FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Engineering Sustainable and Adaptive Systems in Dynamic Environments

Rui Pedro Peixoto Cardoso

MSc DISSERTATION

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rosaldo J.F. Rossetti, PhD

External Supervisor: Jeremy V. Pitt, PhD

July 25, 2018

Engineering Sustainable and Adaptive Systems in Dynamic Environments

Rui Pedro Peixoto Cardoso

Mestrado Integrado em Engenharia Informática e Computação

July 25, 2018

Abstract

Electronic institutions are socially-inspired multi-agent systems, typically operating under a set of policies, which are required to determine system operation and to prevent violations and other non-compliant behaviour. They are often faced with a dynamic environment, potentially undergoing unpredictable changes, and their policy should suit it. However, there is usually a large space of possible system policies, but no systematic method to find an appropriate policy given a joint state of the population, social network, and the environment.

Several recent approaches have been proposed in the literature for dealing with potentially unpredictable changes in the environment. Some work has been successful in drawing concepts and principles from social, political, and economic science and applying them to electronic institutions, operationalising mechanisms which enable the agents to collectively modify system policies. Other work, inspired by Biology, has proposed evolutionary computing techniques which enable adaptation of the agents in a system as a response to environmental change. Drawing motivation from the above-mentioned socially- and biologically-inspired approaches, our top-level goal is to explore how adaptation through evolution of policies can assist the design of collective adaptive systems which remain sustainable over time in the face of dynamic environments. The specific problem we have addressed has been to approximately optimise system policies according to some performance criterion using evolutionary computing techniques, namely genetic programming.

We have developed a model of an energy system which encompasses several inter-connected community energy systems. The communities have energy demands, but are also capable of generating energy from a number of renewable resources, making them active prosumers, rather than passive consumers. They can trade energy with their neighbours and are also able to store energy. The central energy system is capable of generating energy from flexible conventional fuels if the energy produced by the communities is not enough to meet the total demand. We propose two methods, an *offline* and an *online* procedure, which enable this system model to optimise its performance through adaptation and evolution of its operating policy. The offline procedure tests alternative policies on the system model *a priori* and returns the one which maximises performance. The online procedure is intended to be applied to systems which are continuously running, evolving increasingly better policies over time. The policies evolved by our procedures *clearly outperform* a baseline policy we have designed by hand.

Both procedures return policies which are appropriate for a system, given some performance criterion, without a human designer's intervention. This could lay the foundations for the development of a new methodological paradigm for the engineering of collective adaptive systems. Our approach could be used to assist system designers in *systematically* finding good policies, which could lead to better performance and provide support for adaptation mechanisms in the face of non-deterministic changes in dynamic environments.

Keywords— agent, multi-agent, policy, sustainability, adaptive, dynamic, environment, social network, evolutionary computing, genetic programming, socially-inspired, biologically-inspired, community energy, simulation

Resumo

Instituições eletrónicas são sistemas multi-agente inspirados pelas ciências sociais, tipicamente operando de acordo com um conjunto de políticas, que são necessárias para determinar a forma como o sistema deve operar e para prevenir transgressões ou outros comportamentos indesejáveis. Um sistema deste tipo é frequentemente deparado com um ambiente dinâmico, que pode sofrer mudanças potencialmente imprevisíveis, e a sua política deve ser apropriada em relação a esse ambiente. No entanto, o espaço de possíveis políticas é normalmente muito grande, não existindo nenhum método sistemático para encontrar uma política apropriada dado um estado caracterizado pela população, a sua rede social e o ambiente.

A literatura apresenta várias abordagens recentes para lidar com mudanças potencialmente imprevisíveis no ambiente. Alguns trabalhos recorrem a conceitos da ciência social, política e económica e aplicam-nos a instituições eletrónicas, operacionalizando mecanismos que permitem que os agentes modifiquem coletivamente as políticas de um sistema. Outros trabalhos, inspirados pela Biologia, propõem técnicas de computação evolutiva que permitem que os agentes de um sistema se adaptem em resposta a alterações no ambiente. Estes dois tipos de abordagem motivam este trabalho, cujo objetivo de alto nível é explorar de que modo a adaptação através da evolução de políticas pode assistir o desenho de sistemas coletivos que se mantêm sustentáveis ao longo do tempo face a ambientes dinâmicos. O problema específico aqui tratado é a otimização aproximada das políticas de um sistema através de técnicas de computação evolutiva, nomeadamente programação genética.

Neste trabalho, é desenvolvido um modelo de um sistema energético consistindo em vários sistemas energéticos comunitários interligados. As comunidades têm requisitos de energia, mas também são capazes de a gerar a partir de várias fontes renováveis, o que faz com que sejam "prossumidores" ativos e não consumidores passivos. Podem comprar e vender energia aos seus vizinhos e também têm capacidade de armazenamento. O sistema energético central pode gerar energia a partir de combustíveis convencionais flexíveis se a energia produzida pelas comunidades não for suficiente para satisfazer a totalidade das necessidades. Este trabalho propõe dois métodos, um procedimento *offline* e outro *online*, para otimizar o desempenho do sistema através da adaptação e evolução da sua política. O procedimento *offline* testa várias alternativas para políticas no modelo do sistema *a priori* e retorna aquela que maximiza o desempenho. O procedimento *online* destina-se a sistemas que estão em execução contínua, evoluindo políticas cada vez melhores ao longo do tempo. As políticas que resultam destes procedimentos conduzem a um *desempenho claramente superior* quando comparadas com uma política de base construída manualmente.

Ambos os procedimentos retornam políticas que são apropriadas para um sistema, dado um certo critério de desempenho, sem a intervenção de um *designer* humano. Isto poderá estar na base do desenvolvimento de um novo paradigma metodológico para a construção de sistemas coletivos adaptativos. A abordagem aqui descrita pode ser usada para ajudar *designers* a encontrar políticas boas *de forma sistemática*, o que poderá conduzir a um desempenho superior e facilitar mecanismos para adaptação face a mudanças não determinísticas em ambientes dinâmicos.

Acknowledgements

I would like to thank both my supervisors, Rosaldo Rossetti and Jeremy Pitt, for their guidance and support throughout this project; my colleague and friend David Kurka for his helpful suggestions and for contributing to a positive work environment with his good sense of humour; Emma Hart for her insightful comments on both the ISoLA paper and this dissertation report; and my family for always being there for me.

Rui Cardoso

Contents

CONTENTS

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Abbreviations

EI	Electronic Institution
CPR	Common-pool resource
EC	Evolutionary Computing
GP	Genetic Programming
CES	Community Energy System
RL	Reinforcement Learning
SD	System Dynamics
IDE	Integrated Development Environment
MAS	Multi-Agent System
AEI	Autonomic Electronic Institution
PVP	Principled Violation of Policy
LPG	Linear Public Good
MABS	Multi-Agent-Based Simulation
DER	Distributed Energy Resource
CH	Cluster Head
IoT	Internet of Things
GA	Genetic Algorithm
MOEA	Multi-Objective Evolutionary Algorithm
ML	Machine Learning
MOGA	Multi-Objective Genetic Algorithm
MOGP	Multi-Objective Genetic Programming
WGP	Weighted Genetic Programming
LGP	Linear Genetic Programming
CEGP	Co-Evolutionary Genetic Programming
MS	Mass Spectrometry
LML	Lifelong Machine Learning
AIS	Artificial Immune System
JSSP	Job Shop Scheduling Problem
SDP	Software Defect Prediction
CIN	Cognitive Immune Network
CEP	Community Energy Planning
LP	Linear Programming
NLP	Non-Linear Programming

Chapter 1

Introduction

In this chapter, we introduce the work carried out during this dissertation project. We start by framing the project within its context and scope in Section 1.1. We then describe the goals of our project and the problem which we have addressed in Section 1.2. In Section 1.3, we summarise our results and achievements. In Section 1.4, we present an overview of the structure of this report.

1.1 Scope

Some agent systems are socially-inspired: they are governed by rules and policies (are "rule-based" or "norm-governed") and the agents form virtual societies, referred to as electronic institutions (EIs). These are typically open systems - with heterogeneous and autonomous agents -, with no central control or decision-making, and may be characterised by a dynamically changing environment. They should ideally have mechanisms for dealing with unpredictable changes, responding adequately when the performance is deteriorating, enabling sustainability and durability. In order to determine their operation and to prevent undesirable or non-compliant behaviour, a possible consequence of their openness to autonomous agents acting on behalf of third parties, these systems must have a policy in place. Examples of the EI paradigm include sensor networks, robotic swarms, and smart grids.

When this type of system is used to manage the access to a shared resource, the problem is referred to as common-pool resource (CPR) management. Ostrom [Ost15] presents several design principles for enduring institutions in the context of CPR management, including the notion that policies should be *mutable* in order to suit the environment. Some authors have proposed mechanisms to operationalise these principles and apply them to the design of EIs [PSA12], while also drawing other concepts from political and economic science to enable the agents to both *self-govern* and *self-organise* the adaptation of policies in the face of potentially unpredictable changes in the environment, such as distributive justice [Res82, PSA12] and knowledge management [POD]. Self-governing and self-organisation both imply the active participation of the actors within an EI in the decision-making process.

Other approaches for dealing with dynamic environments have been inspired by Biology. Methods have been proposed to adapt autonomic components using evolutionary computing (EC) techniques as a response to environmental changes, e.g. [CHC16, CHC14b]. These components exhibit cognition, namely learning and decision-making abilities, leading to collective *self-awareness*. Evolutionary approaches have also been used in other contexts. For example, genetic programming (GP) has been widely used to provide approximate solutions to optimisation problems, e.g. [BBBMM14], and to evolve and adapt rules of different sorts over multiple time scales in the face of a problem space whose structure changes dynamically, e.g. [SHP15, HS16].

Integrated community energy systems (CESs) may be viewed as EIs for the management of a CPR. They integrate distributed energy resources, such as photovoltaic cells and wind turbines, into local energy systems, meeting some or all of the local energy demands. The local energy systems are connected to a wider regional/national grid and local communities are not just passive consumers, but also active prosumers who generate and supply energy and may provide services to the larger system. This system has a dynamically changing environment: it faces fluctuations in the availability of resources, load, and demand over time, caused by seasonality, geographic location, and shifts in weather patterns, amongst other factors. The literature on CESs is mostly devoted to optimisation models for the planning and integration of these systems. In this project, we have modelled and simulated an energy system consisting of many inter-connected CESs and proposed methods for automatically constructing system policies. The following sections provide more detail.

1.2 Goals

Our top-level goal is to explore how adaptation through evolution of policies can assist the design of collective adaptive systems which remain sustainable over time when faced with a dynamically changing environment, since policy modification mechanisms are necessary in order to cope with potentially unpredictable environmental changes. In a norm-governed system, a single policy may not be appropriate for all situations. For example, an energy system could have the following modes of operation: decentralised (peer-to-peer) when demands are low, with all the energy being produced by the local communities; centralised when the system is overloaded, with communities trading exclusively with the regional/national grid; or a hybrid approach for normal levels of load and demand. Besides this, surprising events could occur which result in deterioration of the performance, rendering the current policy no longer fit. This is expected to be the case in energy systems with several distributed energy resources across multiple communities: weather can be unpredictable and unstable, affecting the production rate of intermittent renewable resource converters such as solar panels and wind turbines. Ideally, systems should be able to recover from performance losses after a reasonable number of time steps. In general, there could be a very large space of possible system policies. There is no systematic way of finding an appropriate policy given a joint state of the population of agents, their social network, and the environment [PH17]; it may not be tractable or possible to search the entire space of possible policies exhaustively.

Our research question is whether we can use GP to generate, adapt, and evolve policies under which systems operate in order to ensure that they remain sustainable over time. The specific problem we have addressed in this project is to automatically find operating policies which are approximately optimal for a given system according to some performance criterion - i.e., policies we would consider *appropriate*. This could assist designers in building systems for which it is hard to come up with a policy leading to good performance and which may be faced with a dynamic environment requiring constant modification and adaptation of policies. Even a human expert might lack not only the knowledge necessary to determine whether a given policy will result in good performance or to compare alternative policies, but also the creativity needed to design sufficiently good policies. The "ideal" policy for a given system may be counter-intuitive to a person, but an heuristic search over the space of possible policies, which is the base of what we propose in this work, is not sensitive to that.

1.3 Expected Contributions

In order to address the problem specified in Section 1.2, we have started by creating a model of an energy system in which several communities produce and consume energy and used it to run simulations to observe how different policies behave. We have used binary decision trees to represent policies. The key contributions expected of this work are two optimisation methods for automatically finding appropriate policies for this system model. The first one is an *offline* procedure which returns a policy that approximately optimises system performance using GP. The second method is an *online* procedure which evolves and adapts a population of policies over time by applying them to the system in turn and using performance history to increasingly improve the general quality of the policies in each new generation, drawing inspiration from reinforcement learning (RL) techniques. Results show that the policies resulting from these procedures clearly outperform a baseline policy which we have designed by hand.

The modelling approach we propose and the procedures we have implemented and tested for finding approximately optimal policies could provide the foundations for the development of a new methodological paradigm for the engineering of collective adaptive systems. The results are encouraging and provide insight into the effects of adaptation and innovation, through evolution of a set of policies, on the sustainability of a distributed system for CPR management, applied to the context of CESs. This work is also innovative in the sense that it brings together the paradigms of socially-inspired and biologically-inspired computing, as we have drawn notions from EIs when modelling a system in which energy is treated as a CPR and have used GP to evolve and adapt its policy.

1.4 Structure

The remaining of this report is structured as follows. In Chapter 2, we review the literature on the state of the art, focusing on EIs, EC and GP, and CESs. In Chapter 3, we provide a description

of the steps we have followed and the methods we have implemented. In Chapter 4, we discuss experimental results. In Chapter 5, we present the main conclusions which have emerged from this work and reflect on its limitations and on directions for future research.

Chapter 2

Literature Review

In a position paper, Pitt and Hart [PH17] propose the integration of the socially-inspired design patterns of EIs with the biologically-inspired techniques used in EC and GP to adapt and innovate the policy of a system as a response to dynamic and unpredictable changes in the environment. In this chapter, we review some key concepts which have enabled the implementation of this approach. In Section 2.1, we explore the notion of EIs. In Section 2.2, we review work on EC and GP. In Section 2.3, we present concepts and issues related to CESs. In Section 2.4, we elaborate on System Dynamics (SD) as a modelling and simulation paradigm. At last, we summarise the main conclusions of this chapter in Section 2.5.

2.1 Electronic Institutions

Agent-based systems which are governed by rules and policies, mirroring human institutions, are referred to as electronic institutions (EIs). Heterogeneous agents form societies and often seek individual goals, as well as common objectives. Sierra *et al.* [SRAN+04] define EIs as a "computational analogue of human organisations". They mention an increasing need for "organisational abstractions" that make it easier to design systems in which humans and agents interact to achieve individual and collective goals. They also note that, in open MASs, agents may exhibit very diverse and even fraudulent behaviour and that cooperation is not predetermined, but rather *emerges in real time*. Engineering these systems may therefore appear to be a daunting task; structures enforcing rules and regulations must be in place in order to avoid chaos. They present an integrated development environment (IDE) which supports the engineering of MASs as EIs. De Jonge *et al.* [dJRAGS15] survey the infrastructures and tools which have been developed over the years to support the engineering of open systems by means of EIs.

Self-organisation means that a certain system is able "to change its organisation without explicit command during its execution time" [DGK05]. Ye *et al.* [YZV17] point out that the operation within a self-organising system is "localised and decentralised" and that the cooperative behaviour of individuals happens "without any external control or influence". They also note that this concept has been applied to many fields, among which multi-agent systems (MASs). Arcos

et al. [ARAR08] define EIs with self-governing and self-organising capabilities as *Autonomic Electronic Institutions* (AEIs). They stress the need for "adapting regulations to comply with institutional goals" and present a framework to support the design of this type of system.

Much of the literature about EIs is concerned with translating, representing, and enforcing norms, rules, and protocols within a system. Dignum [Dig06] discusses how norms and regulations can be incorporated in an EI and how to ensure that it is operating according to those norms. He points out that the norms are usually specified on an abstract level and therefore need to be translated to both a level on which "their impact on the institution can be described directly" and an implementation level. He proposes that some formalism be developed for performing this translation; this formalism is also necessary for verifying whether an EI complies to the original abstract norms. Ibrahim *et al.* [IKS03] also point out that there are several levels of abstraction concerning norms in EIs and propose a way to categorise these abstract norms into how they regulate the behaviour of the participants and to translate them into concrete "actions and concepts that can be handled within the institution". Aldewereld *et al.* [ADGC+07] stress the importance of regulatory norms in open systems, given the heterogeneity of the agents, which may intentionally or unintentionally behave in ways that are not acceptable. They address a "big gap between the theoretical work on norms and the practice" of EIs, pointing out that abstract norms are usually "very vague and ambiguous" and that implementations limit the autonomy of agents by imposing constraints on their behaviour. They define norms through means of an "operational semantics" and propose a way to translate them to the implementation level. The norms are no longer enforced by behavioural constraints, but rather by mechanisms which can detect violations and react to them; this gives the agents "more freedom and flexibility", while still ensuring their compliance to the norms.

