [132] S. Forrest, A.S. Perelson, L. Allen, and R. Cherukuri. Self-nonsel self discrimination in a computer. In *Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 202–212, 1994. doi:[10.1109/RISP.1994.296580](https://doi.org/10.1109/RISP.1994.296580). Cited on pages 63, 64, and 65.
- [133] Leandro Nunes De Castro and Fernando J. Von Zuben. The clonal selection algorithm with engineering applications. In *In GECCO 2002 - Workshop Proceedings*, pages 36–37. Morgan Kaufmann, 2000. Cited on pages 63 and 67.
- [134] Jon Timmis, Mark Neal, and John Hunt. An artificial immune system for data analysis. *Biosystems*, 55(1):143–150, 2000. URL: <https://www.sciencedirect.com/science/article/pii/S0303264799000921>, doi:[https://doi.org/10.1016/S0303-2647\(99\)00092-1](https://doi.org/10.1016/S0303-2647(99)00092-1). Cited on page 63.

- [135] T. Knight and J. Timmis. Aine: an immunological approach to data mining. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 297–304, Los Alamitos, CA, USA, dec 2001. IEEE Computer Society. URL: <https://doi.ieeecomputersociety.org/10.1109/ICDM.2001.989532>, doi:10.1109/ICDM.2001.989532. Cited on page 63.
- [136] Uwe Aickelin and Steve Cayzer. The danger theory and its application to artificial immune systems. *CoRR*, abs/0801.3549, 2008. URL: <http://arxiv.org/abs/0801.3549>, arXiv:0801.3549. Cited on page 63.
- [137] Julie Greensmith, Uwe Aickelin, and Steve Cayzer. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In Christian Jacob, Marcin L. Pilat, Peter J. Bentley, and Jonathan I. Timmis, editors, *Artificial Immune Systems*, pages 153–167, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. Cited on pages 63 and 72.
- [138] Sahar Aldhaheri, Daniyal Alghazzawi, Li Cheng, Ahmed Barnawi, and Bandar A. Alzahrani. Artificial immune systems approaches to secure the internet of things: A systematic review of the literature and recommendations for future research. *Journal of Network and Computer Applications*, 157:102537, 2020. URL: <https://www.sciencedirect.com/science/article/pii/S1084804520300114>, doi:<https://doi.org/10.1016/j.jnca.2020.102537>. Cited on pages 63, 85, 86, and 106.
- [139] J. Greensmith, J. Twycross, and U. Aickelin. Dendritic cells for anomaly detection. In *2006 IEEE International Conference on Evolutionary Computation*, pages 664–671, 2006. doi:10.1109/CEC.2006.1688374. Cited on pages 64 and 116.
- [140] Yousof Al-Hammadi, Uwe Aickelin, and Julie Greensmith. Dca for bot detection. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1807–1816, 2008. doi:10.1109/CEC.2008.4631034. Cited on page 64.
- [141] R. Huang, H. Tawfik, and A. K. Nagar. Artificial dendritic cells algorithm for online break-in fraud detection. In *2009 Second International Conference on Developments in eSystems Engineering*, pages 181–189, 2009. doi:10.1109/DeSE.2009.59. Cited on page 64.
- [142] Moisés Danziger, Marcelo Lacerda, and Fernando B. de Lima Neto. Danger theory and multi-agents applied for addressing the deny of service detection problem in ieee 802.11 networks. In *2009 Ninth International Conference on Intelligent Systems Design and Applications*, pages 695–702, 2009. doi:10.1109/ISDA.2009.136. Cited on page 64.
- [143] Abdulaziz A. Alsulami and Saleh Zein-Sabatto. Detection and defense from false data injection attacks in aviation cyber-physical systems using artificial immune systems. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 69–75, 2020. doi:10.1109/CSCI51800.2020.00019. Cited on page 64.
- [144] Abdulaziz A. Alsulami and Saleh Zein-Sabatto. Resilient cyber-security approach for aviation cyber-physical systems protection against sensor spoofing attacks. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0565–0571, 2021. doi:10.1109/CCWC51732.2021.9376158. Cited on page 64.

- [145] S. Cayzer and U. Aickelin. A recommender system based on the immune network. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 1, pages 807–812 vol.1, 2002. doi:[10.1109/CEC.2002.1007029](https://doi.org/10.1109/CEC.2002.1007029). Cited on page 64.
- [146] Milad Dastan Zand, Mohammad Kalantari, and Shahram Golzari. File transfer scheduling optimization using artificial immune system. In *2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 17–22, 2013. doi:[10.1109/SNPD.2013.52](https://doi.org/10.1109/SNPD.2013.52). Cited on page 64.
- [147] Dorra Moalla, Sabeur Elkosantini, and Saber Darmoul. An artificial immune network to control traffic at a single intersection. In *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 1–7, 2013. Cited on page 64.
- [148] Di Chen and Fengbin Zhang. 5g message service system based on artificial immune dynamic adaptive mechanism. *IEEE Access*, 7:91146–91159, 2019. doi:[10.1109/ACCESS.2019.2927271](https://doi.org/10.1109/ACCESS.2019.2927271). Cited on page 64.
- [149] Takahiro Takeda, Kazuhiko Taniguchi, Kazunari Asari, Kei Kuramoto, Syoji Kobashi, and Yutaka Hata. Biometric personal authentication by one step foot pressure distribution change by fuzzy artificial immune system. In *International Conference on Fuzzy Systems*, pages 1–6, 2010. doi:[10.1109/FUZZY.2010.5584216](https://doi.org/10.1109/FUZZY.2010.5584216). Cited on page 64.
- [150] Chengquan Xu, Suxia Xu, and Wei Chen. Artificial immune system and its applications in gps single frequency precise point positioning. In *2008 3rd International Conference on Intelligent System and Knowledge Engineering*, volume 1, pages 180–183, 2008. doi:[10.1109/ISKE.2008.4730921](https://doi.org/10.1109/ISKE.2008.4730921). Cited on page 64.
- [151] H. Wang, D. Peng, W. Wang, H. Sharif, J. Wegiel, D. Nguyen, R. Bowne, and C. Backhaus. Artificial immune system based image pattern recognition in energy efficient wireless multimedia sensor networks. In *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pages 1–7, 2008. doi:[10.1109/MILCOM.2008.4753651](https://doi.org/10.1109/MILCOM.2008.4753651). Cited on page 64.
- [152] Zhou Ji and Dipankar Dasgupta. Revisiting negative selection algorithms. *Evolutionary Computation*, 15(2):223–251, 2007. doi:[10.1162/evco.2007.15.2.223](https://doi.org/10.1162/evco.2007.15.2.223). Cited on page 65.
- [153] Dipankar Dasgupta. Advances in artificial immune systems. *IEEE Computational Intelligence Magazine*, 1(4):40–49, 2006. doi:[10.1109/MCI.2006.329705](https://doi.org/10.1109/MCI.2006.329705). Cited on page 65.
- [154] Justin Balthrop, Fernando Esponda, Stephanie Forrest, and Matthew Glickman. Coverage and generalization in an artificial immune system. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, GECCO'02*, page 3–10, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. Cited on page 65.
- [155] Berna Haktanirlar Ulutas and Sadan Kulturel-Konak. A review of clonal selection algorithm and its applications. *Artificial Intelligence Review*, 36(2):117–138, 2011. URL: <https://doi.org/10.1007/s10462-011-9206-1>, doi:[10.1007/s10462-011-9206-1](https://doi.org/10.1007/s10462-011-9206-1). Cited on pages 66 and 85.