Grossi *et al.* [GAVSD06] also address the translation of top-level, abstract norms into concrete constraints. They note that abstracting norms from their implementation is necessary to ensure that they remain stable "over time without need for modification". They point out that "concrete situations are generally described in terms of ontologies which differ from the abstract ontology in which, instead, norms are specified" and argue that a connection between these two ontological levels is required in order to make "the relation between the concrete and abstract specifications explicit". García-Camino *et al.* [GCRASV09] propose a rule language to specify the agents' *normative positions* - i.e., their obligations, permissions, and prohibitions - and constraints. This specification is then implemented with techniques for constraint solving. The use of this language is illustrated with EIs. Cardoso and Oliveira [CO04] describe a *normative framework* for contract validation and enforcement within an EI. This normative system "establishes a level of trust enabling the interaction of heterogeneous" agents and enforces electronic contracts in a business-to-business scenario. The same authors propose a contract model which takes advantage of an existing normative framework [CO08]. Contracts established among a group of agents are defined as a set of applicable norms. The model they present uses the normative background of an EI to specify and enforce contracts. The contracts can be underspecified, in which case they rely on default norms by means of inheritance; this is inspired by the hierarchical organisation of norms

in the real world.

Esteva *et al.* [EVSRA04, ERASV04] propose a computational approach to verify whether the norms of an EI are *consistent*. Their goal is to verify not only that an EI complies with its norms, but also whether the norms prevent what should be norm-compliant executions from occurring. To do this model checking, they analyse the messages exchanged among the agents. Khalil-Ibrahim *et al.* propose an algorithm for checking whether an EI complies with its norms. They note that the values governing an EI have to be translated into actions and concepts which can be implemented and checked and introduce "substitution rules" which map these values to concrete implementation-level norms. Kurka and Pitt [KP17b] pose the question of whether full norm compliance is always desirable and investigate the possibility of intentionally not acting upon transgressions, known as Principled Violation of Policy (PVP). They consider scenarios in which monitoring for violations has costs, there may be accidental non-compliant behaviour, and norms can be subjective or unfair. They conclude that, by introducing flexibility in these scenarios, PVP is more appropriate than strict enforcement of norms, as the EIs become more tolerant and resilient to accidental behaviour and more cost effective.

Ostrom [Ost15] proposes a view of self-organising institutions for the management of CPRs in which the rules of an institution govern the appropriation and provision of shared resources. She identifies eight design principles for the management of CPRs in enduring institutions after arguing that, unlike predicted by game theory, CPR management does not necessarily result in a "tragedy of the commons", in which a group of self-interested and rational agents eventually depletes a shared resource. One of those principles states that the rules *should be mutable by other rules and adaptable to suit the environment*. Besides encouraging compliance and tolerating unintentional errors, these rules should be the result of collective decisions and ensure a sustainable distribution of resources, necessary for the institution to be enduring and long-lasting.

Pitt *et al.* [PSA12] axiomatise these principles, expressing them in logical form. This formal specification is used to implement a test bed to show that they result in enduring EIs for the management of CPRs. They note that, in open systems, there is no centralised control and operation must take into account the possibility of errors, non-compliance, and violations. The endurance of an institution is considered more important than the optimal distribution of resources and a strategy resulting in a sub-optimal distribution in the short term might prove better in the long term if the resource is not depleted. After making a distinction between exogenous systems, in which resources come from the environment, endogenous systems, in which the agents must supply the resources, and hybrid systems, a combination of both, they analyse the problem of allocating endogenous resources with an implementation of the Linear Public Good (LPG) game [Gäc07]. They resort to a framework which enables the specification of a protocol stack which agents can use to alter the policies of a system at runtime [Art12]. The specification space is formally defined by a number of degrees of freedom, such as the allocation method (ration, queue, etc.). They also note the difference between brute facts, concerning the physical state - such as the amount of resources available -, and institutional facts, concerning the conventional state - such as the resources each agent demands and provides. The rules are formalised using Event Calculus [KS86], which is an

action- and event-oriented language. The experimental results show that the principles defined by Ostrom entail enduring management of CPRs in self-organising EIs.

The principles defined by Ostrom do not explicitly concern a notion of fairness and justice. Pitt *et al.* [PBM14] build on previous work by analysing the mechanisms influencing the fairness of the result of a resource allocation. Agents self-organise the allocation process by participating in a voting procedure. The authors note that an outcome which is unfair at a given time step could be part of a sequence of fair cumulative outcomes. These temporal aspects are especially important for economies of scarcity, in which an agent's need may not be satisfied at a particular time step but could be after a sequence of allocations. Rescher [Res82] presents the concept of distributive justice, identifying several ways of distributing resources based on *legitimate claims*. Each legitimate claim taken in isolation is inadequate; therefore, he argues that several criteria should be taken into account when distributing resources. Pitt *et al.* use each of those canons in a Borda count procedure, ranking the agents. These, on the other hand, vote on the weight which corresponds to each claim, ranking them in order of preference; this leads to self-organisation of the weights. The LPG game is once again used as an example application of resource allocation. A logical specification in Event Calculus is used as an executable specification in a Prolog simulation. Experimental results are then presented using a testbed implemented with Presage2 [MPSB12], a multi-agent-based simulation (MABS) platform. These results reveal robustness to purposeful violations, with the weights associated to each canon adjusting to hinder non-compliant agents. The allocation strategy is fairer for compliant agents; in fact, full compliance is revealed to be the optimal strategy. It is also observed that a sequence of allocations which individually seem unfair results in a cumulatively fair allocation. Among the metrics used for assessing the results were the number of remaining agents in a cluster of the LPG game, the utility for the agents, and the fairness of the allocation method.

This work is the basis for a proposal by Torrent-Fontbona *et al.* [TFLBP16] of a method for energy demand allocation in a power system with distributed energy resources (DERs). The method is based on self-organisation to set the rules governing the allocation, thus eliminating the need for a central authority. The rules are based on different canons of justice providing legitimate claims, which are, as before, implemented as voting functions, enabling the agents to agree on an allocation method. Pitt *et al.* [PBR15] elaborate on the concept of fairness in open systems for CPR management. They argue that self-organisation is necessary in these systems because the decision-making required is too complex for human intervention. Self-organisation implies that the agents agree on the rules used for allocating resources and on the mechanisms for changing those rules. They argue that determining whether an allocation is fair requires a formal characterisation of *computational justice*, which must then be operationalised as system policies. They present a formal model and some experimental results, concluding that there is a need for further research into computational justice. The same authors propose a framework based on computational justice which enables the agents of a system to determine whether a rule set is fit for its purpose [PBR13]. They argue that this is necessary for the agents to be able to learn and adapt rule sets which are appropriate for particular states of the environment. They define a new

metric for evaluating "fitness for purpose" based on the "distribution of institutionalised power" and demonstrate that it can expose fairness or unfairness. They conclude that assessing "fitness for purpose" is crucial for adaptive EIs.

Pitt *et al.* [POD] note that, in self-governing socio-technical systems - involving interaction between people and information technologies - based on conventional rule sets, unrestricted self-modification of rules and policies may expose the system to a number of risks, including a subset of the participants exploiting it for their own self-interest. They propose a three-layer architecture by integrating an operational layer and a governance layer with an "open and transparent" knowledge management layer, based on the knowledge management processes of democracy in classical Athens. They make the case that, for addressing collective action problems and self-organisation, self-governing socio-technical systems are needed which "materially represent shared values, distribute power fairly, and manage knowledge transparently". They conclude that this study can "provide the foundations for sustainable democratic self-governance in socio-technical systems".

Pitt [Pit17] also addresses the dilemma of mutable rules, which are vulnerable to adaptation to serve the interests of a few rather than the collective. Self-modification of rules should ensure: *congruence*, i.e., that the rules are appropriate given the current state of the environment (as required by Ostrom's principles); *acceptability*, as determined by the collective assessment of the participants; and a minimal risk of the institution being exploited for the self-interest of a few. Pitt defines a formal characterisation of interactional justice as "a mechanism for self-organisation of institutional rules in an open system". This characterisation focuses on two main aspects: the "interaction between each member and the institution", which is the "personal treatment", and the interaction between members who compare their personal treatments amongst themselves, the "interpersonal treatment". Interactional justice uses *opinion formation* to derive a collective assessment from individual assessments. A negative collective assessment can lead to "self-organised reformation of the rules". Interactional justice can thereby address congruence, as the rules are adapted to fit the environment, and acceptability, as participants take part in collective assessment. This also minimises the risk of the rules being adapted to serve the interests of only a few of the participants. Pitt concludes that interactional justice contributes to good self-governance by managing knowledge, increasing stability, and reducing conflict.

Garbiso *et al.* [GDC+17] improve an existing self-adaptive clustering algorithm for vehicles connected by a hybrid network. Vehicles communicate with each other via a licence-free radio technology, but must pay usage costs for accessing the cellular network. Their algorithm improves the cost efficiency of using the cellular network by electing a Cluster Head (CH), which is a "gateway to the Internet", offloading the data on to the vehicle-to-vehicle links, and aggregating it. The problem with the original procedure is that the individuals elected as CHs bear all the cellular network access costs in this process. If the members of a cluster perceive that the cost distribution is not fair, they may leave the system, which in turn could cause its collapse. Garbiso *et al.* focus on the fairness and social acceptability of the procedure by developing a *fairness-aware* election algorithm based on the theory of distributive justice, which has been mentioned before. This new algorithm enables vehicles to influence CH elections via voting on a set of *canons* of justice

(legitimate claims). Each vehicle votes on the canon it finds most convenient. By comparing it to the original fairness-agnostic algorithm, they observe that this approach significantly improves fairness criteria while having no adverse effects upon network performance.

Some of the ideas presented in this section are applied to the study of smart grid systems by Pitt *et al.* [PDB17]. They begin by pointing out that smart grids are a "crossover point between cyber-physical systems and socio-technical systems". Cyber-physical systems integrate computing devices and other physical artefacts, the paradigm of the Internet of Things (IoT). Socio-technical systems have an impact upon communities and their behaviours. Smart meters in residences, serving as a point of contact with consumers, are what makes smart grids socio-technical systems; they enable demand-side management and demand-side self-organisation, which are an important aspect of community energy systems (CESs), discussed in Section 2.3. Pitt *et al.* argue that, if there is "an asymmetry in the distribution of power and information", then collective action will not succeed. Their main conclusion is that integration of the active participation of people in socio-technical systems requires open platforms which distribute power and manage knowledge fairly and transparently.

Chen *et al.* [CAA+] use concepts of self-organisation within socio-technical systems for voltage regulation in a power distribution network with distributed generation units. The distribution network is decomposed into smaller subnetworks, breaking down the original problem into several subproblems. The distributed generation agents then coordinate with the other agents in their subnetworks to self-organise voltage regulation and detect violations. This was an improvement of the work by Chen *et al.* [CMKP15], in which the authors demonstrate the suitability of multi-agent systems (MASs) to address the problems of voltage control and power flow management in distribution networks with increased integration of renewable generation and demand-side management.

The relationship between humans and technology is also a relevant topic in the literature. Bogdanovych et al. [BBSS05] analyse the relationship between humans and autonomous agents in EIs. They note that people are reluctant to delegate full control of the decision-making process to agents in e-commerce scenarios and propose 3D virtual worlds as a solution to this issue, providing an "immersive user interface" where humans can observe and intervene in the behaviour of their agents. They argue that this human-centred perspective can increase people's trust in agents in e-commerce applications. Gärtner *et al.* [GSFB10] also present an immersive 3D e-tourism environment. The virtual world enables users to interact with other humans and software agents in an "intuitive and easy way". In a position paper, Pitt and Diaconescu [PD16] elaborate on how people and technology can interact successfully in smart cities to solve collective action problems. They advocate for "interactive self-governance" as an extension to algorithmic self-governance and reflect on some issues that may arise from the symbiosis between humans and technology. They present guidelines for the design and implementation of interactive self-governance and stress the need for socio-technical systems to incorporate *human values* in a concrete form.

Other relevant work has been published related to self-organising EIs. Petruzzi *et al.* [PPB17] address the issue of electronic social capital for MASs. Social capital refers to intangible collective

resources held by a group of individuals [Col88]. The following dimensions of social capital are analysed: trustworthiness, networks, and institutions, as identified by Ostrom and Ahn [OA03]. Petruzzi *et al.* propose a framework with the goal of optimising self-organised collective action, taking electronic social capital into account. They note that social capital enables individuals to solve problems requiring collective action more effectively. They tested the framework in different scenarios. Experimental results enable them to conclude that social capital maximises satisfaction and utility in the long term and is scalable for large populations. They argue that electronic social capital is of the utmost importance for self-organised community systems. The paradigm of self-organising EIs has also been explored by Sanderson *et al.* [SRK+15] as a mechanism to achieve flexible and adaptive assembly and manufacturing lines. Kurka and Pitt [KP17a] present an innovative approach to CPR management applied to a scenario of industrial cooperation. They combine principles of self-organisation with blockchain and smart contracts to organise communication and transactions, with the goal of solving the problem of distributed supply and demand more efficiently. They demonstrate that the system makes justifiable decisions regarding resource allocation and is robust to abuse and non-compliant behaviour by its participants.

2.2 Evolutionary Computing and Genetic Programming

Evolutionary Computing (EC), in its broader sense, draws inspiration from biological evolution to solve optimisation problems, involving population-based stochastic search approaches [BBBMM14]. Genetic Algorithms (GAs) are based on Darwin's theory of evolution and the mechanisms it describes, namely natural selection, evolving solutions to problems according to the principle of "survival of the fittest". Genetic Programming (GP) additionally employs tree-like representations of candidate solutions. These approaches have been used extensively to find approximate solutions to many optimisation problems. For example, Aguilar-Rivera *et al.* [ARVRRO15] present a survey of EC methods applied to financial systems, such as financial markets. Among other techniques, they review the application of GAs, GP, and multi-objective evolutionary algorithms (MOEAs) to this sort of problem. They observe that, while the interest in particular methods has changed over time, GAs have remained the most popular approach in the domain of financial systems. Hu *et al.* [HLZ+15] review EC techniques for *rule discovery* in stock algorithmic trading, which is an approach for automatic analysis and trading. They note that companies have been providing an increasingly larger amount of performance data and that extracting useful knowledge from it can result in advantages against other investors. Machine learning (ML) techniques have proved effective for this analysis; however, they usually do not provide clear explanations, which hinders investors' trust in these systems. Rule discovery can tackle this issue, as rules describe relationships between variables explicitly. EC has been used for rule discovery, as it can "find a sufficiently good solution for a wide range of problems within a relatively short time". They observe that GP and GAs are the most widely used techniques for rule discovery in stock algorithmic trading. Florentino *et al.* [FCSB14] use EC to study ways of controlling populations of mosquitoes which transmit dengue fever. They point out that there are several models which

assist the search for an optimal control of mosquito populations, e.g. dynamic models which take into account the effect of investing in insecticides, sanitation, and educational campaigns. They discuss an optimisation model which is based on introducing sterile male mosquitoes in the environment for population control and present a multi-objective genetic algorithm (MOGA) to solve it. Miller and Mohid [MM13] apply GP to function optimisation, which they define as finding a "vector of real numbers that optimises a complex multi-modal fitness function". Their approach is successful in optimising benchmark functions. Schuster [Sch03] describes a multi-objective genetic programming (MOGP) approach to derive analytical approximations to optimisation models applied to the investment in stock markets. The results reveal that the approximations are more accurate than other approaches in the literature. Tsai and Lin [TL10] point out that the trees evolved by GP are not capable of encoding weight coefficients easily and propose a new paradigm called Weighted Genetic Programming (WGP); the links in the trees evolved by their WGP algorithm have weights, which enables the generation of weighted formulas. They apply their approach to predict high-strength concrete parameters and observe that it provides accurate results. Guven *et al.* [GAZ09] use Linear Genetic Programming (LGP) to predict circular pile scour caused by waves and currents in loose sedimentary beds. Scour affects the stability of structures such as bridges and, therefore, accurate estimations of this phenomenon are required. LGP is an extension to GP in which sequences of instructions from an imperative programming language are evolved; the individuals are represented by graphs describing data flow, rather than by trees. Their results show marked improvement over conventional regression analysis for predicting circular pile scour.

Hart *et al.* [HSGK17] present a new method to predict wind damage to trees using *feature construction*, with the goal of assisting forest management. They address the problem caused by the fact that the datasets which are available are small and that very few features are collected from each tree. Additionally, the measured features may fail to discriminate between classes effectively. A GP algorithm is used to augment the original data set with new features. The algorithm is run several times, resulting in a large number of new features. Each run produces a single feature, which is the result of applying a program represented by a tree of operations, built from the original features (terminal nodes) and a range of predefined function nodes. After generating this set of features, a *feature selection* method is applied in order to obtain a minimal set for prediction. The observations show that the evolved features are better than all of the original features in terms of accurately discriminating between classes, which results in a significant classification improvement. This is a clever application of GP to improve the performance of classification models. It is computationally easy to generate a big set of new features using the method they propose and an efficient feature selection method can afterwards be used to avoid redundancy. This work can be considered as an improvement of the proposal by Krawiec [Kra02], who also uses GP for feature construction to improve classification accuracy. He notes that symbolic ML classifiers, such as decision tree classifiers, are usually unable to construct appropriate representations of external inputs. On the other hand, non-symbolic or sub-symbolic classifiers, such as neural networks, while capable of constructing those representations, do not produce knowledge intelligible to a human. By expressing features as LISP-like expressions, feature construction using GP is able to create

representations of inputs which are both intelligible and appropriate for discriminating between classes correctly. Krawiec's approach requires the number of evolved features to be set as an input parameter of the GP procedure, but also results in significant improvements in the accuracy of ML classifiers.

Other authors have used GP for the same purpose of transforming the original input features to provide more appropriate representations for ML classifiers. Smith and Bull [SB05] use GP for feature construction followed by feature selection to improve the accuracy of the C4.5 decision-tree classifier. They test their approach on known datasets and conclude that it "provides marked improvement in a number of cases". Lin and Bahnu [LB05] use co-evolutionary genetic programming (CEGP) to improve the performance of object recognition. CEGP algorithms are an extension of GP in which several subpopulations are maintained; each subpopulation evolves a partial solution and a complete solution is obtained by combining partial solutions. In their approach, individuals in subpopulations are "composite operators", represented by binary trees whose terminal nodes are the original features and whose function nodes are *domain-independent* operators. A "composite operator vector" is thereby evolved cooperatively and applied to the original features to produce "composite feature vectors", which are then used for classification. They point out that the original features can either be very simple or incorporate an expert's domain knowledge. Their results show that CEGP can evolve composite features which lead to better classification performance. Neshatian and Zhang [NZA12] also use GP for feature construction to improve the accuracy of symbolic classifiers. They use an entropy-based fitness measure to evaluate the constructed features according to how well they discriminate between classes. Their results reveal "consistent improvement in learning performance" on benchmark problems. Ahmed *et al.* [AZPX14] also use a GP approach to feature construction for biomarker identification in mass spectrometry (MS) datasets. Biomarker identification means detecting the features which discriminate between classes. They point out that it is a difficult task for most MS datasets because the number of features is much larger than the number of samples and that feature construction can solve this problem by reducing the dimensionality of the inputs. Their method produces nonlinear high-level features from low-level features and is shown to improve the classification performance when tested on a number of MS datasets.

Sim *et al.* [SHP15] describe an innovative hyper-heuristic system. Hyper-heuristic algorithms search a space of heuristics which provide solutions to problems, in contrast with meta-heuristics, which search a space of problem solutions. They propose a lifelong machine learning (LML) system called NELLI, which learns continuously over time using prior knowledge and incorporating some form of long-term memory, applying it to a combinatorial optimisation problem. They argue that the natural immune system is an example of a lifelong learning system and, therefore, their system solves optimisation problems using hyper-heuristic methods inspired by immunology. NELLI consists of generators for covering the problem space and providing a continuous source of new heuristics, as well as an artificial immune system (AIS). The AIS encompasses heuristics and problems interacting in a network, with problems viewed as pathogens and heuristics as antibodies. The key idea is that the problems "provide a minimal representative map of the problem

space" (so that not all problems are included in the network) and each heuristic solves a niche of problems. The network adapts efficiently to changes in the structure of the problems. A heuristic only remains in the network if it solves at least one problem better than any other heuristic. On the other hand, only problems for which a single heuristic gives the best solution are kept, since this is a clue that they are hard to solve. This results in a competitive exclusion effect between heuristics and problems and the problem space is thereby covered efficiently. The system was applied to the 1D bin-packing problem, using a generator of problem instances. The results show that it is efficient and scalable, outperforming human-designed heuristics, and adapting efficiently to unseen problems. For that reason, they argue that NELLI has significant advantages over other hyper-heuristic approaches for dealing with dynamic data. Burke *et al.* [BHKW12] apply a hyper-heuristic approach to the 2D strip packing problem, in which a certain number of rectangles must be placed on a sheet while minimising the length of sheet used. They point out the significance of a hyper-heuristic GP procedure for automatically generating heuristics from a set of building blocks, as it can assist human experts in the process of designing new heuristics for providing solutions to a number of problems. Their results show that the approach can automatically generate reusable heuristics whose quality competes with that of the best human-designed heuristics in the literature.

Hart and Sim [HS16] describe NELLI-GP, the successor of NELLI. They address the Job Shop Scheduling Problem (JSSP), in which several operations are scheduled for execution in multiple machines. Heuristics are sequences of rules and they propose an ensemble of heuristics which are evolved using GP, with existing dispatching rules described in the literature as building blocks. A solution to a problem instance is obtained by applying each rule in the heuristic sequentially; applying a rule results in a single operation being scheduled. The rules themselves are formulated as trees of operations, returning a real value which determines the priority of an operation. GP is used to evolve new heuristics to be included in the ensemble, as well as new rules to be part of the sequence of dispatching rules which make up a heuristic. They argue that an ensemble of heuristics resembles bagging methods. An interesting detail to note is that, since the goal is to have a diverse population of heuristics, the system does not apply the crossover operation when evolving new heuristics using GP, since it "inevitably leads to convergence". The results show that using an ensemble is preferable over a single heuristic and that the system generalises well from the training set. The ensembles are reusable: after being fitted to a data set, they can be used with a different one (adaptation). As in their previous work, they used a problem generator for covering as much of the problem space as possible. They conclude that new rules can be evolved which result in a better performance than other scheduling rules and hyper-heuristic approaches for the JSSP and also that the performance of the evolved heuristics is improved with a large and diverse training set. Mauša *et al.* [MG17] also use an ensemble approach to address a classification problem for software defect prediction (SDP). The goal of SDP is to optimise testing by "identifying defect-prone software components in advance" and adjusting strategies. Ensembles divide the original problem space into several subproblems, which is relevant because, in the case of SDP, the datasets are often very unbalanced. They investigate the application of

MOGP to evolve ensemble solutions. Folino *et al.* [FPS03] address the classification problem for very large data sets which do not fit in the main memory. They evolve an ensemble of decision tree predictors using GP; each predictor is trained on a subset of the original data. New data points are classified using a majority voting algorithm. Their results show that the ensemble of classifiers leads to higher accuracy than a single classifier which has been trained on the entire data set. An approach based on hyper-heuristics has also been proposed by Segredo *et al.* [SPHGV] for *parameter control* applied to meta-heuristics such as MOEAs. Parameter control refers to runtime modification of the parameters of a meta-heuristic approach, as opposed to *parameter tuning*, i.e., searching for an optimal parameter setting which remains fixed during a run. They apply their approach to frequency assignment problems and observe that it outperforms parameter tuning.

Capodieci *et al.* [CHC16] address the issue of adaptation in collective systems of autonomous components, such as MASs, facing unpredictable and dynamically changing environments. In particular, they investigate how to develop distributed systems capable of exhibiting *self-expression*, which is the ability to modify architecture and coordination patterns in runtime as a response to changes in the environment. Notice the similarity between this concept and self-organisation, discussed in Section 2.1. Drawing inspiration from the natural immune system, which is "able to make collective decisions based on the simultaneous sensing of environmental changes", they resort to the AIS paradigm to propose a framework for designing and implementing collective adaptive systems which are capable of self-expression. They apply the framework to three different case studies and conclude that the systems are able to select "an optimal coordination pattern or organisational architecture during runtime" in environments which change dynamically. The same authors explore the application of the Cognitive Immune Network (CIN) paradigm to the design of autonomic systems [CHC14b]. These systems are made up of ensembles of heterogeneous components and are expected to self-adapt with little or no human intervention in the face of non-deterministic changes in the environment. The CIN paradigm draws inspiration from the natural immune system to address both adaptation and *self-awareness*, which refers to the ability to exhibit *cognition* and *learning* mechanisms. They propose the application of the CIN paradigm for designing collective adaptive systems and discuss experiments conducted under a robotic swarm scenario. They observe the emergence of self-awareness in the system as the autonomic components are able to make decisions on their own and learn from past experience. In other related work [CHC14a], they apply similar concepts to the study of evolution within *coalescing* chemical swarms, in which particles collectively adopt robust geometric shapes according to sets of parameters called *recipes*. They propose an algorithm inspired by the AIS paradigm for "controlling the evolution of single particles" in the swarm which results in the *emergence* of new shapes and behaviours. They also present a new approach to measuring the fitness of recipes and the distance between recipes during the evolutionary process.

2.3 Community Energy Systems

Integrated community energy systems (CESs) are "a modern development to reorganise local energy systems to integrate distributed energy resources and engage local communities" [KKF+16]. They ensure self-supply of energy and are also capable of supplying the larger energy system. Local communities are no longer considered passive consumers, but rather active prosumers, who also produce energy. Following the motto "think globally, act locally", CESs can help tackle global energy and climate issues by increasing renewable energy production and reducing demand for fossil fuels. Much of the literature on this subject concerns the planning and optimisation of integrated CESs. Huang *et al.* [HYPZ15] review methodologies and software which address community energy planning (CEP). CEP happens on the demand side (at the communities) and its operating object is secondary energy, which does not require further transformation (e.g. hot water and electricity). The goal is to optimise energy production, distribution, and usage, subject to constraints of investment, resources, and demand. Linear (LP) and non-linear programming (NLP) are common techniques to obtain solutions to this optimisation problem, although many approximation algorithms, such as GAs, have also been developed [BD18]. They describe both tools for the design of CESs and research on methods for predicting community demand, pointing out that *demand is difficult to forecast* due to uncertainty.

Koirala *et al.* [KKF+16] argue that CESs can "provide valuable flexibility in the market", since local communities also generate and supply energy, and are important for self-sufficiency and sustainability. They note that integration of CESs is a challenge, as energy systems have an increasingly higher share of intermittent renewable resources, *whose generation is difficult to forecast*. Flexible generation can be achieved with conventional fuels. Increasing *self-consumption* and *meeting local demand with local supply* is an important goal for integrated CESs. They also review some progress in smart grids which are the basis for integrated CESs. Strickland *et al.* [SVS+16] review work on community electrical energy projects. They note that schemes have tended to focus on thermal energy to improve efficiency in communities and that "the number and variety of deployments have increased", as these projects are widely acknowledged as "a method for decarbonising energy systems". They state some of the benefits of these systems for communities, such as "more independence from the grid and reduced emissions", as well as opportunities to reduce energy bills. It is important to consider communities as units so that they have "greater negotiation power over energy contracts". They focus on control schemes, which concern generation, storage, and load, and point out that appliances can be controlled, shifted, and throttled by a local house controller which communicates with a central controller. Most control schemes use an MAS approach where each property is treated as an autonomous agent.

Mendes *et al.* [MIF11] survey tools for optimising, planning, and analysing integrated CESs. They note that this sort of system can have many "technical, economical, environmental, and social benefits" and that, besides delivering sustainable energy within communities, it can also provide valuable assistance to the distribution network. They argue that demand-side management, "entailing actions influencing the quantity or patterns of energy use by consumers", is vital. They

point out that the global financial crisis has led to policies promoting the reduction of fossil-fuel dependency and the increase in the share of renewable energy. They conclude that there is only a small number of tools for modelling at the local level.

Xu *et al.* [XJJ+15] propose a hierarchical framework for the management of integrated CESs. They note that coordinating several energy conversion processes is important for increasing the share of energy generated from renewable resources. Their framework encompasses daily scheduling with hourly adjustments. An optimal operating plan for the day is generated using load and renewable generation forecasts. The hourly adjustments consider possible errors in these predictions, taking into account the unpredictability of intermittent renewable energy generation. They conclude that this method for scheduling in integrated CESs can reduce operating costs and smooth power fluctuations in electric lines.

Bourazeri *et al.* [BP14] describe a serious game for decentralised CESs in which several houses share a CPR, generating and storing energy locally. They address the issue of collective action, related to self-organisation, arguing that collective awareness is necessary for it to be "fair, sustainable, and successful". In the game they developed, residences are installed with both renewable energy resources and storage equipments and participants need to work together to ensure that the community is sustainable over time. They used Presage2 [MPSB12] for the simulation. Their observations indicate that the game developed leads to collective awareness among the participants and to a more responsible and sustainable energy use.

Other work includes that of Huang *et al.* [HYPF17], in which they describe a "future energy system perspective based on the principle of industry 4.0", involving information sharing, consumer participation, cross-business operation, and decentralised independent decision-making. They argue that MASs are "especially suitable for simulating and modelling a distributed decision-making system". Koirala *et al.* [PHCG] assess integrated CESs, estimating costs and benefits, which are "highly subjected to the system of prices and charges", and comparing them to the current energy system. Huang and Yu [HY14] and Huang *et al.* [HYPL15][HYCP17] describe further optimisation approaches for CEP. The latter two use goal programming, which takes data uncertainty into account. Karunathilake *et al.* [KPR+16] propose a framework for the integration of renewable energy in community development projects. Lin *et al.* [LZH+17] discuss a model for supporting the adaptation of CESs in the face of uncertainty.

2.4 System Dynamics as a Modelling and Simulation Paradigm

System Dynamics (SD), created by Jay Forrester [For71], is an approach to modelling and simulation of complex systems, involving feedback loops, control structures, and delay structures. Originally intended to help improve the understanding of industrial processes [For61], it has been applied to many other contexts, such as the work by Rossetti *et al.* [RLCB02, RBB+02] regarding complex transportation systems. In fact, Vázquez and Liz [VL14] argue that it is "a paradigmatic case in the field of modelling and simulation", especially when dealing with complex social systems, such as EIs. They point out several features of SD modelling which are common throughout

the field of modelling and simulation. The most relevant feature is that SD modelling can handle systems for which there is no underlying theory available. These cases, in which a set of mental models is "the only reliable source of structural knowledge" [VL14], are very common and SD modelling is a useful tool to improve our understanding of these systems and our decision-making processes. Vázquez and Liz propose an interpretation of the SD modelling and simulation paradigm under the philosophical theories of *constructivism* and *expressivism*.

Searle [Sea95] proposes a *constructivist* perspective of social reality. He argues that social reality, which encompasses social phenomena and language, is a result of our "collective intentionality", which is a biological feature. He points out two mechanisms used to construct social reality: the assignment of functions and constitutive rules. The functions refer to those which we impose on reality, such as when we assign a role to an object which it would not be able to perform *per se* (e.g. a bank note). Systems of constitutive rules are what make certain practices possible and institutional facts accepted. Objects such as bank notes must be accepted to perform their function in a given context (in this case, count as money). These constitutive rules are a result of collective intentionality. For this reason, social and institutional facts are *ontologically subjective*, as they only exist as a result of our subjectivity, depending on us. However, they are *epistemologically objective*, as the claims we can make about them are not dependent on personal opinions or preferences; this is a relevant distinction made by Searle. Another important point of Searle's proposal is that, while the assignment of functions and constitutive rules enable us to construct social and institutional reality, this construction, as well as the intentionality behind it, is usually only *implicit*, as we make use of a set of skills which he calls *the background*.

Brandom [Bra96, Bra00, Bra02] proposes an alternative *expressivist* perspective. The most important aspects of his views concern meaning, an expressivist theory of logic, and the rational process of making explicit what is implicit. According to Brandom, all meaning and semantics depend on the representational features of our language. Those representational features are the result of a practical inferential ability. Under this perspective, Logic merely expresses the inferential relations we implicitly make, making them explicit, instead of referring to an "ideal realm of logical truths" [VL14]. Expression, according to Brandom, is *making conceptually explicit* something which is implicit in what we do. He argues that this enables us to carry out the rational process of expressing, understanding, and improving our practices.

Vázquez and Liz [VL14] find some common ground between Searle's constructivism and Brandom's expressivism and propose an analysis of the SD paradigm under these two perspectives. When building an SD model from a set of mental models, usually our only source of knowledge, they argue that we are making structures which are implicit in the social and institutional reality explicit. The "mathematical and computational tools" used to construct these structures then have the same expressive role as Logic does according to Brandom's expressivist perspective. A common concern when modelling social systems is *realism*, i.e., whether the model could represent something objectively real. They claim that the structures of SD models make up the same sort of reality as the "social and institutional worlds"; indeed, both the structures of SD models and social reality are ontologically subjective, as they depend on us in order to exist - they are the result of

collective intentionality. Therefore, they argue that, "in many cases", it is appropriate to assume that those structures "represent or describe something objectively real" in the systems being modelled. Despite being ontologically subjective, the SD models are epistemologically objective, just like social reality, as what we can say about them does not depend on our subjective opinions. By making explicit the structures of social systems which are otherwise implicit, we can then draw explanations about these systems, as their dynamic consequences also become explicit. The "self-consciousness" which results from this enables us to rationally improve the decision-making processes guiding the systems which have been modelled.

2.5 Summary

Table 2.1 summarises the work reviewed in this chapter. The most important aspects to note concern the approaches which have been proposed so far to deal with unpredictable changes in dynamic environments. Among the references on EIs, there is a focus on the application of concepts from social, political, and economic science, such as self-organisation, self-governance, distributive justice, or knowledge management, to digital organisations as a mechanism for enabling their actors to collectively adapt and modify policies. The literature reports successful attempts to operationalise Ostrom's principles for the design of enduring self-organising institutions. The literature on EC includes studies of how different biologically-inspired techniques may be used to adapt autonomic components, such as agents, and their social network as a response to environmental changes. A relevant concept in this context is collective self-awareness, which is achieved when the agents are capable of learning from past experience and making decisions autonomously. We also reviewed several applications of GP to a number of problems, e.g. optimisation. Rules of different sorts are represented as trees, e.g. trees of mathematical functions returning a real number used for prioritisation purposes, and are evolved using GP. The sources on CESs are mainly concerned with planning this sort of system and optimising energy consumption, making use of tools for modelling and simulation. The ultimate goal of this dissertation project has been to draw inspiration from the concepts discussed in the reviewed literature to propose a new methodological paradigm for the design of collective adaptive systems. In particular, we have explored methods for approximately optimising policies in an EI using GP.