- [156] L.N. de Castro and F.J. Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251, 2002. doi: [10.1109/TEVC.2002.1011539](https://doi.org/10.1109/TEVC.2002.1011539). Cited on page 67.
- [157] Andrew Watkins, Jon Timmis, and Lois Boggess. Artificial immune recognition system (airs): An immune-inspired supervised learning algorithm. *Genetic Programming and Evolvable Machines*, 5(3):291–317, 2004. URL: <https://doi.org/10.1023/B:GENP.0000030197.83685.94>, doi:10.1023/B:GENP.0000030197.83685.94. Cited on page 68.
- [158] Johnny Kelsey and Jon Timmis. Immune inspired somatic contiguous hypermutation for function optimisation. In Erick Cantú-Paz, James A. Foster, Kalyanmoy Deb, Lawrence David Davis, Rajkumar Roy, Una-May O’Reilly, Hans-Georg Beyer, Russell Standish, Graham Kendall, Stewart Wilson, Mark Harman, Joachim Wegener, Dipankar Dasgupta, Mitch A. Potter, Alan C. Schultz, Kathryn A. Dowsland, Natasha Jonoska, and Julian Miller, editors, *Genetic and Evolutionary Computation — GECCO 2003*, pages 207–218, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. Cited on page 68.
- [159] S.M. Garrett. Parameter-free, adaptive clonal selection. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 1, pages 1052–1058 Vol.1, 2004. doi:10.1109/CEC.2004.1330978. Cited on page 68.
- [160] N. K. JERNE. Towards a network theory of the immune system. *Ann.Immunol.*, 125C:373–389, 1974. URL: <https://ci.nii.ac.jp/naid/10004787821/en/>. Cited on page 69.
- [161] L. Nunes de Casto and F.J. Von Zuben. An evolutionary immune network for data clustering. In *Proceedings. Vol.1. Sixth Brazilian Symposium on Neural Networks*, pages 84–89, 2000. doi:10.1109/SBRN.2000.889718. Cited on page 69.
- [162] Polly Matzinger. The danger model: A renewed sense of self. *Science*, 296(5566):301–305, 2002. URL: <https://www.science.org/doi/abs/10.1126/science.1071059>, arXiv:<https://www.science.org/doi/pdf/10.1126/science.1071059>, doi:10.1126/science.1071059. Cited on page 70.
- [163] Julie Greensmith, Uwe Aickelin, and Jamie Twycross. Articulation and clarification of the dendritic cell algorithm. In Hugues Bersini and Jorge Carneiro, editors, *Artificial Immune Systems*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. Cited on pages 71, 72, 117, and 125.
- [164] Ralph M. Steinman and Zanvil A. Cohn. IDENTIFICATION OF A NOVEL CELL TYPE IN PERIPHERAL LYMPHOID ORGANS OF MICE : I. MORPHOLOGY, QUANTITATION, TISSUE DISTRIBUTION . *Journal of Experimental Medicine*, 137(5):1142–1162, 05 1973. URL: <https://doi.org/10.1084/jem.137.5.1142>, arXiv:<https://rupress.org/jem/article-pdf/137/5/1142/1085365/1142.pdf>, doi:10.1084/jem.137.5.1142. Cited on page 71.
- [165] Jamie Paul Twycross. *Integrated innate and adaptive artificial immune systems applied to process anomaly detection*. PhD thesis, University of Nottingham, 2007. Cited on pages 72 and 117.

- [166] C.A. Laurentys, R.M. Palhares, and W.M. Caminhas. A novel artificial immune system for fault behavior detection. *Expert Systems with Applications*, 38(6):6957–6966, 2011. URL: <https://www.sciencedirect.com/science/article/pii/S0957417410013825>, doi:<https://doi.org/10.1016/j.eswa.2010.12.019>. Cited on pages 72 and 117.
- [167] Julie Greensmith and Uwe Aickelin. The deterministic dendritic cell algorithm. In Peter J. Bentley, Doheon Lee, and Sungwon Jung, editors, *Artificial Immune Systems*, pages 291–302, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. Cited on pages 72, 75, and 125.
- [168] Zeineb Chelly and Zied Elouedi. A survey of the dendritic cell algorithm. *Knowledge and Information Systems*, 48(3):505–535, 2016. URL: <https://doi.org/10.1007/s10115-015-0891-y>, doi:<https://doi.org/10.1007/s10115-015-0891-y>. Cited on pages 73 and 120.
- [169] Jon Timmis. Artificial immune systems—today and tomorrow. *Natural Computing*, 6(1):1–18, 2007. URL: <https://doi.org/10.1007/s11047-006-9029-1>, doi:<https://doi.org/10.1007/s11047-006-9029-1>. Cited on pages 84 and 85.
- [170] Kamran Shafi and Hussein A. Abbass. Biologically-inspired complex adaptive systems approaches to network intrusion detection. *Information Security Technical Report*, 12(4):209–217, 2007. URL: <https://www.sciencedirect.com/science/article/pii/S1363412707000416>, doi:<https://doi.org/10.1016/j.istr.2007.09.001>. Cited on pages 85, 86, and 106.
- [171] J. Timmis, P. Andrews, N. Owens, and E. Clark. An interdisciplinary perspective on artificial immune systems. *Evolutionary Intelligence*, 1(1):5–26, 2008. URL: <https://doi.org/10.1007/s12065-007-0004-2>, doi:<https://doi.org/10.1007/s12065-007-0004-2>. Cited on pages 84 and 85.
- [172] Jieqiong Zheng, Yunfang Chen, and Wei Zhang. A survey of artificial immune applications. *Artificial Intelligence Review*, 34(1):19–34, 2010. URL: <https://doi.org/10.1007/s10462-010-9159-9>, doi:<https://doi.org/10.1007/s10462-010-9159-9>. Cited on page 85.
- [173] Ahmad Shahrizal Muhamad and Safaai Deris. An artificial immune system for solving production scheduling problems: a review. *Artificial Intelligence Review*, 39(2):97–108, 2013. URL: <https://doi.org/10.1007/s10462-011-9259-1>, doi:<https://doi.org/10.1007/s10462-011-9259-1>. Cited on pages 85 and 86.
- [174] Nawel Bayar, Saber Darmoul, Sonia Hajri-Gabouj, and Henri Pierreval. Fault detection, diagnosis and recovery using artificial immune systems: A review. *Engineering Applications of Artificial Intelligence*, 46:43–57, 2015. URL: <https://www.sciencedirect.com/science/article/pii/S0952197615001840>, doi:<https://doi.org/10.1016/j.engappai.2015.08.006>. Cited on pages 85 and 86.
- [175] Mercy Bere and Hippolyte Muyingi. Initial investigation of industrial control system (ics) security using artificial immune system (ais). In *2015 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, pages 79–84, 2015. doi:<https://doi.org/10.1109/ETNCC.2015.7184812>. Cited on pages 85, 86, and 106.

- [176] Ali Raza and Benito R. Fernandez. Immuno-inspired robotic applications: A review. *Applied Soft Computing*, 37:490–505, 2015. URL: <https://www.sciencedirect.com/science/article/pii/S156849461500561X>, doi:<https://doi.org/10.1016/j.asoc.2015.08.050>. Cited on pages 85 and 86.
- [177] Ray Y. Zhong, Xun Xu, Eberhard Klotz, and Stephen T. Newman. Intelligent manufacturing in the context of industry 4.0: A review. *Engineering*, 3(5):616–630, 2017. URL: <https://www.sciencedirect.com/science/article/pii/S2095809917307130>, doi:<https://doi.org/10.1016/J.ENG.2017.05.015>. Cited on pages 84 and 85.
- [178] Y. J. Qu, X. G. Ming, Z. W. Liu, X. Y. Zhang, and Z. T. Hou. Smart manufacturing systems: state of the art and future trends. *The International Journal of Advanced Manufacturing Technology*, 103(9):3751–3768, 2019. URL: <https://doi.org/10.1007/s00170-019-03754-7>, doi:[10.1007/s00170-019-03754-7](https://doi.org/10.1007/s00170-019-03754-7). Cited on pages 84 and 85.
- [179] Faisal Alkhateeb, Ra’ed M. Al-Khatib, and Iyad Abu Doush. A survey for recent applications and variants of nature-inspired immune search algorithm. *International Journal of Computer Applications in Technology*, 63(4):354–370, 2020. URL: <https://www.inderscienceonline.com/doi/abs/10.1504/IJCAT.2020.110417>, arXiv:<https://www.inderscienceonline.com/doi/pdf/10.1504/IJCAT.2020.110417>, doi:[10.1504/IJCAT.2020.110417](https://doi.org/10.1504/IJCAT.2020.110417). Cited on page 85.
- [180] Petar Radanliev, David De Roure, Max Van Kleek, Omar Santos, and Uchenna Ani. Artificial intelligence in cyber physical systems. *AI & SOCIETY*, 36(3):783–796, 2021. URL: <https://doi.org/10.1007/s00146-020-01049-0>, doi:[10.1007/s00146-020-01049-0](https://doi.org/10.1007/s00146-020-01049-0). Cited on pages 84 and 85.
- [181] Daniel Stock, Thomas Bauernhansl, Michael Weyrich, Matthias Feurer, and Rolf Wutzke. System architectures for cyber-physical production systems enabling self-x and autonomy. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 148–155, 2020. doi:[10.1109/ETFA46521.2020.9212182](https://doi.org/10.1109/ETFA46521.2020.9212182). Cited on pages 84 and 85.
- [182] Hadeel Alrubayyi, Gokop Goteng, Mona Jaber, and James Kelly. Challenges of malware detection in the iot and a review of artificial immune system approaches. *Journal of Sensor and Actuator Networks*, 10(4), 2021. URL: <https://www.mdpi.com/2224-2708/10/4/61>, doi:[10.3390/jsan10040061](https://doi.org/10.3390/jsan10040061). Cited on pages 85, 86, 116, and 123.
- [183] Andre Dionisio Rocha, Pedro Lima-Monteiro, Mafalda Parreira-Rocha, and Jose Barata. Artificial immune systems based multi-agent architecture to perform distributed diagnosis. *Journal of Intelligent Manufacturing*, 30(4):2025–2037, 2019. URL: <https://doi.org/10.1007/s10845-017-1370-y>, doi:[10.1007/s10845-017-1370-y](https://doi.org/10.1007/s10845-017-1370-y). Cited on pages 90 and 96.
- [184] M R Lokesh and Y. S. Kumaraswamy. State awareness towards resiliency in cyber-physical system: A modified danger theory based deterministic dendritic cell algorithm approach. In *2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*, pages 201–208, 2015. doi:[10.1109/CGVIS.2015.7449922](https://doi.org/10.1109/CGVIS.2015.7449922). Cited on pages 90, 99, and 110.

- [185] Sendren Sheng-Dong Xu, Hsu-Chih Huang, Yu-Chieh Kung, and Yung-Yu Chu. A networked multirobot cps with artificial immune fuzzy optimization for distributed formation control of embedded mobile robots. *IEEE Transactions on Industrial Informatics*, 16(1):414–422, 2020. doi:[10.1109/TII.2019.2936045](https://doi.org/10.1109/TII.2019.2936045). Cited on pages 90 and 102.
- [186] Obinna Igbe, Ihab Darwish, and Tarek Saadawi. Deterministic dendritic cell algorithm application to smart grid cyber-attack detection. In *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 199–204, 2017. doi:[10.1109/CSCloud.2017.12](https://doi.org/10.1109/CSCloud.2017.12). Cited on pages 90, 100, 106, 107, 110, 124, and 128.
- [187] G.A. Samigulina and Z.I. Samigulina. Development of smart-technology for complex objects control based on approach of artificial immune systems. In *2018 Global Smart Industry Conference (GloSIC)*, pages 1–6, 2018. doi:[10.1109/GloSIC.2018.8570142](https://doi.org/10.1109/GloSIC.2018.8570142). Cited on pages 90 and 97.
- [188] Lokesh M R and Y.S Kumaraswamy. Modified danger theory based optimized artificial immune network on resiliency in cyber-physical system. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1228–1235, 2015. doi:[10.1109/ICGCIoT.2015.7380651](https://doi.org/10.1109/ICGCIoT.2015.7380651). Cited on pages 90, 96, and 110.
- [189] M R Lokesh and Y.S Kumaraswamy. Healing process towards resiliency in cyber-physical system: A modified danger theory based artificial immune recognition2 algorithm approach. In *2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*, pages 226–232, 2015. doi:[10.1109/CGVIS.2015.7449926](https://doi.org/10.1109/CGVIS.2015.7449926). Cited on pages 90, 96, and 110.
- [190] Fernando P.A. Lima, Anna D.P. Lotufo, and Carlos R. Minussi. Disturbance detection for optimal database storage in electrical distribution systems using artificial immune systems with negative selection. *Electric Power Systems Research*, 109:54–62, 2014. URL: <https://www.sciencedirect.com/science/article/pii/S0378779613003398>, doi:<https://doi.org/10.1016/j.epsr.2013.12.010>. Cited on pages 90 and 99.
- [191] W.D. Li and X.T. Cai. Intelligent immune system for sustainable manufacturing. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, pages 190–195, 2018. doi:[10.1109/CSCWD.2018.8465214](https://doi.org/10.1109/CSCWD.2018.8465214). Cited on pages 90, 102, and 106.
- [192] Fernando P.A. Lima, Anna Diva P. Lotufo, and Carlos Roberto Minussi. Wavelet-artificial immune system algorithm applied to voltage disturbance diagnosis in electrical distribution systems. *IET Generation, Transmission & Distribution*, 9(11):1104–1111, 2015. URL: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-gtd.2014.1102>, arXiv:<https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-gtd.2014.1102>, doi:<https://doi.org/10.1049/iet-gtd.2014.1102>. Cited on pages 90 and 101.
- [193] Fábio Bertequini Leão, Rodrigo A.F. Pereira, and José R.S. Mantovani. Fault section estimation in electric power systems using an optimization immune algorithm. *Electric Power Systems Research*, 80(11):1341–1352, 2010. URL: <https://www.sciencedirect.com/science/article/pii/S0378779610001306>, doi:<https://doi.org/10.1016/j.epsr.2010.05.010>. Cited on pages 90 and 99.