Concepts and issues addressed	**References on EIs**	**References on EC and GP**	**References on CESs**
Agent-based systems	Yes	Yes	Yes
Self-organisation, self-governance, and other concepts from social, political, and economic science, such as distributive justice and knowledge management, as mechanisms for modification of policies	Yes	No	No
Adaptation of autonomic components in the face of environmental changes through EC, a set of computational methods inspired by Biology	No	Yes	No
Representation of different sorts of rules as trees	No	Yes	No
Optimisation and planning	No	Yes	Yes

Table 2.1: Aspects of related work

Chapter 3

Methodological Approach

In this chapter, we describe the methodological approach followed in this work to solve the problem defined in Section 1.2. We have developed a model of an energy system which encompasses several inter-connected CESs and treats energy as a CPR. We propose two methods, an offline and an online procedure, which enable this system model to optimise its performance through adaptation and evolution of its operating policy. The offline procedure tests alternative policies on the system model *a priori* and returns the one which maximises performance. The online procedure is intended to be applied to systems which are continuously running, evolving increasingly better policies over time.

3.1 Model

The first step towards answering the research question we propose in this work, whether adaptation of the policy of a system through evolution is capable of leading to improved endurance and sustainability, has been to model an energy system encompassing several CESs. While this model has been based on knowledge arising from some of the literature on CESs, reviewed in Section 2.3, it is mostly intuitive. It abstracts a hypothetical energy system connecting several communities which both produce and consume energy and are able to trade it directly amongst themselves. We are not aware of an *actual system of this kind existing* or of *any theories* which can be applied directly to our model; our mental model is the only source of knowledge about the problem domain. This places our modelling approach within the System Dinamycs (SD) paradigm, created by Forrester [For71]. It is important to note that we are not excessively concerned with the *realism* of our model, as it has been designed mostly for enabling the study of adaptation and evolution of operating policies. Vázquez and Liz [VL14] propose an interpretation of the SD modelling paradigm which brings together the approaches of *constructivism* and *expressivism*. According to them, SD modelling entails the *construction of a reality*, starting from a set of mental models of a system. While the components which we use to construct the model are usually *implicit*, they argue that we can use mathematical and computational tools to express these components and their dynamic

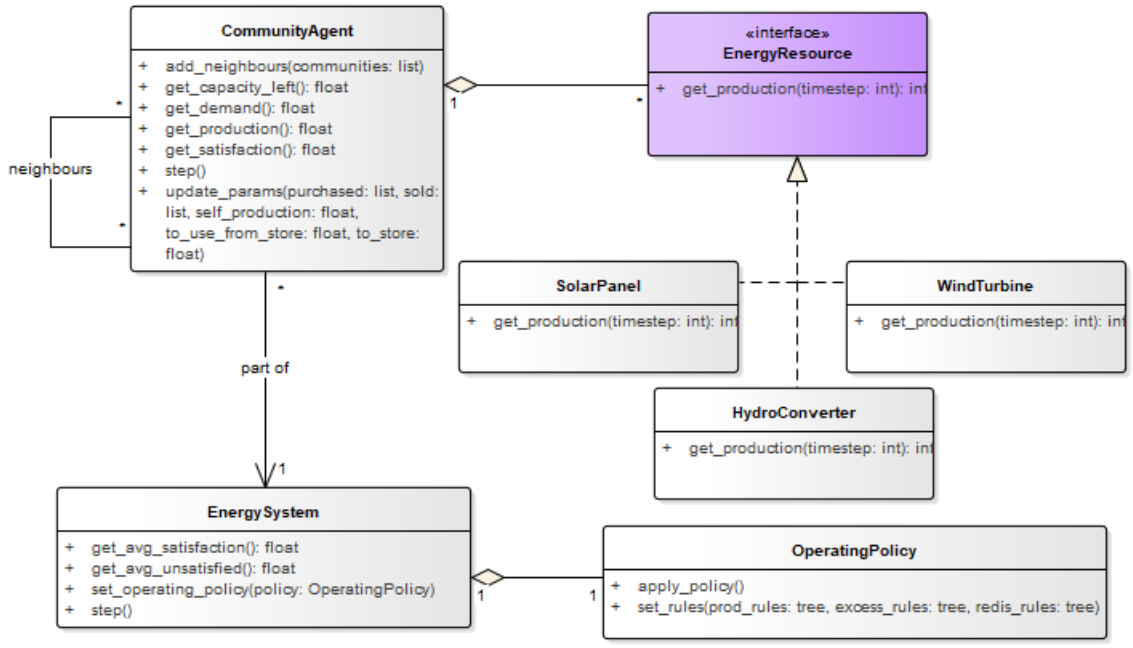

Figure 3.1: Domain model of the energy system

consequences *explicitly* and use the newly acquired knowledge to improve decision-making processes.

In the model which we have constructed, energy is treated as a CPR and communities as agents. The communities are part of an energy system; they have energy demands and can generate energy from a number of renewable resources. Three sources of renewable energy have been considered, namely solar power, wind turbines, and hydropower converters. The energy system can also generate and feed energy to compensate for any lack of self-generated power. Communities are located on a *grid* and have neighbours with which they are able to trade energy, using a simplified version of the Contract Net Protocol [Smi80], and with the central system. A time step corresponds to an hour in the simulation time. At each time step, the energy system uses the current operating policy to determine the mode of operation of the system for that time step, as explained in detail in Section 3.2. Figure 3.1 summarises the domain model of the system.

As an attempt to create a flexible model, we have defined a number of input parameters required for instantiating it. These parameters are summarised in Table 3.1. There are a certain number of communities connected to the energy system, given by *num_communities*. They are placed randomly on a grid, each community occupying a single cell and each cell being occupied, at most, by a single community. The size of the grid is given by parameters *width* and *height*. Each community is able to store energy up until a certain maximum amount, given by its capacity; the capacity is a random positive integer, with *min_capacity* being its minimum value and *maximum_capacity* its maximum value. The cost of storing energy is drawn from a uniform dis-

tribution, such that the minimum possible cost is given by *min_storage_cost* and the maximum cost is given by *max_storage_cost*. The neighbours of a community are determined by the parameter *neighbour_radius*, which is a positive integer representing a number of cells; all communities within this number of cells from a certain community in the grid, including diagonals, are considered its neighbours. The notion of neighbour is important, as communities are only allowed to trade energy with their neighbours. The central system is able to flexibly produce energy at each time step if there is at least one community whose demands have not been satisfied; in this case, the unit cost of production is given by *production_cost*. As part of its operation, the system will purchase energy from the communities and then resell it. The parameter *purchasing_price* is the unit price for which the central system purchases energy and the parameter *selling_price* is the unit price for which it sells it. The parameter w is used to update the satisfaction of each community, as in Equation 3.7, and the parameters $alpha$ and $beta$ are used to calculate the system performance, as in Equation 3.8. The parameter *seed* sets the random seed of the model, ensuring reproducible results. Leaving this parameter undefined causes the model to use sources of randomness offered by Python, the language in which it has been implemented.

The communities are initialised with some renewable energy resources, whose energy production is simulated. The energy produced by each community is obtained at each time step by summing over the energy produced by each of their energy resources. Communities also demand a certain amount of energy at each time step. We have used intuitive notions for coming up with functions which return the energy produced by each resource and each community's demand at every time step. These functions are defined by a sum of *radial basis functions* [BL04]. Given a set of *centres*, $\{c_i\}_{i=1}^N$, a set of *bandwidths*, $\{h_i\}_{i=1}^N$, and a set of weights, $\{w_i\}_{i=1}^N$, the sum of radial basis functions at time step t is given by:

$$S(t|\{w_i\}_{i=1}^N, \{c_i\}_{i=1}^N, \{h_i\}_{i=1}^N) = \sum_{i=1}^{N} w_i \times \exp\left(t \bmod 24 - c_i\right)^2 / h_i^2 \tag{3.1}$$

The number of radial basis functions, N, used to define this sum is given by the model input parameter *num_rbfs*. The greater the value of *num_rbfs*, the more fine-grained the sum defined by Equation 3.1, which enables more complex functions (more inflection points) to be modelled. The weights, centres, and bandwidths are set intuitively according to each energy resource and the expected demand from communities. For each type of energy resource and for the demand of each community, the centres are set to fixed values; the weights and bandwidths, on the other hand, are drawn from probability distributions, as described ahead. For solar panels, the centres are set and the weights are drawn from Gaussian distributions such that there is a maximum at around midday everyday. The bandwidths are each drawn from a uniform distribution. The energy produced by a

solar panel at time step t is then given by:

$$\begin{aligned} & x \sim \mathcal{U}(0,1) \\ & E_{\text{solar}}(t|Q = \{\{w_i\}_{i=1}^N, \{c_i\}_{i=1}^N, \{h_i\}_{i=1}^N\}) \\ & = \begin{cases} (1 - loss\_cloudy) \times \max(S(t|Q), 0) & \text{if } x <= perc\_cloudy \\ \max(S(t|Q), 0) & \text{otherwise} \end{cases} \end{aligned} \tag{3.2}$$

The value *perc_cloudy* is the probability that the weather is cloudy at any given time step and *loss_cloudy* is the decrease (a percentage) in energy produced by the solar panel when the weather is cloudy. They are both drawn from uniform distributions. For both wind turbines and hydropower converters, the centres are set with a fixed step and the weights are drawn from a Gaussian distribution and are independent from the time of the day. The bandwidths are drawn from uniform distributions, as well as a positive *offset* intended to ensure that the energy produced by these two kinds of resources is never 0. At time step t, the energy produced by either a wind turbine or a hydropower converter is then given by:

$$E_{\text{wind,hydro}}(t|Q = \{\{w_i\}_{i=1}^N, \{c_i\}_{i=1}^N, \{h_i\}_{i=1}^N\}) = offset + \max(S(t|Q), 0) \tag{3.3}$$

When defining the function returning a community's demand, the weights are set, again intuitively, so that there is a peak in the early morning and in the evening, by drawing from Gaussian distributions. The demand of a community at time step t is then given by:

$$D(t|Q = \{\{w_i\}_{i=1}^N, \{c_i\}_{i=1}^N, \{h_i\}_{i=1}^N\}) = \max(S(t|Q), 0) \tag{3.4}$$

We defined the sum of radial basis functions, given by Equation 3.1, by calculating the modulo 24 of the input time step t. This has enabled us to define some daily trends, such as the energy produced by solar panels peaking at around midday or the energy demands peaking in the early morning and in the evening. However, it also causes this function to repeat itself every 24 time steps. We address this by adjusting the weights according to another input parameter, *noise*. We start by generating a base set of weights, $\{w'_i\}_{i=1}^N$, as described before. When calculating the energy produced by each energy resource or the demand of a community at a given time step, we create a new set of weights, $\{w_i\}_{i=1}^N$, by adding a random perturbation to each w'_i which depends on the *noise* parameter:

$$\begin{aligned} & \sigma \sim \mathcal{U}(-noise, noise) \\ & w_i = w'_i + \sigma \end{aligned} \tag{3.5}$$

The greater the value of *noise*, the greater the potential difference between w_i and w'_i and, therefore, the greater the *unpredictability* of the functions which return the energy produced by

num_communities	The number of communities in the energy system
width	Width of the grid (number of cells) where the communities are located
height	Height of the grid (number of cells) where the communities are located
num_rbfs	Number of *radial basis functions* used for defining the functions that return the energy produced by each energy resource at a given time step
noise	Positive value for introducing unpredictable variation in the amount of energy produced by each energy resource
min_capacity	The minimum storage capacity of any community in the system
max_capacity	The maximum storage capacity of any community in the system
min_storage_cost	The minimum unit cost for a community to store energy
max_storage_cost	The maximum unit cost for a community to store energy
neighbour_radius	Radius (number of cells) used to determine the neighbours of each community
production_cost	The unit cost for the central energy system of producing energy
purchasing_price	The unit price for which the central energy system purchases energy from the communities
selling_price	The unit price for which the central energy system sells energy to the communities
w	Weight of latest utility when updating the satisfaction of a community (Equation 3.7)
alpha	Weight of the average satisfaction when calculating the performance of the system (Equation 3.8)
beta	Weight of the average proportion of unsatisfied communities when calculating the performance of the system (Equation 3.8)
seed	Random seed (for obtaining reproducible results)

Table 3.1: Model parameters

each energy resource and the demand for each community. This model and particularly the functions simulating energy production are merely intuitive and make many simplifying assumptions. However, as we mentioned earlier in this section, realism has not been a major concern and the plausibility of the model we have created is satisfactory. It has enabled us to study the impact of adaptation and evolution of system policies.

3.2 Representation and Application of Policies

At every time step, each community's energy demand and the total energy which they have produced from their resources is calculated. The system's operating policy is then applied, returning a *mode of operation* which will determine what to do with the energy the communities have produced and how to meet their demands. In our model, the mode of operation has three degrees of freedom:

- What the communities should do with the energy they have produced at the current time step: either use it to satisfy their own demands (self-supply) or sell it all to the central system.
- In the case of self-supply, what the communities should do with any excess of energy: sell to the central system; store as much as capacity allows and sell the excess; trade it with neighbours and sell the excess; store, trade, and sell; or trade, store, and sell.
- If any demands have not been satisfied, the central system will ensure they are met by first reselling the energy which has been purchased from the communities and producing energy on demand (accounting for production costs) when necessary. The order in which the communities receive the energy is determined by several possible criteria: greatest demand, greatest production, lowest satisfaction, random, or ration.

Table 3.2 summarises these degrees of freedom and the values they can be set to when instantiating a mode of operation. For example, a possible mode of operation would be the triple {SELF_SUPPLY, STORE + TRADE + SELL, GREATEST_DEMAND}. There are therefore $2 \times 5 \times 5 = 50$ possible modes of operation which can be selected at each time step.

Regarding the first degree of freedom, if the value SELL is selected, then all the energy produced by each community at the current time step will be sold to the central system, the unit price being the input parameter *purchasing_price*. We make the simplifying assumption that the central system has a sufficiently large storage capacity and may therefore purchase all the energy which the communities have produced. If the value SELF_SUPPLY is selected, then the communities will try to meet their energy demands with their own *energy assets*. The energy assets of a community are the total energy it has available at the current time step: the energy produced by its resources and the energy it has stored. The communities use the energy they have produced and the energy stored, in this order, to cover their own demands; they may only use energy which has been stored after all the energy produced has been used.

When the communities are using a self-supply of energy, they may find themselves with an excess of energy produced if they are able to completely cover their own demands. This excess is handled according to the value of the second degree of freedom. The possible values represent the order in which the different options (selling to the central system, trading with other communities, or storing) should be tried. For example, the value STORE + TRADE + SELL means that communities should first attempt to store as much of the excess as their storage capacity allows; if there is still any energy left after storing, they will make it available for their neighbours to purchase (trade); and, if there is still any energy left after the trade, they will sell it to the central system. All possible values end with the option SELL because the communities may not be able to allocate all of their excess of energy by storing it or trading it with their neighbours. As before, the unit price of any energy sold to the central system is given by *purchasing_price*. Communities trade energy by using a simplified version of the Contract Net Protocol [Smi80]; taking turns in random order, each community whose demand has not yet been satisfied attempts to purchase as much energy from its neighbours as possible, sorting them by the least selling price. When trading its excess of energy, each community has a unit selling price which is drawn from the uniform distribution $\mathcal{U}(selling\_price/2, selling\_price)$, where *selling_price* is the model input parameter which sets the unit price for which the central system sells energy to the communities (Table 3.1). Storing energy has a unit cost associated, which, as mentioned in Section 3.1, is drawn from the uniform distribution $\mathcal{U}(min\_storage\_cost, max\_storage\_cost)$, where *min_storage_cost* and *max_storage_cost* are also input parameters of the model.

If the communities are not using a self-supply of energy *or* if there are communities whose demands have not been fully satisfied, then the central system will sell them energy. It starts by selling the energy it has stored, which has been previously purchased from the communities, for the unit price *selling_price*. If the energy in the storage is not enough, then the system will produce more energy. The selling price of this additional energy now increases according to a unit production cost given by the model input parameter *production_cost*. This means that, when this energy sale takes place, the last communities may have to purchase it for a higher unit price. The third degree of freedom of the mode of operation determines the order in which the communities purchase energy from the central system. They can be sorted by greatest demand, greatest production (both measured at the current time step), lowest satisfaction (which is cumulative, as explained ahead), or randomly. If the value RATION is selected, then the energy stored by the central system will be split equally among the communities; each community receives at most this ration and any additional demand is met with energy produced by the central system at a greater unit cost.