- [194] Albert Ko, H. Y. K. Lau, and T. L. Lau. An immuno control framework for decentralized mechatronic control. In Giuseppe Nicosia, Vincenzo Cutello, Peter J. Bentley, and Jon Timmis, editors, *Artificial Immune Systems*, pages 91–105, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. Cited on pages 90 and 97.
- [195] Franz J. Rammig, Stefan Grösbrink, Katharina Stahl, and Yuhong Zhao. Designing self-adaptive embedded real-time software – towards system engineering of self-adaptation. In *2014 Brazilian Symposium on Computing Systems Engineering*, pages 37–42, 2014. doi: [10.1109/SBESC.2014.15](https://doi.org/10.1109/SBESC.2014.15). Cited on pages 90 and 95.
- [196] Viktoriya Degeler, Richard French, and Kevin Jones. Combined danger signal and anomaly-based threat detection in cyber-physical systems. In Benny Mandler, Johann Marquez-Barja, Miguel Elias Mitre Campista, Dagmar Cagáňová, Hakima Chaouchi, Sherali Zeadally, Mohamad Badra, Stefano Giordano, Maria Fazio, Andrey Somov, and Radu-Laurentiu Vieriu, editors, *Internet of Things. IoT Infrastructures*, pages 27–39, Cham, 2016. Springer International Publishing. Cited on pages 90, 100, 106, and 124.
- [197] Xavier Clotet, José Moyano, and Gladys León. A real-time anomaly-based ids for cyber-attack detection at the industrial process level of critical infrastructures. *International Journal of Critical Infrastructure Protection*, 23:11–20, 2018. URL: <https://www.sciencedirect.com/science/article/pii/S1874548217300884>, doi:<https://doi.org/10.1016/j.ijcip.2018.08.002>. Cited on pages 90, 100, 106, and 110.
- [198] Carlos A.V. Guerrero, Paulo M. Silveira, and José M.C. Filho. Adaptation of the clonal selection algorithm to the real-time coordinated volt/var control through a software-in-the-loop strategy. *Electric Power Systems Research*, 194:107092, 2021. URL: <https://www.sciencedirect.com/science/article/pii/S0378779621000730>, doi:<https://doi.org/10.1016/j.epsr.2021.107092>. Cited on pages 90 and 98.
- [199] Jianhua Guo and Haidong Yang. An anti-jamming artificial immune approach for energy leakage diagnosis in parallel-machine job shops. *Computers in Industry*, 101:13–24, 2018. URL: <https://www.sciencedirect.com/science/article/pii/S016636151730756X>, doi:<https://doi.org/10.1016/j.compind.2018.05.004>. Cited on pages 90, 99, and 106.
- [200] Tushar Semwal and Shivashankar B. Nair. A decentralized artificial immune system for solution selection in cyber-physical systems. *Applied Soft Computing*, 86:105920, 2020. URL: <https://www.sciencedirect.com/science/article/pii/S156849461930701X>, doi:<https://doi.org/10.1016/j.asoc.2019.105920>. Cited on pages 90 and 102.
- [201] Jianfeng Zhu, Yidan Shu, Jinsong Zhao, and Fan Yang. A dynamic alarm management strategy for chemical process transitions. *Journal of Loss Prevention in the Process Industries*, 30:207–218, 2014. URL: <https://www.sciencedirect.com/science/article/pii/S0950423013001393>, doi:<https://doi.org/10.1016/j.jlpp.2013.07.008>. Cited on pages 90 and 101.
- [202] Jinsong Zhao, Yidan Shu, Jianfeng Zhu, and Yiyang Dai. An online fault diagnosis strategy for full operating cycles of chemical processes. *Industrial & Engineering Chemistry Research*, 53(13):5015–5027, 2014. URL: <https://doi.org/10.1021/>

- ie400660e, arXiv:<https://doi.org/10.1021/ie400660e>, doi:10.1021/ie400660e. Cited on pages 90 and 101.
- [203] Dong Hwa Kim. Tuning of 2-dof pid controller by immune algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 1, pages 675–680 vol.1, 2002. doi:10.1109/CEC.2002.1007007. Cited on pages 90 and 97.
- [204] X. D. Li, Y. Q. Xu, and L. Zhang. Distribution service restoration with dgs based on multi-agent immune algorithm. In *2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS)*, volume 1, pages 1–4, 2009. doi:10.1109/PEITS.2009.5407060. Cited on pages 90 and 96.
- [205] M. R. Aghaebrahimi, M. Amiri, and S. H. Zahiri. An immune-based optimization method for distributed generation placement in order to minimize power losses. In *2009 International Conference on Sustainable Power Generation and Supply*, pages 1–6, 2009. doi:10.1109/SUPERGEN.2009.5348247. Cited on pages 90 and 102.
- [206] Diego Lizondo, Pedro Araujo, Adrian Will, and Sebastian Rodriguez. Multiagent model for distributed peak shaving system with demand-side management approach. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 352–357, 2017. doi:10.1109/IRC.2017.50. Cited on pages 90, 98, and 106.
- [207] M. Kayama, Y. Sugita, Y. Morooka, and S. Fukuoka. Distributed diagnosis system combining the immune network and learning vector quantization. In *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, volume 2, pages 1531–1536 vol.2, 1995. doi:10.1109/IECON.1995.484178. Cited on pages 90 and 101.
- [208] Bin Wang, Sun-An Wang, and Jian Zhuang. A distributed immune algorithm for learning experience in complex industrial process control. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, volume 4, pages 2138–2141 Vol.4, 2003. doi:10.1109/ICMLC.2003.1259859. Cited on pages 90 and 97.
- [209] Jin-Hyung Jun, Dong-Wook Lee, and Kwee-Bo Sim. Realization of cooperative strategies and swarm behavior in distributed autonomous robotic systems using artificial immune system. In *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, volume 6, pages 614–619 vol.6, 1999. doi:10.1109/ICSMC.1999.816622. Cited on pages 90 and 96.
- [210] X. Jin, J. Zhao, and H. F. Wang. On-line stability control of power systems integrated with distributed generation systems. In *Proceedings of the 41st International Universities Power Engineering Conference*, volume 2, pages 472–476, 2006. doi:10.1109/UPEC.2006.367522. Cited on pages 90 and 98.
- [211] Hsu-Chih Huang. An evolutionary optimal fuzzy system with information fusion of heterogeneous distributed computing and polar-space dynamic model for online motion control of swedish redundant robots. *IEEE Transactions on Industrial Electronics*, 64(2):1743–1750, 2017. doi:10.1109/TIE.2016.2562613. Cited on pages 90 and 97.
- [212] H.Y.K. Lau and V.W.K. Wong. A strategic behavioral-based intelligent transport system with artificial immune system. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 4, pages 3909–3914 vol.4, 2004. doi:10.1109/ICSMC.2004.1400955. Cited on pages 90 and 97.

- [213] Tang Xiaobo and Tang Guoqing. Risk distribution network planning including distributed generation based on particle swarm optimization algorithm with immunity. In *2009 International Conference on Sustainable Power Generation and Supply*, pages 1–5, 2009. doi:[10.1109/SUPERGEN.2009.5348246](https://doi.org/10.1109/SUPERGEN.2009.5348246). Cited on pages 90, 102, and 106.
- [214] Dong-Wook Lee and Kwee-Bo Sim. Artificial immune network-based cooperative control in collective autonomous mobile robots. In *Proceedings 6th IEEE International Workshop on Robot and Human Communication. RO-MAN'97 SENDAI*, pages 58–63, 1997. doi:[10.1109/ROMAN.1997.646953](https://doi.org/10.1109/ROMAN.1997.646953). Cited on pages 90 and 96.
- [215] H.Y.K. Lau and A.K.S. Ng. Immunology-based control framework for multi-jointed redundant manipulators. In *IEEE Conference on Robotics, Automation and Mechatronics, 2004.*, volume 1, pages 318–323 vol.1, 2004. doi:[10.1109/RAMECH.2004.1438938](https://doi.org/10.1109/RAMECH.2004.1438938). Cited on pages 90 and 97.
- [216] Guiliang Yin. A distributed generation islanding detection method based on artificial immune system. In *2005 IEEE/PES Transmission Distribution Conference Exposition: Asia and Pacific*, pages 1–4, 2005. doi:[10.1109/TDC.2005.1547072](https://doi.org/10.1109/TDC.2005.1547072). Cited on pages 90, 98, and 106.
- [217] R. Bhuvaneswari, S.K. Srivastava, C.S. Edrington, D.A. Cartes, and S. Subramanian. Intelligent agent based auction by economic generation scheduling for microgrid operation. In *2010 Innovative Smart Grid Technologies (ISGT)*, pages 1–6, 2010. doi:[10.1109/ISGT.2010.5434745](https://doi.org/10.1109/ISGT.2010.5434745). Cited on pages 90, 98, and 106.
- [218] Jing Yuan, Aihong Tang, Xiaohui Zhai, Hui Yan, Xu Zheng, and Hongsheng Zhao. Co-ordination control strategy of dssc converter based on multi-objective optimal immune algorithms. In *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pages 1987–1991, 2019. doi:[10.1109/IMCEC46724.2019.8983846](https://doi.org/10.1109/IMCEC46724.2019.8983846). Cited on pages 90 and 102.
- [219] H.Y.K. Lau and V.W.K. Wong. An immunity-based distributed multiagent-control framework. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 36(1):91–108, 2006. doi:[10.1109/TSMCA.2005.859103](https://doi.org/10.1109/TSMCA.2005.859103). Cited on pages 90 and 97.
- [220] R. Michelan and F.J. Von Zuben. Decentralized control system for autonomous navigation based on an evolved artificial immune network. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1021–1026 vol.2, 2002. doi:[10.1109/CEC.2002.1004383](https://doi.org/10.1109/CEC.2002.1004383). Cited on pages 90 and 97.
- [221] B. Hanumantha Rao and S. Sivanagaraju. Optimum allocation and sizing of distributed generations based on clonal selection algorithm for loss reduction and technical benefit of energy savings. In *2012 International Conference on Advances in Power Conversion and Energy Technologies (APCET)*, pages 1–5, 2012. doi:[10.1109/APCET.2012.6302004](https://doi.org/10.1109/APCET.2012.6302004). Cited on pages 90 and 102.
- [222] Yunyuan Gao and Zhizeng Luo. Dynamic task allocation method based on immune system for cooperative robots. In *2008 7th World Congress on Intelligent Control and Automation*, pages 1015–1020, 2008. doi:[10.1109/WCICA.2008.4593060](https://doi.org/10.1109/WCICA.2008.4593060). Cited on pages 90 and 102.

- [223] Sang-Joon Sun, Dong-Wook Lee, and Kwee-Bo Sim. Artificial immune-based swarm behaviors of distributed autonomous robotic systems. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 4, pages 3993–3998 vol.4, 2001. doi:[10.1109/ROBOT.2001.933241](https://doi.org/10.1109/ROBOT.2001.933241). Cited on pages 90 and 96.
- [224] Arjan N. Rimal and Rabie Belkacemi. Cps compliant adaptive immune based load frequency control with varying wind penetrations. In *2016 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pages 1–5, 2016. doi:[10.1109/ISGT.2016.7781032](https://doi.org/10.1109/ISGT.2016.7781032). Cited on pages 90 and 98.
- [225] Alberto Diez-Olivan, Patxi Ortego, Javier Del Ser, Itziar Landa-Torres, Diego Galar, David Camacho, and Basilio Sierra. Adaptive dendritic cell-deep learning approach for industrial prognosis under changing conditions. *IEEE Transactions on Industrial Informatics*, 17(11):7760–7770, 2021. doi:[10.1109/TII.2021.3058350](https://doi.org/10.1109/TII.2021.3058350). Cited on pages 90 and 99.
- [226] Maryam Khoie, Karim Salahshoor, Ehsan Nouri, and Ali Khaki Sedigh. Pid controller tuning using multi-objective optimization based on fused genetic-immune algorithm and immune feedback mechanism. In De-Shuang Huang, Yong Gan, Phalguni Gupta, and M. Michael Gromiha, editors, *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, pages 267–276, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. Cited on pages 90 and 97.
- [227] Yun Ji Kim, Weonwoo Nam, and Jongsoo Lee. Multiclass anomaly detection for unsupervised and semi-supervised data based on a combination of negative selection and clonal selection algorithms. *Applied Soft Computing*, 122:108838, 2022. URL: <https://www.sciencedirect.com/science/article/pii/S1568494622002332>, doi:<https://doi.org/10.1016/j.asoc.2022.108838>. Cited on pages 90 and 99.
- [228] Roberto Outa, Fábio Roberto Chavarette, Aparecido Carlos Gonçalves, Sidney Leal da Silva, Vishnu Narayan Mishra, Alan Rodrigo Panosso, and Lakshmi Narayan Mishra. Reliability analysis using experimental statistical methods and ais: application in continuous flow tubes of gaseous medium. *Acta Scientiarum. Technology*, 43(1):e55825, Sep. 2021. URL: <https://periodicos.uem.br/ojs/index.php/ActaSciTechnol/article/view/55825>, doi:[10.4025/actascitechnol.v43i1.55825](https://doi.org/10.4025/actascitechnol.v43i1.55825). Cited on pages 90 and 101.
- [229] Hongmei He, Carsten Maple, Tim Watson, Ashutosh Tiwari, Jörn Mehnen, Yaochu Jin, and Bogdan Gabrys. The security challenges in the iot enabled cyber-physical systems and opportunities for evolutionary computing and other computational intelligence. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1015–1021, 2016. doi:[10.1109/CEC.2016.7743900](https://doi.org/10.1109/CEC.2016.7743900). Cited on page 106.
- [230] Wei Dong Leong, Hon Loong Lam, Wendy Pei Qin Ng, Chun Hsion Lim, Chee Pin Tan, and Sivalinga Govinda Ponnambalam. Lean and green manufacturing—a review on its applications and impacts. *Process Integration and Optimization for Sustainability*, 3(1):5–23, 2019. URL: <https://doi.org/10.1007/s41660-019-00082-x>, doi:[10.1007/s41660-019-00082-x](https://doi.org/10.1007/s41660-019-00082-x). Cited on page 107.
- [231] Wadhah Abualfaraa, Konstantinos Salonitis, Ahmed Al-Ashaab, and Maher Ala’raj. Lean-green manufacturing practices and their link with sustainability: A critical review. *Sustainability*, 12(3), 2020. URL: <https://www.mdpi.com/2071-1050/12/3/981>, doi:[10.3390/su12030981](https://doi.org/10.3390/su12030981). Cited on page 107.