Applying the selected mode of operation at time step t determines how much energy each community will produce for itself and how much it will sell, purchase, and store for the following time step. The resulting costs and revenues are taken into account when calculating the *utility* of

Degree of Freedom	Possible Values
Production	SELL, SELF_SUPPLY
Excess	SELL, STORE + SELL, TRADE + SELL, STORE + TRADE + SELL, TRADE + STORE + SELL
Redistribution	GREATEST_DEMAND, GREATEST_PRODUCTION, LOWEST_SATISFACTION, RANDOM, RATION

Table 3.2: Degrees of freedom and their possible values when instantiating a mode of operation

this energy allocation for community i at time step t, u_t^i:

$$u_t^i = \text{total\_sales}_t^i - \text{total\_purchases}_t^i - \text{storage\_costs}_t^i \tag{3.6}$$

The cumulative satisfaction for community i at time step t is calculated with the most recent utility value as in Equation 3.7. The w parameter weights the importance of past satisfactions and the current utility when updating a community's satisfaction.

$$s_t^i = (1-w) \times s_{t-1}^i + w \times u_t^i, \text{ with } s_0^i = 0 \tag{3.7}$$

The operating policy which selects the mode of operation at each time step is represented by a triple of decision trees, one for each degree of freedom. The reason why we use decision trees is the need to evolve and adapt system policies using GP, which traditionally operates over tree structures. The inner nodes of the tree test the values of system-wide variables which are collected at each time step, returning a Boolean value (i.e., the decision trees are binary). Based on the literature about EIs, CPR management, and CESs, as well as on intuitive knowledge regarding the system model we have created, we have selected the following system variables to be collected and tested at each time step:

- Number of communities (fixed throughout a model run)
- Average satisfaction across all communities
- Total energy production at the communities
- Total self-supply of energy
- Average difference between self-supplied energy and demand
- Average difference between current assets (energy produced and stored) and demand
- Number of unsatisfied agents (negative satisfaction)
- Total energy stored
- Total demand

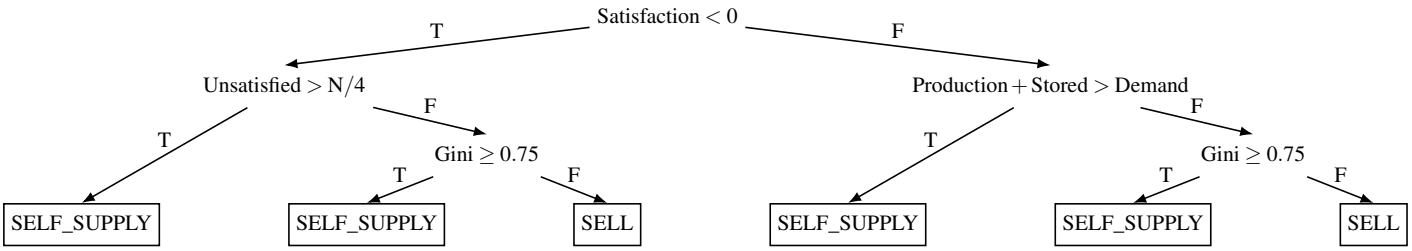

Figure 3.2: Default decision tree encapsulating rules which determine what the communities should do with the energy they have produced at each time step

- Total difference between current assets (energy produced and stored) and demand
- Average capacity left
- Gini index of satisfaction inequality

As a first step, we devised a default policy whose baseline performance could be compared to that of the operating policies which are evolved by our procedures. As an example of the type of decision trees which are evolved and manipulated by our procedures, Figures 3.2, 3.3, and 3.4 show the trees which make up the triple representing this default policy, selecting what the communities should do with the energy they have produced at each time step, what to do with any excess of energy in the case of self-supply, and the criteria under which the central system should redistribute energy, respectively. This default policy is the result of our intuition about the model we have created. Each function (inner) node of the decision trees could be *any test* on the values of system variables; we do not have a fixed set of function nodes. For this reason, they are represented internally as trees of operations. The root node of these trees is always a binary Boolean operator, whereas all the other inner nodes are binary arithmetic operators. The leaf nodes are either the values of system-wide variables or real constants. This representation has enabled us to generate random function nodes when building new decision trees automatically with our optimisation procedures.

We have considered two approaches for approximately finding an optimal policy for the system. The first approach is GP optimisation and is detailed in Section 3.3. The second approach consists of adapting and evolving policies in runtime and is explained in Section 3.4.

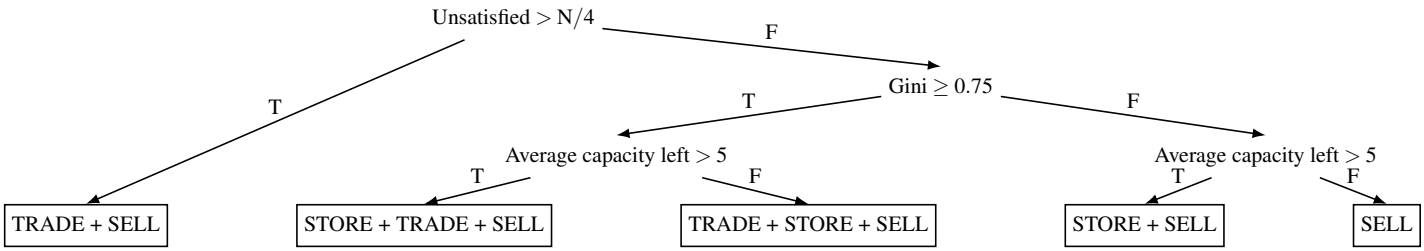

Figure 3.3: Default decision tree encapsulating rules which determine what the communities should do at each time step with any excess of energy (in the case of self-supply)

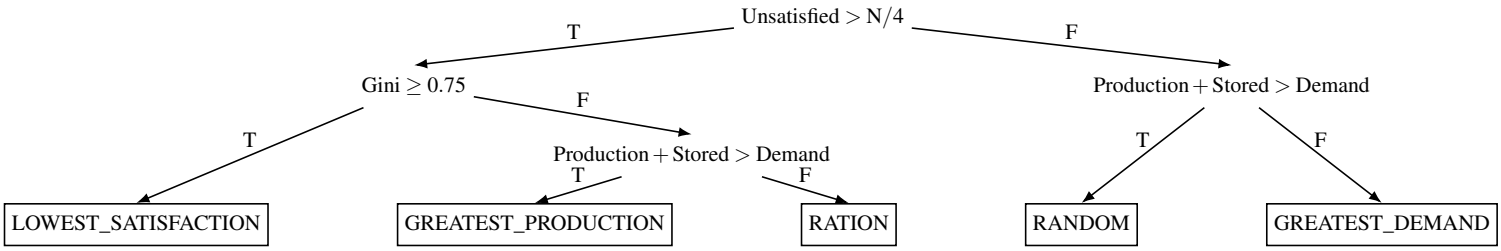

Figure 3.4: Default decision tree encapsulating rules which determine how the central system redistributes energy

3.3 Offline Procedure

In order to find an optimal operating policy *a priori*, we have implemented a GP algorithm which evaluates alternatives by running the model with each of a population of policies for the number of time steps corresponding to a week (168, since time steps correspond to hours). A performance metric for each policy i is calculated as shown in Equation 3.8, where satisfaction and the proportion of unsatisfied agents are averaged out after a week has passed. An agent is unsatisfied at a certain time step if its satisfaction is negative. The α parameter determines how much the average satisfaction favours the performance measure and the β parameter determines how much the average proportion of unsatisfied agents penalises it ($\alpha > 0$ and $\beta \geq 0$). They are both input parameters of the model, as mentioned in Section 3.1.

$$\begin{aligned} \text{performance} = \alpha \times \text{average satisfaction} \\ - \beta \times \text{average proportion of unsatisfied agents} \end{aligned} \tag{3.8}$$

The procedure begins by randomly generating an initial population of operating policies (each a triple of decision trees, as explained before) using "ramped half and half" [Koz92], with the possible values for each degree of freedom as leaf nodes. There is usually a fixed and finite set of function nodes in GP optimisation; however, as we have mentioned before, the set of possible function nodes for the decision trees is theoretically infinite in this case, as they can be any test on the values of system variables, so we randomly initialise a large set of these nodes at the start of the procedure. Since they are represented internally as trees of operations, we also use "ramped half and half" to generate them, with a fixed set of Boolean and arithmetic operators as inner nodes and the set of system variables as leaf nodes.

At each iteration of the optimisation procedure, the operating policies are evaluated by running *instances of the same model* for 168 time steps (a week) and computing a performance value, as in Equation 3.8. Running the model in order to obtain a performance value is a costly operation; for this reason, the alternative policies are tested on the model *in parallel*. The performance values are then used as fitness values to evolve a new generation using standard GP operations, namely reproduction, crossover, and mutation, which are described by Koza [Koz92]. Reproduction randomly selects individuals to be copied to the following generation with a probability which depends on

their fitness value. If an elitist strategy is in place, then the policies with the highest fitness values are copied on to the following generation. The crossover operation randomly selects pairs of individuals, again with a probability which is higher the higher their fitness, and crosses them element-wise, each element being a tree in the triple which makes up an operating policy. The mutation operation also selects individuals based on their fitness and creates new individuals by replacing parts of their trees with randomly generated subtrees; its goal is to introduce variability when searching for new solutions. For all of these operations, the probability of selecting a policy i is a Softmax probability calculated with the fitness values of the policies after they have been evaluated on the system, as in Equation 3.9. The reason why we have used Softmax probabilities is that they grow monotonically with the fitness values and this formulation handles negative fitness values elegantly.

$$p_i = \frac{e^{\text{fit}_i}}{\sum_j e^{\text{fit}_j}} \tag{3.9}$$

This offline procedure has some parameters which are summarised in Table 3.3. When generating the decision trees in the initial population, the procedure requires a parameter indicating their maximum depth, *max_dt_depth_gen*; the generated trees have depths ranging from 1 to *max_dt_depth_gen* (the root has depth 0). The "ramped half and half" method uses two approaches to generate trees, "grow" and "full". The parameter *n_dt_each* sets the number of trees to be generated for each possible maximum depth (from 1 to *max_dt_depth_gen*) and each generation method ("grow" and "full"). As mentioned before, the function nodes in the decision trees are represented by trees of operations which are also generated using "ramped half and half"; *n_ot_each* and *max_ot_depth_gen* are the analogous parameters for generating these trees. When evolving a new generation, the crossover operation may produce decision trees whose depth is larger than *max_dt_depth_gen*; the parameter *max_dt_depth* sets the maximum permissible depth for trees resulting from crossover to be included in the following generation.

The procedure keeps track of the best operating policy it has found so far and returns it after a certain number of generations have been evolved. This policy is the one which led to the greatest performance value after running the model, and, therefore, is approximately optimal. Algorithm 1 describes the procedure in pseudocode.

3.4 Online Procedure

Adapting and evolving policies in runtime has posed further challenges. When searching the space of possible operating policies in order to optimise system performance, we do not have a way of assigning a fitness value to alternative policies in order to compare them *a priori*, as would be necessary to implement "hill climbing" or other local search methods. The performance of a policy must be measured by first running the model with it for a certain amount of time. The method we propose draws inspiration from both GP and RL. An initial population of operating policies is randomly generated using the "ramped half and half" method, just as in the offline

Algorithm 1 Finding an optimal policy *a priori* using GP

Choose model *params* to generate instances of the same model
Choose parameters for the initial population: *max_dt_depth_gen*, *n_dt_each*, *max_ot_depth_gen*, *n_ot_each*
Choose parameters for evolution: *max_dt_depth*, *copy_perc*, *cross_perc*, *mut_perc*, *elitist*
Choose number of iterations: N
$population \leftarrow$ generate_initial_population($max\_dt\_depth\_gen, n\_dt\_each, max\_ot\_depth\_gen$), n_ot_each)
$max\_fit \leftarrow -\infty$
$best\_policy \leftarrow \emptyset$
for $Gen = 1, Gen \leq N; Gen \leftarrow Gen + 1$ **do**
    $fits \leftarrow \emptyset$
    for $policy \in population$ **do**
        $model \leftarrow$ Model($params$)
        $model$.set_policy($policy$)
        for $i = 0, i < WEEK\_DURATION, i \leftarrow i + 1$ **do**
            $model$.step()
        end for
        $fit = model$.get_average_fitness() (Equation 3.8)
        if $fit > max\_fit$ **then**
            $max\_fit \leftarrow fit$
            $best\_policy \leftarrow policy$
        end if
        $fits \leftarrow fits \cup \{(policy, fit)\}$
    end for
    $population \leftarrow$ evolve_new_generation($population, fits, max\_dt\_depth, copy\_perc$, $cross\_perc, mut\_perc, elitist$)
end for
return *best_policy*

n_dt_each	Number of decision trees to be generated for each tree depth and generation method ("grow" and "full") using "ramped half and half"
max_dt_depth_gen	Maximum depth for the decision trees generated using "ramped half and half"
max_dt_depth	Maximum depth for decision trees resulting from the crossover operation
n_ot_each	Number of trees of operations, which represent the function nodes in the decision trees, to be generated for each tree depth and generation method ("grow" and "full") using "ramped half and half"
max_ot_depth_gen	Maximum depth for the trees of operations generated using "ramped half and half"
copy_perc	Percentage of policies to be copied on to the following generation (by either an elitist or non-elitist strategy)
cross_perc	Percentage of policies in the following generation resulting from the crossover operation
mut_perc	Percentage of policies in the following generation resulting from the mutation operation
elitist	Boolean parameter indicating whether or not to employ an elitist strategy when copying policies on to the following generation

Table 3.3: Parameters of the offline procedure

procedure described in Section 3.3, and a policy is selected when the model starts running. A decision is made periodically about which operating policy in the current population should be tried. An operating policy is selected every 24 time steps (hours) based on Softmax probabilities calculated from the current fitness values. After a policy i has been put in use for 24 time steps, a *reward* is calculated based on the observed performance:

$$r_i^t = \text{performance}^t \tag{3.10}$$

The observed system performance, performance^t, is calculated as in Equation 3.8, considering the average satisfaction and average proportion of unsatisfied agents over the most recent 24 time steps. The fitness value of a policy i, fit_i, is initialised to 0. After applying policy i on the system for 24 time steps, its fitness value is updated as follows:

$$\text{fit}_i \leftarrow \begin{cases} r_i^t & \text{if the policy had not yet been tried} \\ (1-\Omega) \times \text{fit}_i + \Omega \times r_i^t & \text{otherwise} \end{cases} \tag{3.11}$$

Ω is the *learning rate*, weighting the importance of the most recent reward to the overall fitness of the operating policy. At each 336 time steps, *two weeks' time*, the fitness values are used to evolve a new generation of operating policies, using standard GP techniques as those described earlier. In order to promote variability among individuals, new random policies are added to each generation, besides those resulting from the reproduction, crossover, and mutation operations. The probability that a policy is selected for any of these operations when evolving a new generation is given by Equation 3.9, as in the offline procedure described in Section 3.3. However, when

calculating Softmax probabilities for selecting policies to be *tried* on the system every 24 time steps, we have found it beneficial to make the probability distribution more uniform by dividing all fitness values by a *temperature* parameter, which is a positive value that is decremented over time, divided by 2 every 168 time steps (a week's time) until it reaches 1. This is intended to promote early *exploration* of many different policies and thereby to prevent premature convergence to good but sub-optimal policies by making initial differences between fitness values less relevant. This way, the probability of selecting a policy i to be tried on the system at time step t is:

$$p^t_{\text{selection}\,i} = \frac{e^{\frac{\text{fit}_i}{\tau^t}}}{\sum_j e^{\frac{\text{fit}_j}{\tau^t}}} \tag{3.12}$$

Where τ^t is the temperature value at time step t. After some time has passed, the population size is decreased linearly over time as the procedure singles out the best policies; this increases the probability that the best policies are selected to be tried on the system, promoting convergence, less exploration, and more *exploitation*. If an elitist strategy is employed, the best policies found so far are guaranteed to be passed on to the following generation, thus becoming increasingly likely to be selected as less and less exploration takes place.

The parameters of this online procedure are summarised in Table 3.4. Most of them are the same as those for the offline procedure, described in Section 3.3, with three new parameters. The learning rate (Ω) used in Equation 3.11 is given by the parameter lr. The initial temperature, τ^0, is given by *initial_temperature* and the parameter *gen_threshold* sets the number of generations after which the population size is decreased linearly each time a new generation is evolved. A high-level pseudocode description of this runtime procedure is given in algorithm 2.

This approach does, in our view, address the problem of reconciling the following:

- We want to evolve and adapt the current set of policies, converging to an approximately optimal performance.
- We are unable to know how good a policy is until it has been tried on the system model.
- Policy selection and adaptation must be done in runtime; the system must not backtrack after trying a policy and policies must be tried sequentially.