- [232] Sarvesh P.S. Rajput and Suprabeet Datta. Sustainable and green manufacturing – a narrative literature review. *Materials Today: Proceedings*, 26:2515–2520, 2020. 10th International Conference of Materials Processing and Characterization. URL: <https://www.sciencedirect.com/science/article/pii/S2214785320312906>, doi:<https://doi.org/10.1016/j.matpr.2020.02.535>. Cited on page 107.
- [233] Dietrich Brandt, Inga Tschiersch, and Klaus Henning. The design of human-centered manufacturing systems. In *The Design of Manufacturing Systems*, pages 5–1. CRC Press, 2019. Cited on page 108.
- [234] Tamás Ruppert, Szilárd Jaskó, Tibor Holczinger, and János Abonyi. Enabling technologies for operator 4.0: A survey. *Applied Sciences*, 8(9), 2018. URL: <https://www.mdpi.com/2076-3417/8/9/1650>, doi:[10.3390/app8091650](https://doi.org/10.3390/app8091650). Cited on page 108.
- [235] A. Hariharasudan and Sebastian Kot. A scoping review on digital english and education 4.0 for industry 4.0. *Social Sciences*, 7(11), 2018. URL: <https://www.mdpi.com/2076-0760/7/11/227>, doi:[10.3390/socsci7110227](https://doi.org/10.3390/socsci7110227). Cited on page 108.
- [236] Pierre Parrend, Fabio Guigou, Julio Navarro, Aline Deruyver, and Pierre Collet. For a refoundation of artificial immune system research: Ais is a design pattern. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1122–1129, 2018. doi:[10.1109/SSCI.2018.8628868](https://doi.org/10.1109/SSCI.2018.8628868). Cited on page 108.
- [237] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018. doi:[10.1109/ACCESS.2018.2870052](https://doi.org/10.1109/ACCESS.2018.2870052). Cited on page 109.
- [238] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. In Grigore Havarneanu, Roberto Setola, Hypatia Nassopoulos, and Stephen Wolthusen, editors, *Critical Information Infrastructures Security*, pages 88–99, Cham, 2017. Springer International Publishing. Cited on page 110.
- [239] Iurii D. Katser and Vyacheslav O. Kozitsin. Skoltech anomaly benchmark (skab). <https://www.kaggle.com/dsv/1693952>, 2020. [Online; accessed July 2022]. doi:[10.34740/KAGGLE/DSV/1693952](https://doi.org/10.34740/KAGGLE/DSV/1693952). Cited on page 111.
- [240] Pls Jayalaxmi, Rahul Saha, Gulshan Kumar, Neeraj Kumar, and Tai-Hoon Kim. A taxonomy of security issues in industrial internet-of-things: Scoping review for existing solutions, future implications, and research challenges. *IEEE Access*, 9:25344–25359, 2021. doi:[10.1109/ACCESS.2021.3057766](https://doi.org/10.1109/ACCESS.2021.3057766). Cited on page 114.
- [241] Keith Stouffer, Victoria Pillitteri, Suzanne Lightman, Marshall Abrams, and Adam Hahn. Guide to industrial control systems (ics) security, 2015-06-03 2015. doi:<https://doi.org/10.6028/NIST.SP.800-82r2>. Cited on page 114.
- [242] Yan Hu, An Yang, Hong Li, Yuyan Sun, and Limin Sun. A survey of intrusion detection on industrial control systems. *International Journal of Distributed Sensor Networks*, 14(8):1550147718794615, 2018. URL: <https://doi.org/10.1177/1550147718794615>, arXiv:<https://arxiv.org/abs/10.1177/1550147718794615>, doi:[10.1177/1550147718794615](https://doi.org/10.1177/1550147718794615). Cited on pages 114 and 115.

- [243] Mohamad Kaouk, Jean-Marie Flaus, Marie-Laure Potet, and Roland Groz. A review of intrusion detection systems for industrial control systems. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1699–1704, 2019. doi:[10.1109/CoDIT.2019.8820602](https://doi.org/10.1109/CoDIT.2019.8820602). Cited on page 115.
- [244] Song Han, Miao Xie, Hsiao-Hwa Chen, and Yun Ling. Intrusion detection in cyber-physical systems: Techniques and challenges. *IEEE Systems Journal*, 8(4):1052–1062, 2014. doi:[10.1109/JSYST.2013.2257594](https://doi.org/10.1109/JSYST.2013.2257594). Cited on page 115.
- [245] Juan E. Rubio, Cristina Alcaraz, Rodrigo Roman, and Javier Lopez. Analysis of intrusion detection systems in industrial ecosystems. In *14th International Conference on Security and Cryptography (SECRYPT 2017)*, volume 6, pages 116–128. SciTePress, SciTePress, 2017. doi:[10.5220/0006426301160128](https://doi.org/10.5220/0006426301160128). Cited on page 115.
- [246] Nilufer Tuptuk and Stephen Hailes. Security of smart manufacturing systems. *Journal of Manufacturing Systems*, 47:93–106, 2018. URL: <https://www.sciencedirect.com/science/article/pii/S0278612518300463>, doi:<https://doi.org/10.1016/j.jmsy.2018.04.007>. Cited on page 115.
- [247] SN Sivanandam and SN Deepa. *Principles of soft computing (with CD)*. John Wiley & Sons, 2007. Cited on page 116.
- [248] Uwe Aickelin, Julie Greensmith, and Jamie Twycross. Immune system approaches to intrusion detection – a review. In Giuseppe Nicosia, Vincenzo Cutello, Peter J. Bentley, and Jon Timmis, editors, *Artificial Immune Systems*, pages 316–329, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. Cited on page 116.
- [249] Farhoud Hosseinpour, Kamalrulnizam Abu Bakar, Amir Hatami Hardoroudi, and Nazanin-sadat Kazazi. Survey on artificial immune system as a bio-inspired technique for anomaly based intrusion detection systems. In *2010 International Conference on Intelligent Networking and Collaborative Systems*, pages 323–324, 2010. doi:[10.1109/INCOS.2010.40](https://doi.org/10.1109/INCOS.2010.40). Cited on page 116.
- [250] Vyacheslav Tokarev, Alexey Sychugov, and Alexander Anchishkin. Detection of anomalies in the information networks of industrial automation systems based on artificial immune detectors. In *2019 International Russian Automation Conference (RusAutoCon)*, pages 1–5, 2019. doi:[10.1109/RUSAUTOCON.2019.8867593](https://doi.org/10.1109/RUSAUTOCON.2019.8867593). Cited on page 116.
- [251] Guilherme Costa Silva, Reinaldo M. Palhares, and Walmir M. Caminhas. A transitional view of immune inspired techniques for anomaly detection. In Hujun Yin, José A. F. Costa, and Guilherme Barreto, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, pages 568–577, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. Cited on page 116.
- [252] Johannes Textor. A comparative study of negative selection based anomaly detection in sequence data. In Carlos A. Coello Coello, Julie Greensmith, Natalio Krasnogor, Pietro Liò, Giuseppe Nicosia, and Mario Pavone, editors, *Artificial Immune Systems*, pages 28–41, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. Cited on page 116.
- [253] Julie Greensmith, Uwe Aickelin, and Gianni Tedesco. Information fusion for anomaly detection with the dendritic cell algorithm. *Information Fusion*, 11(1):21–34, 2010. Special Issue on Biologically-Inspired Information Fusion. URL: <https://doi.org/10.1016/j.inffus.2010.01.001>.