The method we propose is intended to be a mechanism for enabling exploration of different policies, *ideally* converging to policies which maximise performance. Past history is taken into account when iteratively updating the fitness values of the operating policies which have been tried, drawing inspiration from RL techniques in the sense that we reward good policies and penalise bad policies after their performance on the system has been observed. The GP part of the procedure is the search method, intended to find a population of policies which approximately optimise system performance, using the fitness values to evolve new generations.

n_dt_each	Number of decision trees to be generated for each tree depth and generation method ("grow" and "full") using "ramped half and half"
max_dt_depth_gen	Maximum depth for the decision trees generated using "ramped half and half"
max_dt_depth	Maximum depth for decision trees resulting from the crossover operation
n_ot_each	Number of trees of operations, which represent the function nodes in the decision trees, to be generated for each tree depth and generation method ("grow" and "full") using "ramped half and half"
max_ot_depth_gen	Maximum depth for the trees of operations generated using "ramped half and half"
copy_perc	Percentage of policies to be copied on to the following generation (by either an elitist or non-elitist strategy)
cross_perc	Percentage of policies in the following generation resulting from the crossover operation
mut_perc	Percentage of policies in the following generation resulting from the mutation operation
elitist	Boolean parameter indicating whether or not to employ an elitist strategy when copying policies on to the following generation
lr	Learning rate (Ω parameter in Equation 3.11)
initial_temperature	Initial temperature value, τ^0
gen_threshold	Number of generations evolved after which the population size is decreased linearly

Table 3.4: Parameters of the online procedure

Algorithm 2 Evolving a population of operating policies in runtime

Choose model *params*
Choose parameters for the initial population: *max_dt_depth_gen*, *n_dt_each*, *max_ot_depth_gen*, *n_ot_each*
Choose parameters for evolution: *max_dt_depth*, *copy_perc*, *cross_perc*, *mut_perc*, *elitist*, *lr*, *initial_temperature*, *gen_threshold*
$model \leftarrow$ Model($params$)
$population \leftarrow$ generate_initial_population($max\_dt\_depth\_gen, n\_dt\_each, max\_ot\_depth\_gen$)
$current\_policy \leftarrow$ select_random_policy($population$)
$model$.set_policy($current\_policy$)
$temperature \leftarrow initial\_temperature$
$fits \leftarrow \emptyset$
$initial\_population\_size \leftarrow$ len($population$)
$population\_size \leftarrow initial\_population\_size$
$generation \leftarrow 1$
while not terminated **do**
    $model$.step()
    if $timestep$ mod $DAY\_DURATION = 0$ **then**
        $reward \leftarrow model$.get_last_avg_fitness()
        $fitness \leftarrow$ update_fitness($current\_policy, reward$) (Equation 3.11)
        if $current\_policy$ not in $fits$ **then**
            $fits \leftarrow fits \cup \{(current\_policy, fitness)\}$
        else
            Update $fits$ with $(current\_policy, fitness)$
        end if
        $current\_policy \leftarrow$ select_random_policy($population, fits, temperature$)
        $model$.set_policy($current\_policy$)
    end if
    if $timestep$ mod $WEEK\_DURATION = 0 \wedge temperature > 1$ **then**
        $temperature \leftarrow \max(temperature/2, 1)$
    end if
    if $timestep$ mod $(2 \times WEEK\_DURATION) = 0$ **then**
        if $generation > gen\_threshold \wedge population\_size > initial\_population\_size/2$ **then**
            $population\_size \leftarrow \max(population\_size - initial\_population\_size/3,$
            $initial\_population\_size/2)$
        end if
        $population \leftarrow$ evolve_new_generation($population, fits, population\_size,$
        $max\_dt\_depth, copy\_perc, cross\_perc, mut\_perc, elitist$)
        $generation \leftarrow generation + 1$
    end if
end while

3.5 Summary

In this section, we summarise the main aspects of the methodological approach presented in this chapter. We have developed a model of an energy system which encompasses several interconnected CESs and treats energy as a CPR. At each time step, a policy is applied to determine a mode of operation, which concerns the way the energy produced by the communities is used to satisfy their demands. We have proposed two methods, an offline and an online procedure, which enable this system model to optimise its performance through adaptation and evolution of its operating policy. The offline procedure assumes the existence of a system model upon which alternative policies may be tested beforehand and finds a policy which optimises performance during a single simulation week, following a standard GP approach. The online procedure makes no such assumption and is intended to be applied to systems which are continuously running; it is a hybrid approach which draws inspiration from GP and RL to evolve policies over time, using past performance history to promote the presence of increasingly better policies in each new generation.

Chapter 4

Experimental Results

In this chapter, we present and discuss the results of the experiments we have carried out upon our system model with the methods we propose. We have compared the performances of both the offline and the online procedures in terms of the quality of the solutions they return and have also conducted sets of experiments for analysing the influence of some of the parameters of the online procedure.

4.1 Performance of the Offline Optimisation Procedure

Regarding the offline procedure for the optimisation of a single operating policy *a priori*, experiments have been carried out as an attempt to answer the following questions:

1. For the same model instance, to what extent is the quality of the evolved policy (in terms of the resulting system performance) robust with respect to the stochastic nature of the optimisation procedure?

2. For the same model instance, are the solutions obtained with different runs of the optimisation procedure similar in terms of their consequences, i.e., are the same modes of operations applied in the same context?

In order to answer these questions, the optimisation procedure described in Section 3.4 was run 30 times on instances of the same model, each time returning an operating policy which approximately maximises the performance metric given by Equation 3.8. Table 4.1 presents the values with which the input parameters of the model have been instantiated for this set of experiments. The decision trees in the initial population had a maximum depth of 3, with a maximum permissible depth of 5 for new trees resulting from crossover. *n_dt_each* is 3, which results in a population size of 18[1]. The trees of operations which represent the function nodes of the decision trees have a maximum depth of 3 and *n_ot_each* is 20, which results in a set of 120 function nodes. When evolving a new generation, 10% of the new population results from the reproduction operation

[1]While this would be a small population size for many GP problems, we have empirically determined it to be appropriate in this case.

num_communities	50
width	10
height	10
num_rbfs	10
noise	5
min_capacity	10
max_capacity	30
min_storage_cost	0.1
max_storage_cost	0.3
neighbour_radius	5
production_cost	0.5
purchasing_price	1
selling_price	1.2
w	0.5
alpha	1
beta	5
seed	1995

Table 4.1: Model input arguments used in the experiments

(non-elitist), 40% from crossover, and 50% from mutation. Table 4.2 summarises this parameter setting for the offline procedure.

The median maximum performance value after 30 executions of the procedure was 8.122. The sample standard deviation was 2.94×10^{-2}, which shows that there is little variation in the maximum performance value when running the procedure several times. Although this is dependent on the problem domain and on how easy it is to find an optimal solution, it still enables us to conclude that the procedure is indeed robust with respect to the stochastic nature of GP, as the performance of the solutions found is approximately the same for the same model when comparing different executions. The performance value obtained for the same model with our default policy was 3.519, showing how hard it is for a system designer to find an optimal policy and the usefulness of the optimisation procedure. The policy obtained using GP (approximate) optimisation results in a *clearly better performance* when compared to the default policy we designed.

n_dt_each	3
max_dt_depth_gen	3
max_dt_depth	5
n_ot_each	20
max_ot_depth_gen	3
copy_perc	0.1
cross_perc	0.4
mut_perc	0.5
elitist	False, copy individuals probabilistically

Table 4.2: Arguments passed to the offline optimisation procedure

Median maximum performance	8.122
Sample standard deviation of the maximum performance	2.94×10^{-2}
Median number of unique modes of operation at each time step	5
Performance with the baseline default policy	3.519

Table 4.3: Results after 30 runs of the offline optimisation procedure (baseline default policy for comparison)

In order to answer the second question, we then took each of the 30 operating policies obtained and compared the modes of operation selected at each time step. Recall that a mode of operation is given by instantiating the three degrees of freedom mentioned in Section 3.1. We then counted the number of unique modes of operation selected at each time step; the median value was 5. This means that, in the case of our system model, there are several locally optimal policies, resulting in different sequences of modes of operation, yielding approximately the same system performance. There is, however, a certain degree of similarity between these sequences, given the median value of 5 out of a possible maximum of 30 unique modes of operation at each time step (given that there are 50 possible modes, as mentioned in Section 3.2). Indeed, most of the time the policy selects the value SELF_SUPPLY for the first degree of freedom and TRADE + SELL for the second degree of freedom; the third degree of freedom (controlling how the energy is redistributed among the communities) is the one showing most variability. Table 4.3 summarises the results obtained after 30 runs of the offline optimisation procedure.

All solutions obtained from different runs have approximately the same performance value. However, the fact that these solutions are fairly diverse in terms of the sequences of modes of operation in which they result[2], as discussed above, indicates that they are, in fact, local optima and that there could be an even better solution which the procedure has failed to find. We began the discussion in this report by claiming that it is hard to find an appropriate policy given a certain environment. Indeed, since the mode of operation chosen at a given time step will affect future performance in the case of our system model, we would have to consider all possible sequences of modes up until a certain time step, select one which maximises performance, and then come up with a set of rules which results in that sequence. This is a combinatorial problem which quickly becomes intractable as the final time step grows and this formulation is only applicable to cases where the final time step is bounded; in real-world cases, the system is continuously running and our online method for adaptation and evolution of the operating policy in runtime seems more useful.

4.2 Performance of the Online Optimisation Procedure

The offline optimisation procedure returns a single policy which has been evaluated on the system for 168 time steps (a week). The online procedure, on the other hand, tests several policies on

[2]This refers to functional diversity (a sequence of modes of operation is a consequence of applying one or more policies to the system over time), rather than structural diversity (the shape of the trees which make up a policy).

n_dt_each	2
max_dt_depth_gen	3
max_dt_depth	5
n_ot_each	20
max_ot_depth_gen	3
copy_perc	0.2
cross_perc	0.1
mut_perc	0.1
elitist	True
lr	0.5
initial_temperature	320
gen_threshold	3

Table 4.4: Arguments passed to the online optimisation procedure

the system over time for a number of time steps corresponding to many weeks, with one policy affecting the performance of subsequent policies. In this section, we attempt to compare the performance of the online procedure to that of the offline procedure by calculating an average weekly performance (last 168 time steps), but the reader should keep in mind that the performance metrics for both procedures are not exactly the same. Regarding the online optimisation procedure, experiments have been carried out as an attempt to answer the following questions:

1. Is the system able to improve its performance over time by evolving and adapting its set of rules?

2. For the same model instance, does the system usually converge to approximately the same performance as the one obtained by running the offline optimisation procedure?

In order to answer the questions above, we have executed the online procedure upon the same model instance 30 times. At each time step, daily (last 24 time steps) and weekly (last 168 time steps) performance values have been calculated, with the goal to see how many times the weekly performance successfully converged to a value close to 8, which is the *approximately optimal value found by the offline procedure*. The model input parameters are instantiated as before, summarised in table 4.1. The maximum depth for both the decision trees and the trees of operations (for the function nodes) is the same as before, as well as the number of function nodes randomly generated. *n_dt_each* is now 2, which makes the initial population size 12. The initial *temperature* is 320, the learning rate is 0.5, and, when evolving a new generation, 20% of the new population is the result of copying individuals using an elitist strategy, 10% is the result of crossover, 10% is the result of mutation, and the remaining 60% are new policies generated randomly with the intention of introducing more variability and preventing early convergence to sub-optimal policies. This parameter setting is summarised in table 4.4.

Figure 4.1 shows the weekly performance after 1800 time steps for each of the runs. The performance values tend to be close to the one reported in Section 4.1, which means that the

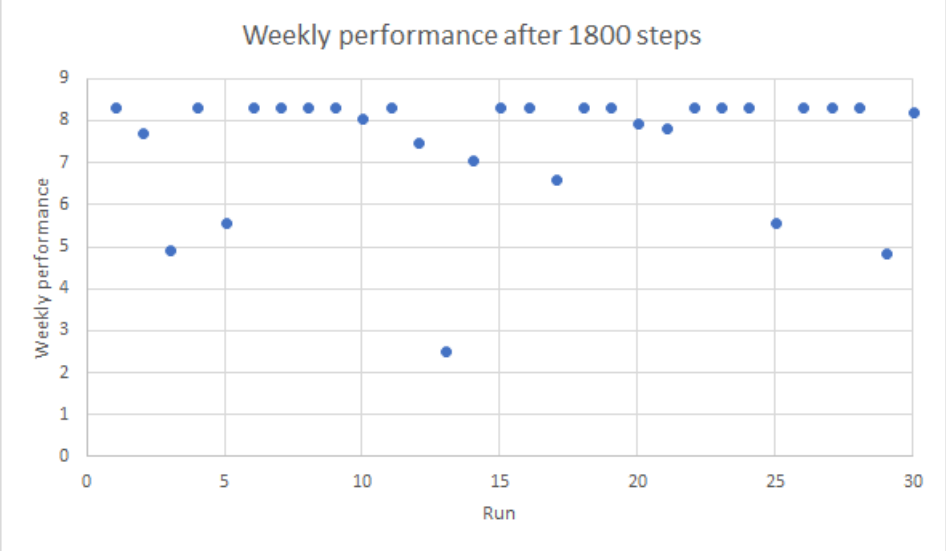

Figure 4.1: Weekly performance after 1800 steps for each of the 30 runs of the online procedure

online procedure does usually converge to the same performance as the one obtained with the offline procedure. These are good results, considering that the procedure is essentially testing several policies in runtime, *optimising by means of trial and error*. However, convergence is expected to depend on how easy it is to find an optimal system policy in a particular problem domain. We argue that it is more important to converge to a population of good policies than it is to find an optimal policy, even though the method did converge to the hypothetically optimal performance (the one obtained with the offline optimisation procedure) in most of the tests which have been carried out.

The graph of Figure 4.2 shows how the daily and weekly performance values evolve over time for one of the runs, in which the performance converged to a value close to the performance obtained with the offline procedure. The graph shows that the procedure is able to improve system performance over time. With some initial instability caused by exploration of several different policies (due to a larger value of the temperature parameter), the weekly fitness increases over time, converging to a value close to 8. Online adaptation and evolution of system policies seems to have more practical advantages if we realistically assume that the system behaviour over time is not known, or hard to predict, *a priori* and that the system is running continuously, without a bounded final time step. These assumptions seem appropriate for real-world use cases of EIs.

4.3 Influence of the Parameters of the Online Procedure on the Quality of the Solutions

In this section, we analyse the influence of some of the parameters of the online procedure upon the quality of the solutions with two sets of experiments. For each set of experiments, we have fixed all model parameters, as well as some of the parameters of the procedure, with the same

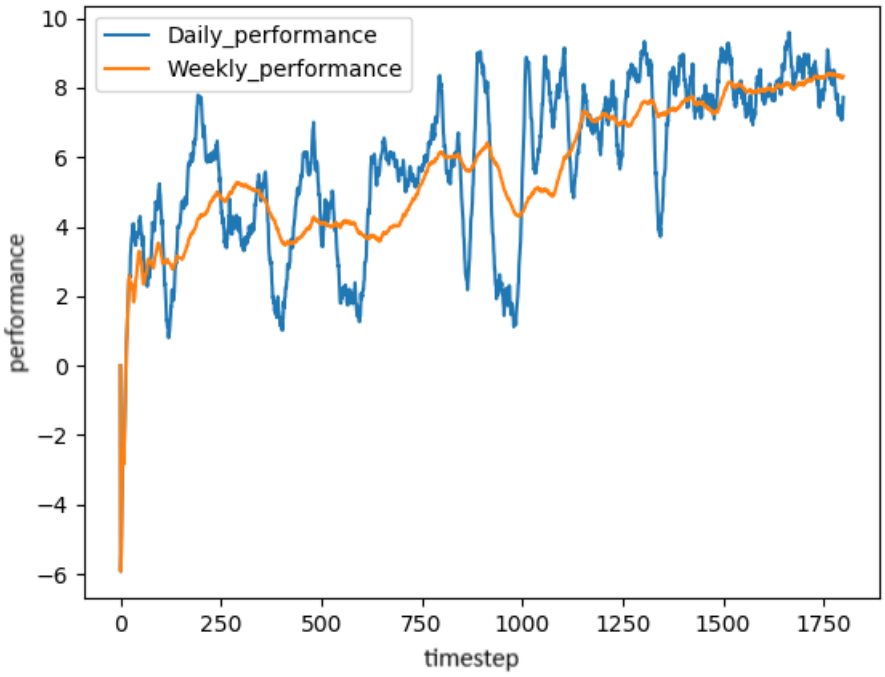

Figure 4.2: Daily (last 24 time steps) and weekly (last 168 time steps) performance when adapting and evolving the operating policy in runtime

values as before (tables 4.1 and 4.4) and have tried different combinations of parameter values by varying the learning rate, the initial temperature, the percentage of individuals copied on to the following generation, the percentage of individuals resulting from crossover, and the percentage of individuals resulting from mutation. Table 4.5 summarises how the value of each parameter has been varied to create different parameter settings for these experiments. Note that we have only included settings for which $copy\_perc + cross\_perc + mut\_perc \leq 1$, which results in a total of 1230 combinations. For each setting, we have run the online procedure on the model 30 times, for a maximum of 1800 time steps each. Table 4.6 summarises these details about each of the sets of experiments which have been carried out.

For each run with each parameter setting, we have calculated the weekly performance after 1800 time steps, which correspond to just under 11 weeks. We have drawn box plots in order to visualise how the distribution of weekly performance values after 1800 time steps varies for

lr	0.1 to 1, with a step of 0.1
initial_temperature	{40, 180, 320}
copy_perc	0.2 to 0.5, with a step of 0.1
cross_perc	0 to 0.8, with a step of 0.2
mut_perc	0 to 0.8, with a step of 0.2

Table 4.5: Range of values for each parameter for testing several parameter settings

Total number of parameter settings	1230
Number of runs for each parameter setting	30
Number of time steps for each run	1800

Table 4.6: Summary of each set of experiments testing several parameter settings

different values of each of the parameters in table 4.5, each time clustering the run data by the possible values of a *single parameter*. In a first instance, we have run the online procedure with the original search space of policies, as described in Section 3.2; the results are presented and discussed in Section 4.3.1. With the goal of studying the performance of our method in a less favourable scenario, we have modified the search space so that the optimal solution is harder to find and run a second set of experiments; the results are presented and discussed in Section 4.3.2.

4.3.1 Experiments with the Original Search Space

This section presents and discusses the results of the set of experiments conducted with the original search space of policies, which is described in Section 3.2 and is also the subject of the experiments of Sections 4.1 and 4.2. In that search space, the possible values with which to instantiate the mode of operation are those of table 3.2, with *equal probability of selection* for each degree of freedom when instantiating leaf nodes of random trees.

Figure 4.3 shows how varying the learning rate affects the weekly performance observed after 1800 steps. The plots reveal that, with the exception of a smaller median and first quartile for $lr = 0.1$, there are not many noticeable differences between the quality of the solutions obtained with different values for the learning rate. For all cases in which $lr > 0.1$, the median weekly performance is approximately 8 and the box plots pretty much overlap. Recall that the learning rate is used to update the fitness value of the last policy tried on the system with the last daily performance observed, as in Equation 3.11 (where the learning rate is denoted by Ω). A plausible reason why changing the learning rate value did not affect the quality of the evolved policies very much is that the daily performance is approximately the same when the same policy is tested more than once on the system. In other words, the daily performance with a certain policy measured at any given time seems to be a good predictor of future performance values when applying the same policy. We would argue that this is very much dependent on the problem domain and the system model.

Figure 4.4 shows the impact on the weekly performance of varying the initial temperature value. In this case, it seems clear that starting the procedure with a greater temperature value leads to solutions of generally better quality, as the median value for the experiments with a higher initial temperature is greater. Recall that the temperature parameter affects the probabilistic choice of the next policy to be tried on the system; the fitness values are divided by the temperature parameter when calculating Softmax probabilities, as in Equation 3.12, which means that, the higher the temperature value, *the more uniform* the resulting probability distribution. A higher initial temperature value will therefore lead to a higher degree of exploration of policies, decreasing the

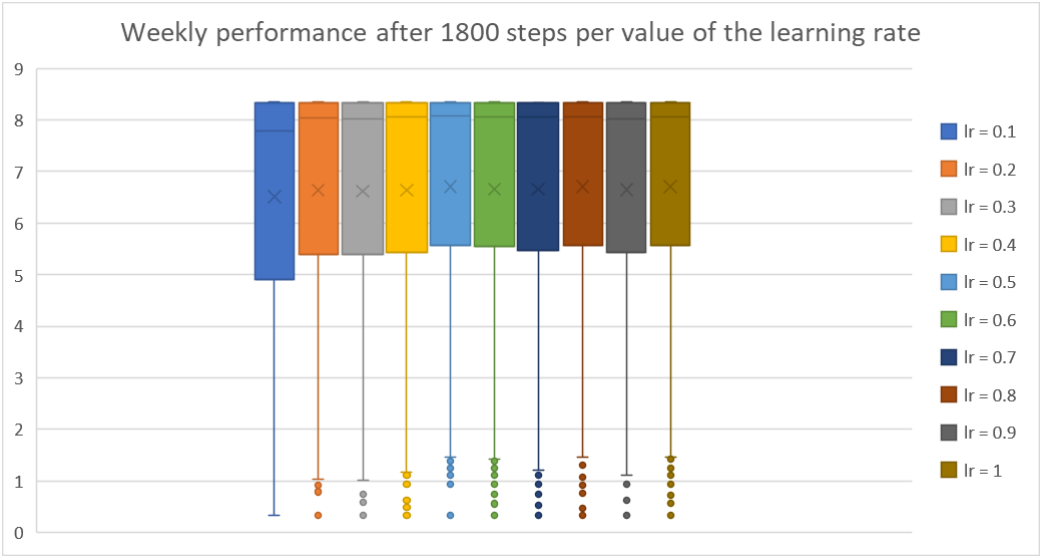

Figure 4.3: Box plots of the weekly performance value after 1800 steps for different values of the learning rate

risk of premature convergence to a population of good but sub-optimal policies; this is why higher initial temperature values have led to better solutions. Note, however, that *a higher initial temperature causes the convergence to be slower*. A higher degree of exploration takes place while the temperature is greater than 1; this parameter is divided by 2 every 168 time steps (corresponding to a week), so table 4.7 shows the number of simulation weeks during which the temperature is greater than 1 for each of the tested initial temperature values. When the temperature reaches 1, a new phase of *exploitation* of the current population of policies begins; however, policies are still selected probabilistically based on their fitness values - so a smaller degree of exploration is kept - and new generations of policies are still evolved over time as before.

Figure 4.5 shows how varying the percentage of individuals copied on to the following generation, *copy_perc*, impacts the weekly performance value after 1800 time steps. It is clear that smaller values of *copy_perc* lead to better solutions, as this implies that there will be greater variability in the following generation with more new policies being added, decreasing the risk of premature convergence to a population of sub-optimal policies. However, as exploration decreases and exploitation increases, it is crucial that at least some of the best policies found so far are kept in the following generation for the system to be able to converge to a set of approximately optimal policies. In this set of experiments, we have employed elitist strategies when copying policies to

Initial temperature	**Number of weeks**
40	6 (1008 time steps)
180	7 (1176 time steps)
320	9 (1512 time steps)

Table 4.7: Number of simulation weeks during which $temperature > 1$ for different initial temperature values

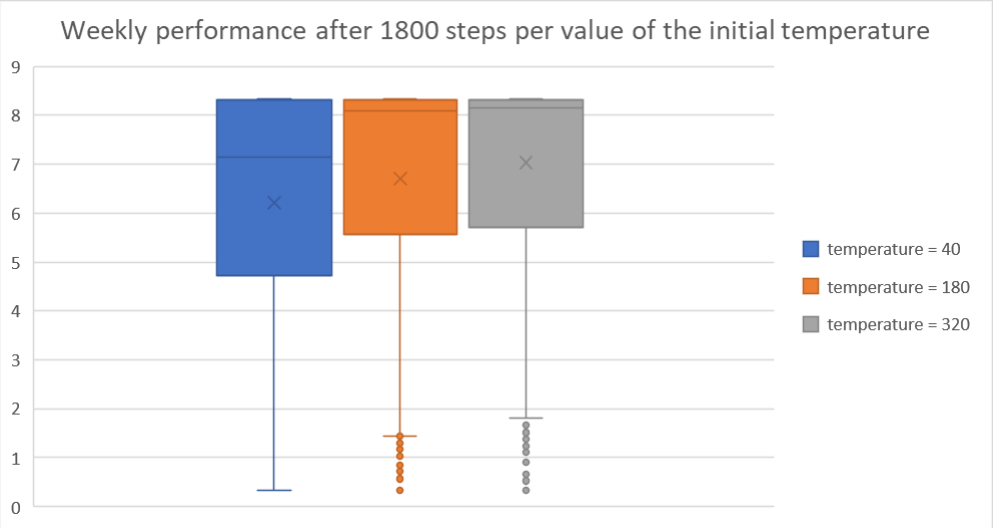

Figure 4.4: Box plots of the weekly performance value after 1800 steps for different values of the initial temperature

the following generation.

Figure 4.6 shows how the distribution of weekly performance values varies for different values of the percentage of individuals resulting from crossover when a new generation is evolved, *cross_perc*. While the box plots look very similar for values less than or equal to 0.6, there is a notable difference for $cross\_perc = 0.8$, with a smaller median value and first quartile. When $cross\_perc = 0.8$, this means that $copy\_perc = 0.2$, $mut\_perc = 0$, and no randomly initialised policies are added to the new generation. Since the crossover operation will likely lead to convergence, these results suggest that it is desirable to promote variability among the individuals of new generations by adding policies resulting from either the mutation operation or a random initialisation. The quality of the solutions appears to be hindered when none of these operations are applied.

The box plot of Figure 4.7 shows the same for the percentage of individuals resulting from mutation when a new generation is evolved, *mut_perc*. Note that, when $p = copy\_perc + cross\_perc + mut\_perc < 1$, then new random policies are included in the following generation which do not result from any of the standard GP operations; the percentage of these policies is $1 - p$. The plots show that there are no noticeable variations in the quality of the solutions found for different values of *mut_perc*. The reason for this might be that it is relatively easy to find an optimal solution and that adding new policies which result from the mutation operation is equally effective at introducing variability as adding new random policies. As discussed in Section 4.1, there are several local optima, yielding solutions of the same quality. If these local optima are structurally (shape of the trees) and functionally (sequences of modes of operation) diverse, then it makes sense that including new randomly initialised policies in each generation is as appropriate for eventually finding at least one local optimum as evolving new policies using GP. In order to find out whether some amount of new policies resulting from GP operations might be necessary for systems for

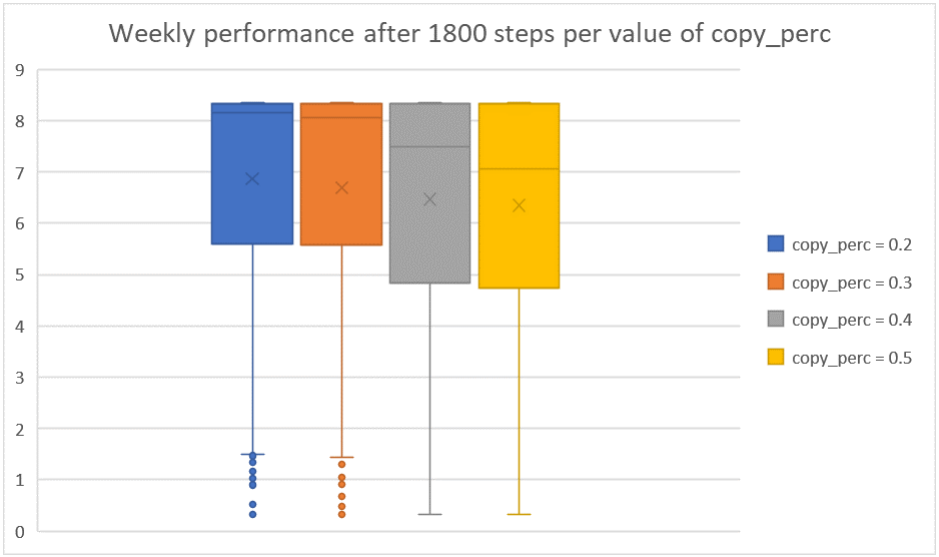

Figure 4.5: Box plots of the weekly performance value after 1800 steps for different values of *copy_perc*

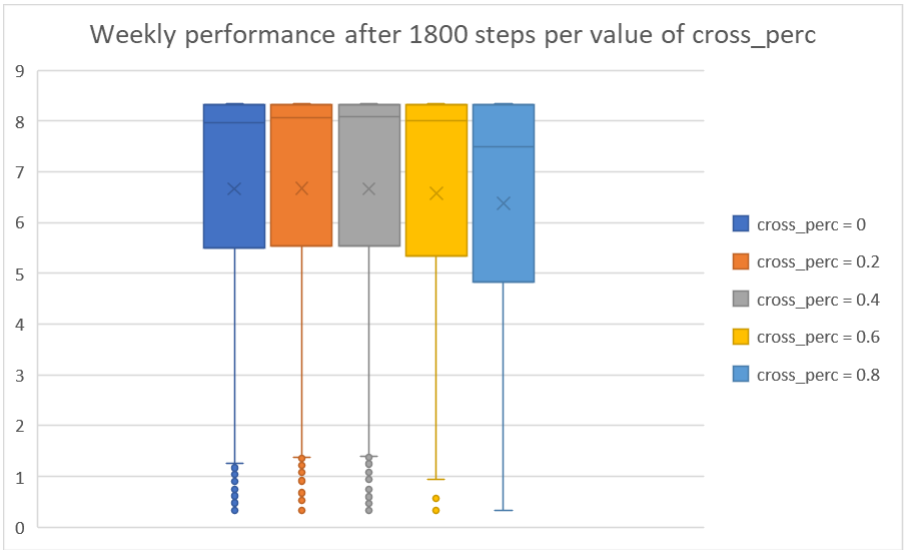

Figure 4.6: Box plots of the weekly performance value after 1800 steps for different values of *cross_perc*

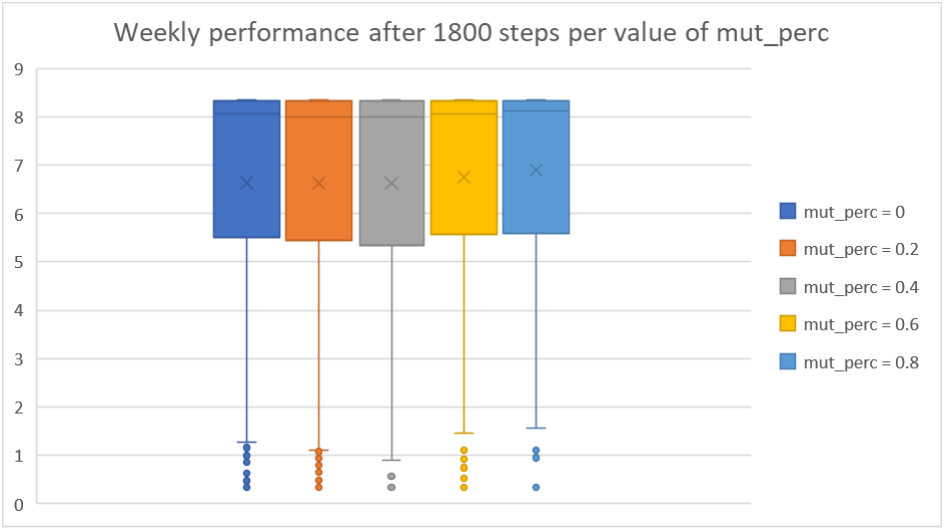

Figure 4.7: Box plots of the weekly performance value after 1800 steps for different values of *mut_perc*

which an optimal policy is harder to find, we have deliberately modified the search space in order to make it harder for the online procedure to find the local optima which it has found in this set of experiments. Section 4.3.2 explains this modification and discusses the new results.

4.3.2 Experiments with the Modified Search Space

As mentioned in Section 4.1, the optimal solutions are policies which, most of the time, select the value SELF_SUPPLY for the first degree of freedom and TRADE + SELL for the second degree of freedom. When initialising random trees or replacing parts of a tree with a random sub-tree as part of the mutation operation, creating a leaf node requires selecting one of the possible values for the degree of freedom corresponding to the tree in question. The possible values for each degree of freedom originally have equal probability of selection. We have modified the search space so that the probabilities of selecting the values SELF_SUPPLY and TRADE + SELL are lower than the probabilities of selecting other values for the respective degree of freedom. This makes the optimal solutions reported earlier harder to find and, therefore, has enabled us to study more deeply the influence of GP operations and the parameters of the online procedure on the quality of solutions when an optimal policy is not so easily found. Empirical observations have enabled us to conclude that, when the value TRADE + SELL is not selected for the second degree of freedom, the other values which involve trading between communities, TRADE + STORE + SELL and STORE + TRADE + SELL, are usually selected, with the former leading to better solutions. Therefore, we have also made these two values less likely to be selected. Tables 4.8, 4.9, and 4.10 show the probabilities of selecting the possible values for each degree of freedom at a leaf node in this modified search space.

Figure 4.8 shows the box plots of the weekly performance for different values of the learning rate. There is a more noticeable difference between the box plots than what has been observed

Value	Probability of selection
SELL	6/7
SELF_SUPPLY	1/7

Table 4.8: Probability of selecting the possible values for the first degree of freedom (concerning the action upon the energy produced by each community) at a leaf node in the modified search space

Value	Probability of selection
SELL	4/13
STORE + SELL	4/13
TRADE + SELL	1/13
STORE + TRADE + SELL	3/13
TRADE + STORE + SELL	1/13

Table 4.9: Probability of selecting the possible values for the second degree of freedom (concerning the action upon the excess of energy produced by each community in case of self-supply) at a leaf node in the modified search space

Value	Probability of selection
GREATEST_DEMAND	1/5
GREATEST_PRODUCTION	1/5
LOWEST_SATISFACTION	1/5
RANDOM	1/5
RATION	1/5

Table 4.10: Probability of selecting the possible values for the third degree of freedom (concerning how the energy is redistributed by the central system) at a leaf node in the modified search space

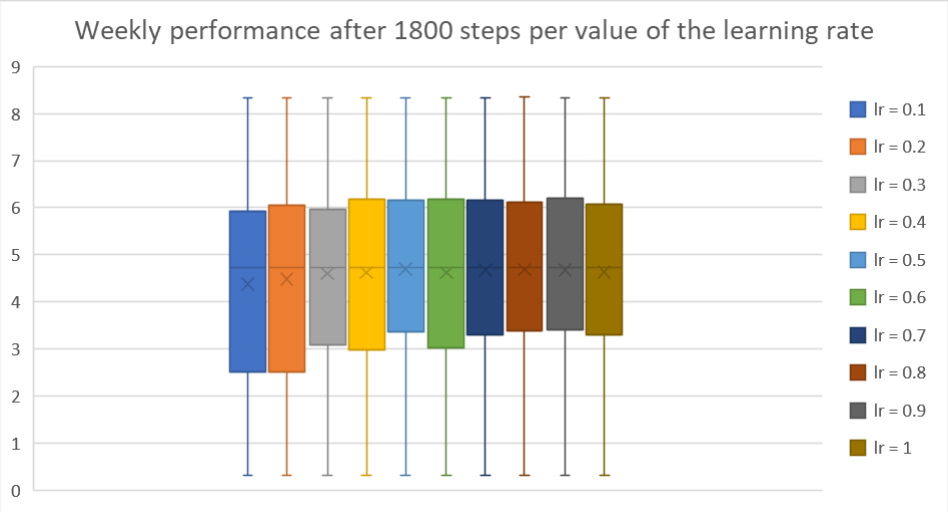

Figure 4.8: Box plots of the weekly performance value after 1800 steps for different values of the learning rate (modified search space)

in the experiments of Section 4.3.1. While the median value remains approximately constant for different learning rates, the distributions vary, with different first and third quartiles. The solutions are generally better for greater values of the learning rate. The reason for this could be that the procedure underestimates the better policies when they are first tried, with smaller learning rate values assigning greater weight to these poor estimations as the fitness values of the policies are updated over time with Equation 3.11. Finding good policies is harder in this search space, so it is expected that a greater number of bad policies will be tried on the system over time compared to when the procedure is searching the original space of policies. Since the performance of the system under a policy will affect future performance, we can expect good policies to be underestimated if they are tried shortly after bad policies. As the procedure evolves populations of increasingly better policies, greater values for the learning rate are likely to lead to more accurate estimates for the fitness of these policies, as they take more recent performance values into account.