- [//www.sciencedirect.com/science/article/pii/S1566253509000384](http://www.sciencedirect.com/science/article/pii/S1566253509000384), doi:<https://doi.org/10.1016/j.inffus.2009.04.006>. Cited on page 116.
- [254] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod. Danger theory: The link between ais and ids? In Jon Timmis, Peter J. Bentley, and Emma Hart, editors, *Artificial Immune Systems*, pages 147–155, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. Cited on page 117.
- [255] Mark Vella, Marc Roper, and Sotirios Terzis. Danger theory and intrusion detection: Possibilities and limitations of the analogy. In Emma Hart, Chris McEwan, Jon Timmis, and Andy Hone, editors, *Artificial Immune Systems*, pages 276–289, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. Cited on pages 118 and 119.
- [256] Feng Gu. *Theoretical and empirical extensions of the dendritic cell algorithm*. PhD thesis, University of Nottingham, 2011. Cited on pages 120, 129, 149, 150, and 152.
- [257] Zeineb Chelly and Zied Elouedi. From the general to the specific: Inducing a novel dendritic cell algorithm from a detailed state-of-the-art review. *International Journal of Pattern Recognition and Artificial Intelligence*, 30(03):1659009, 2016. Cited on page 120.
- [258] Rodrigo Roman, Ruben Rios, Jose A. Onieva, and Javier Lopez. Immune system for the internet of things using edge technologies. *IEEE Internet of Things Journal*, 6(3):4774–4781, 2019. doi:[10.1109/JIOT.2018.2867613](https://doi.org/10.1109/JIOT.2018.2867613). Cited on page 123.
- [259] Marin Emilov Pamukov. Application of artificial immune systems for the creation of iot intrusion detection systems. In *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, volume 1, pages 564–568, 2017. doi:[10.1109/IDAACS.2017.8095144](https://doi.org/10.1109/IDAACS.2017.8095144). Cited on page 123.
- [260] Jungwon Kim, Peter Bentley, Christian Wallenta, Mohamed Ahmed, and Stephen Hailes. Danger is ubiquitous: Detecting malicious activities in sensor networks using the dendritic cell algorithm. In Hugues Bersini and Jorge Carneiro, editors, *Artificial Immune Systems*, pages 390–403, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. Cited on page 123.
- [261] Caiming Liu, Jin Yang, Run Chen, Yan Zhang, and Jinquan Zeng. Research on immunity-based intrusion detection technology for the internet of things. In *2011 Seventh International Conference on Natural Computation*, volume 1, pages 212–216, 2011. doi:[10.1109/ICNC.2011.6022060](https://doi.org/10.1109/ICNC.2011.6022060). Cited on page 123.
- [262] Caiming Liu, Yan Zhang, and Huaqiang Zhang. A novel approach to iot security based on immunology. In *2013 Ninth International Conference on Computational Intelligence and Security*, pages 771–775, 2013. doi:[10.1109/CIS.2013.168](https://doi.org/10.1109/CIS.2013.168). Cited on page 123.
- [263] Pierre Parrend, Pierre David, Fabio Guigou, Cédric Pupka, and Pierre Collet. The awa artificial emergent awareness architecture model for artificial immune ecosystems. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 403–410, 2017. doi:[10.1109/CEC.2017.7969340](https://doi.org/10.1109/CEC.2017.7969340). Cited on page 123.
- [264] Sahar Aldhaheri, Daniyal Alghazzawi, Li Cheng, Bander Alzahrani, and Abdullah Al-Barakati. Deepdca: Novel network-based detection of iot attacks using artificial immune system. *Applied Sciences*, 10(6), 2020. URL: <https://www.mdpi.com/2076-3417/10/6/1909>, doi:[10.3390/app10061909](https://doi.org/10.3390/app10061909). Cited on pages 123 and 129.

- [265] Julie Greensmith. *The dendritic cell algorithm*. PhD thesis, University of Nottingham, 2007. Cited on pages 125, 150, and 152.
- [266] Feng Gu, Julie Greensmith, and Uwe Aickelin. Theoretical formulation and analysis of the deterministic dendritic cell algorithm. *Biosystems*, 111(2):127–135, 2013. URL: <https://www.sciencedirect.com/science/article/pii/S0303264713000063>, doi:<https://doi.org/10.1016/j.biosystems.2013.01.001>. Cited on pages 125 and 133.
- [267] Julie Greensmith and Michael B. Gale. The functional dendritic cell algorithm: A formal specification with haskell. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1787–1794, 2017. doi:[10.1109/CEC.2017.7969518](https://doi.org/10.1109/CEC.2017.7969518). Cited on pages 125 and 133.
- [268] Feng Gu, Julie Greensmith, and Uwe Aickelin. Integrating real-time analysis with the dendritic cell algorithm through segmentation. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO '09*, page 1203–1210, New York, NY, USA, 2009. Association for Computing Machinery. URL: <https://doi.org/10.1145/1569901.1570063>, doi:[10.1145/1569901.1570063](https://doi.org/10.1145/1569901.1570063). Cited on pages 126, 127, and 133.
- [269] Julie Greensmith and Uwe Aickelin. Dendritic cells for syn scan detection. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07*, page 49–56, New York, NY, USA, 2007. Association for Computing Machinery. URL: <https://doi.org/10.1145/1276958.1276966>, doi:[10.1145/1276958.1276966](https://doi.org/10.1145/1276958.1276966). Cited on page 126.
- [270] Song Yuan and Qi-juan Chen. A dendritic cell algorithm for real-time anomaly detection. In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, volume 1, pages 448–451, 2012. doi:[10.1109/CSAE.2012.6272635](https://doi.org/10.1109/CSAE.2012.6272635). Cited on pages 126 and 133.
- [271] UCI Machine Learning Repository. Breast Cancer Wisconsin (Diagnostic) Data Set. [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)), November 1995. [Online; accessed July 2022]. Cited on page 127.
- [272] Simon Duque Antón, Michael Gundall, Daniel Fraunholz, and Hans Dieter Schotten. Implementing scada scenarios and introducing attacks to obtain training data for intrusion detection methods. In *ICCWS 2019 14th International Conference on Cyber Warfare and Security: ICCWS 2019*, page 56. Academic Conferences and publishing limited, 2019. Cited on pages 127, 128, and 133.
- [273] Mahbod Tavallaei, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6, 2009. doi:[10.1109/CISDA.2009.5356528](https://doi.org/10.1109/CISDA.2009.5356528). Cited on page 127.
- [274] Thomas Morris and Wei Gao. Industrial control system traffic data sets for intrusion detection research. In Jonathan Butts and Sujeet Sheno, editors, *Critical Infrastructure Protection VIII*, pages 65–78, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. Cited on page 127.

- [275] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, nov 2009. URL: <https://doi.org/10.1145/1656274.1656278>, doi:10.1145/1656274.1656278. Cited on page 128.
- [276] Antoine Lemay and Jose M. Fernandez. Providing SCADA network data sets for intrusion detection research. In *9th Workshop on Cyber Security Experimentation and Test (CSET 16)*, Austin, TX, August 2016. USENIX Association. URL: <https://www.usenix.org/conference/cset16/workshop-program/presentation/lemay>. Cited on page 128.
- [277] Simon D. Duque Anton, Sapna Sinha, and Hans Dieter Schotten. Anomaly-based intrusion detection in industrial data with svm and random forests. In *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6, 2019. doi:10.23919/SOFTCOM.2019.8903672. Cited on page 128.
- [278] Feng Gu, Julie Greensmith, Robert Oates, and Uwe Aickelin. Pca 4 dca: The application of principal component analysis to the dendritic cell algorithm. Available at SSRN 2830357, 2009. Cited on pages 129, 133, and 161.
- [279] Zeineb Chelly and Zied Elouedi. Rst-dca: A dendritic cell algorithm based on rough set theory. In Tingwen Huang, Zhigang Zeng, Chuandong Li, and Chi Sing Leung, editors, *Neural Information Processing*, pages 480–487, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. Cited on pages 129 and 133.
- [280] Zeineb Chelly and Zied Elouedi. Rc-dca: A new feature selection and signal categorization technique for the dendritic cell algorithm based on rough set theory. In Carlos A. Coello Coello, Julie Greensmith, Natalio Krasnogor, Pietro Liò, Giuseppe Nicosia, and Mario Pavone, editors, *Artificial Immune Systems*, pages 152–165, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. Cited on pages 129 and 133.
- [281] Zeineb Chelly and Zied Elouedi. Qr-dca: A new rough data pre-processing approach for the dendritic cell algorithm. In Marco Tomassini, Alberto Antonioni, Fabio Daolio, and Pierre Buesser, editors, *Adaptive and Natural Computing Algorithms*, pages 140–150, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. Cited on pages 129, 130, and 133.
- [282] Zeineb Chelly and Zied Elouedi. A new data pre-processing approach for the dendritic cell algorithm based on fuzzy rough set theory. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '13 Companion*, page 163–164, New York, NY, USA, 2013. Association for Computing Machinery. URL: <https://doi.org/10.1145/2464576.2464657>, doi:10.1145/2464576.2464657. Cited on pages 130 and 133.
- [283] Zeineb Chelly and Zied Elouedi. Supporting fuzzy-rough sets in the dendritic cell algorithm data pre-processing phase. In Minhoo Lee, Akira Hirose, Zeng-Guang Hou, and Rhee Man Kil, editors, *Neural Information Processing*, pages 164–171, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. Cited on pages 130 and 133.
- [284] Zeineb Chelly and Zied Elouedi. A fuzzy-rough data pre-processing approach for the dendritic cell classifier. In Linda C. van der Gaag, editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 109–120, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. Cited on pages 130 and 133.

- [285] Wen Zhou and Yiwen Liang. A new version of the deterministic dendritic cell algorithm based on numerical differential and immune response. *Applied Soft Computing*, 102:107055, 2021. URL: <https://www.sciencedirect.com/science/article/pii/S1568494620309935>, doi:<https://doi.org/10.1016/j.asoc.2020.107055>. Cited on pages 130 and 133.
- [286] Valeriy Vyatkin. Software engineering in industrial automation: State-of-the-art review. *IEEE Transactions on Industrial Informatics*, 9(3):1234–1249, 2013. doi:[10.1109/TII.2013.2258165](https://doi.org/10.1109/TII.2013.2258165). Cited on page 130.
- [287] Per Lindgren, Marcus Lindner, Andreas Lindner, Johan Eriksson, and Valeriy Vyatkin. Real-time execution of function blocks for internet of things using the rtfm-kernel. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–6, 2014. doi:[10.1109/ETFA.2014.7005232](https://doi.org/10.1109/ETFA.2014.7005232). Cited on page 131.
- [288] Muthukumar N., Seshadhri Srinivasan, K. Ramkumar, Deepak Pal, Juri Vain, and Srin Ramaswamy. A model-based approach for design and verification of industrial internet of things. *Future Generation Computer Systems*, 95:354–363, 2019. URL: <https://www.sciencedirect.com/science/article/pii/S0167739X18321137>, doi:<https://doi.org/10.1016/j.future.2018.12.012>. Cited on page 131.
- [289] Seppo Sierla, Marcin Hurkala, Konstantinia Charitoudi, Chen-Wei Yang, and Valeriy Vyatkin. Security risk analysis for smart grid automation. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pages 1737–1744, 2014. doi:[10.1109/ISIE.2014.6864877](https://doi.org/10.1109/ISIE.2014.6864877). Cited on pages 131 and 226.
- [290] Awais Tanveer, Roopak Sinha, and Stephen G. MacDonell. On design-time security in iec 61499 systems: Conceptualisation, implementation, and feasibility. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 778–785, 2018. doi:[10.1109/INDIN.2018.8472093](https://doi.org/10.1109/INDIN.2018.8472093). Cited on pages 131 and 226.
- [291] Awais Tanveer, Roopak Sinha, Stephen G. MacDonell, Paulo Leitao, and Valeriy Vyatkin. Designing actively secure, highly available industrial automation applications. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 374–379, 2019. doi:[10.1109/INDIN41052.2019.8972262](https://doi.org/10.1109/INDIN41052.2019.8972262). Cited on pages 131 and 226.
- [292] Jack Koziol. *Intrusion detection with Snort*. Sams Publishing, 2003. Cited on page 132.
- [293] Barry Dowdeswell, Roopak Sinha, and Stephen G. MacDonell. Diagnosable-by-design model-driven development for iec 61499 industrial cyber-physical systems. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pages 2183–2188, 2020. doi:[10.1109/IECON43393.2020.9254620](https://doi.org/10.1109/IECON43393.2020.9254620). Cited on pages 132 and 226.
- [294] Barry Dowdeswell, Roopak Sinha, Dennis Jarvis, Jacqueline Jarvis, and Stephen G. MacDonell. Employing agent beliefs during fault diagnosis for iec 61499 industrial cyber-physical systems. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pages 2189–2194, 2020. doi:[10.1109/IECON43393.2020.9254877](https://doi.org/10.1109/IECON43393.2020.9254877). Cited on pages 132 and 226.
- [295] Awais Tanveer, Roopak Sinha, and Matthew M. Y. Kuo. Secure links: Secure-by-design communications in iec 61499 industrial control applications. *IEEE Transactions on Industrial Informatics*, 17(6):3992–4002, 2021. doi:[10.1109/TII.2020.3009133](https://doi.org/10.1109/TII.2020.3009133). Cited on pages 132, 189, and 226.

- [296] KimiNewt. Python packet parser using wireshark's tshark. <https://kiminewt.github.io/pyshark/>, 2020. [Online; accessed July 2022]. Cited on pages 138 and 155.
- [297] FreeOpcUa. FreeOpcUa: Open Source C++ and Python OPC-UA Server and Client Libraries and Tools. <https://freeopcua.github.io/>, 2020. [Online; accessed July 2022]. Cited on page 140.
- [298] Philippe BIONDI. Network packet manipulation with scapy. https://scapy.net/conf/scapy_hack.lu.pdf, 2007. [Online; accessed July 2022]. Cited on page 141.
- [299] Peter Wegner. Concepts and paradigms of object-oriented programming. *SIGPLAN OOPS Mess.*, 1(1):7–87, aug 1990. URL: <https://doi.org/10.1145/382192.383004>, doi:10.1145/382192.383004. Cited on page 142.
- [300] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. River: machine learning for streaming data in python. *Journal of Machine Learning Research*, 22(110):1–8, 2021. URL: <http://jmlr.org/papers/v22/20-1380.html>. Cited on pages 145, 149, and 155.
- [301] Alexander Gepperth and Barbara Hammer. Incremental learning algorithms and applications. In *European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2016. URL: <https://hal.archives-ouvertes.fr/hal-01418129>. Cited on page 146.
- [302] Monowar H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Survey on incremental approaches for network anomaly detection, 2012. [arXiv:1211.4493](https://arxiv.org/abs/1211.4493). Cited on page 147.
- [303] Yuna Sugianela and Tohari Ahmad. Pearson correlation attribute evaluation-based feature selection for intrusion detection system. In *2020 International Conference on Smart Technology and Applications (ICoSTA)*, pages 1–5, 2020. doi:10.1109/ICoSTA48221.2020.1570613717. Cited on page 150.
- [304] Nigel Griffiths. nmon: A free tool to analyze aix and linux performance. https://developer.ibm.com/articles/au-nmon_analyser/, 2020. [Online; accessed July 2022]. Cited on page 187.