Figure 4.9 shows how the value of the initial temperature affects the weekly performance observed after 1800 steps. As before, the box plots show that the higher of the three values for the initial temperature leads to generally better solutions. The difference between the distributions seems greater in this case than in the results of Section 4.3.1, with the procedure clearly performing worse when the initial temperature is 40. As we have deliberately made any optimal solution harder to find, initial exploration of policies has even greater importance than in the previous set of experiments, which explains the stark difference between the quality of the solutions found when the initial temperature is 40 and when it is 180 or 320.

Figure 4.10 shows how the percentage of individuals copied to the following generation, *copy_perc*, affects the weekly performance. As before, we observe that the quality of the solutions is generally worse for higher values of *copy_perc*. As discussed in Section 4.3.1, it is necessary to preserve some of the best solutions found so far in the following generations, but sufficient variability must also be introduced.

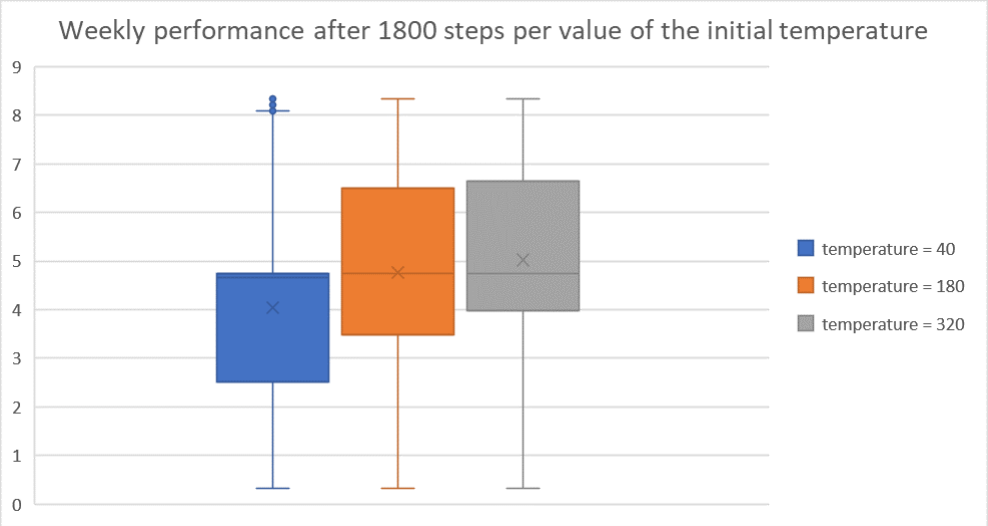

Figure 4.9: Box plots of the weekly performance value after 1800 steps for different values of the initial temperature (modified search space)

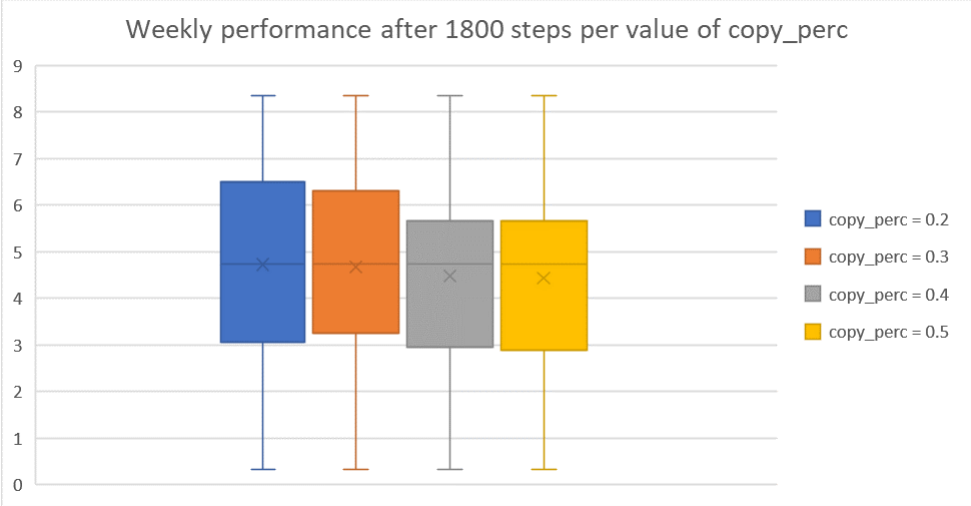

Figure 4.10: Box plots of the weekly performance value after 1800 steps for different values of *copy_perc* (modified search space)

Figure 4.11: Box plots of the weekly performance value after 1800 steps for different values of *cross_perc* (modified search space)

Figure 4.11 shows the influence of the percentage of individuals in new generations resulting from the crossover operation, *cross_perc*, on the quality of the solutions. This parameter seems to have a greater impact here than in the set of experiments of Section 4.3.1, with the weekly performance clearly degrading as *cross_perc* grows. These results suggest very strongly that *cross_perc* should be set to 0, meaning that the crossover operation is not useful in this scenario. Crossover tends to promote convergence of the individuals in a population and, therefore, to hinder variability. In this set of experiments, it appears to be very important to introduce enough variability when evolving a new generation and that this involves not performing the crossover operation.

Figure 4.12 shows the same results for the percentage of individuals in new generations resulting from the mutation operation, *mut_perc*. The image suggests that a higher mutation percentage will generally lead to better solutions, with the third quartile increasing as *mut_perc* grows. However, the first quartile decreases at some point. The previous results show that the quality of the solutions is more noticeably hindered when the initial temperature is 40, $cross\_perc > 0$, or $copy\_perc > 0.3$. In order to gain a better understanding of the influence of the *mut_perc* parameter on the quality of the solutions, we have filtered out those cases, keeping only the experiments for which $initial\_temperature \geq 180$, $copy\_perc \leq 0.3$, and $cross\_perc = 0$, and drawn new box plots for each value of *mut_perc*, shown in Figure 4.13. These show that the median value is greater for $mut\_perc = 0.2$ and $mut\_perc = 0.4$ and that the quality of the solutions plummets when $mut\_perc = 0.8$. Recall that, when $copy\_perc + cross\_perc + mut\_perc < 1$, new random policies are added to the following generation. These observations suggest that, while promoting variability when evolving new generations is important, it should be the result of *both* the mutation operation and new randomly initialised policies. Adding new random policies to each generation is especially important in the online procedure we propose as the population size should not be too large; since policies are selected to be tried on the system every 24 time steps and a new genera-

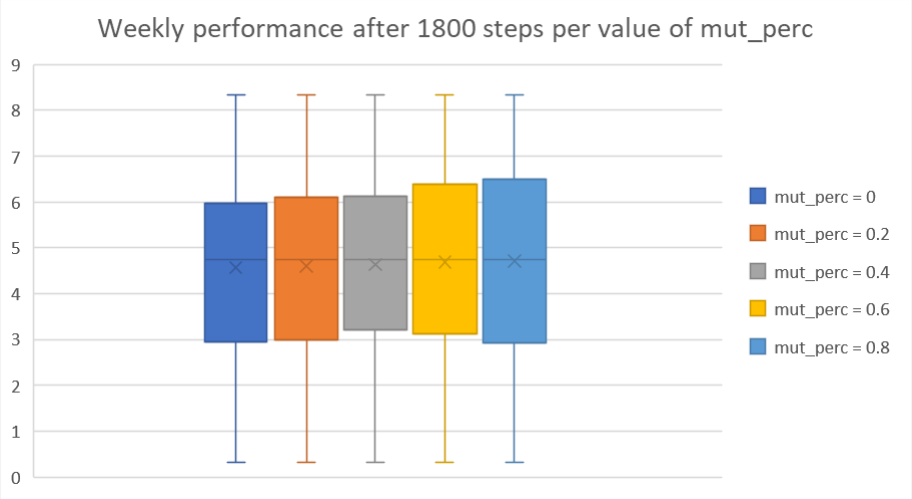

Figure 4.12: Box plots of the weekly performance value after 1800 steps for different values of *mut_perc* (modified search space)

tion is evolved every 336 time steps (two weeks' time in the simulation), only a maximum of 14 policies may be evaluated before evolving a new generation. Therefore, unlike other scenarios in which GP is applied, it is not advisable to start the online procedure with a very large population, as the policies which are not evaluated are assigned a default fitness value of 0 and this may be an overestimation with a potentially negative effect on the composition of following generations. The initial population size in all of our experiments is 12. Adding new random policies to each generation seems to be a reasonable way of introducing variability to cover more of the search space while keeping the population size small.

As an attempt to further validate some of the observations made in this section, we have additionally carried out several significance tests. For that purpose, we have selected a base parameter setting, shown in table 4.11, and performed the Mann-Whitney U test (two-tailed) on pairs of samples corresponding to variations of one of the parameters *while the others are kept fixed*. This statistical test checks whether values of one sample are significantly greater or less than the values of the other sample. Recall that we have collected the results of 30 runs for each parameter setting, so each sample has size 30. Table 4.12 summarises the significance tests which have been carried out and their results, namely p-values and whether they correspond to a *statistically significant* difference at the 0.05 significance level (the usual convention).

The first test compares samples obtained with $temperature = 320$ to samples obtained with $temperature = 40$. Previous results suggest that starting the online procedure with $temperature = 40$ leads to considerably worse results than starting it with $temperature = 320$. The test results support this hypothesis with a p-value of 2.503×10^{-5}, which means that the difference between the tested samples is significant at the 0.05 significance level. The reason for such a significant difference, with a very small p-value, has been discussed earlier in this section and concerns the fact that a higher temperature value promotes greater initial exploration. The second test compares samples obtained with $copy\_perc = 0.2$ to samples obtained with $copy\_perc = 0.4$. The p-value

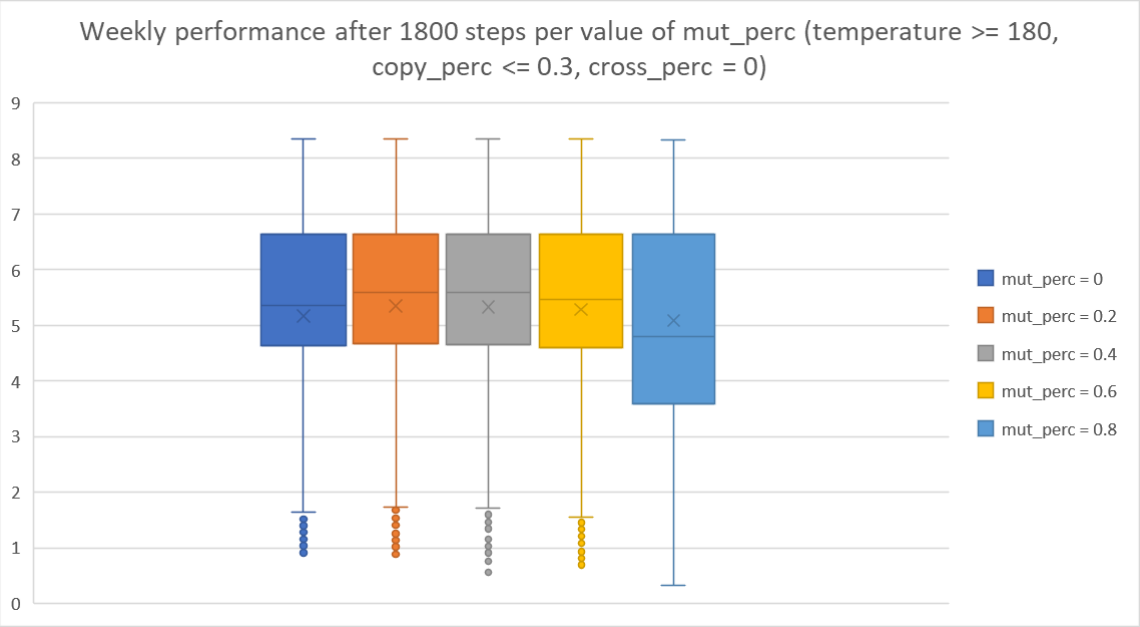

Figure 4.13: Box plots of the weekly performance value after 1800 steps for different values of *mut_perc* (modified search space)

is 1.978×10^{-2}, which means that the difference between the samples is significant at the 0.05 level. This supports our claim that the percentage of individuals copied to the following generation should be kept small, just enough for enabling an increased exploitation of the best policies found so far over time. If too many individuals are copied to subsequent generations, both variability and, consequently, the quality of the solutions will be hindered. The third test compares samples obtained with $cross\_perc = 0$ to samples obtained with $cross\_perc = 0.2$. Previous observations show that generally better solutions are the result of setting $cross\_perc = 0$. Indeed, with a p-value of 1.381×10^{-2}, the difference between the samples is significant at the 0.05 level. The final two tests concern the *mut_perc* parameter. The first of the two compares samples obtained with $mut\_perc = 0.2$ to samples obtained with $mut\_perc = 0$, while the latter compares samples obtained with $mut\_perc = 0.2$ to samples obtained with $mut\_perc = 0.8$. The p-values of 8.13×10^{-2} and 6.8×10^{-2}, respectively, mean that the difference in both cases is not significant at the 0.05 level. However, these p-values are still small enough to enable us to suggest that some amount of policies resulting from mutation could lead to better results than when adding no policies resulting from mutation and that it is advisable to add new random policies to each generation other than those resulting from mutation, i.e., that *mut_perc* should be greater than 0 and less than 0.8 in this case. Note that the p-values imply significance at the 0.05 level, being half of the values presented above, when performing one-tailed tests under the assumption that the values from the sample obtained with $mut\_perc = 0.2$ are greater than the values from the samples obtained with $mut\_perc = 0$ and $mut\_perc = 0.8$. However, one-tailed tests neglect the possibility of the values from the sample corresponding to $mut\_perc = 0.2$ being less than the values from the other samples, which should not be ignored in this case.

lr	0.9
temperature	320
copy_perc	0.2
cross_perc	0
mut_perc	0.2

Table 4.11: Base parameter setting for the significance tests

4.4 Summary

This section summarises the main observations resulting from the experiments which have been conducted. The results of the experiments of Section 4.1 have revealed that the offline optimisation procedure is robust to the stochastic nature of GP optimisation, with little variation between the quality of the solutions found in different runs. We have also observed that, while different solutions (policies) produce different sequences of modes of operation which lead to approximately the same performance, there is a certain degree of similarity between these sequences; this means that, while many different policies may result in a similar approximately optimal performance, their *consequences*, in terms of system operation, do not vary much. The results of the experiments of Section 4.2 have shown not only that the online procedure is capable of improving system performance over time by promoting the presence of increasingly better policies in new generations, but also that it is usually capable of converging to the same performance as the one obtained by running the offline procedure, all the while being a more useful method which does not assume the existence of a system model upon which policies may be tested beforehand. The main observations which have emerged from the detailed discussion of Section 4.3 are that the value of the initial temperature should be sufficiently large for the online procedure to be able to converge to better policies; the percentage of individuals copied on to the following generation should be just enough to guarantee that the system is able to increasingly exploit the better policies, but not too much that it hinders variability; the percentage of individuals resulting from crossover should be small or 0, as crossover may not promote enough variability, leading to convergence to sub-optimal policies; and that variability should be the result of both the mutation operation and random policies added to each new generation, with the latter compensating for a population size which should be smaller than the usual for GP optimisation, since no more than 14 policies may be evaluated by the online procedure before evolving a new generation.

Tested parameter	**Value range**	**p-value**	**Significant at the 0.05 level**
temperature	{320, 40}	2.503×10^{-5}	Yes
copy_perc	{0.2, 0.4}	1.978×10^{-2}	Yes
cross_perc	{0, 0.2}	1.381×10^{-2}	Yes
mut_perc	{0.2, 0}	8.13×10^{-2}	No
mut_perc	{0.2, 0.8}	6.8×10^{-2}	No

Table 4.12: Results of the Mann-Whitney *U* tests (two-tailed)

Chapter 5

Conclusions

In this chapter, we present the main conclusions which have emerged from this project. We start by summarising the work we have carried out and its results in Section 5.1. We point out some limitations of our approach in Section 5.2. In Section 5.3, we discuss possible directions for future research which may follow as a consequence of this work concerning the adaptation and evolution of system policies. At last, in Section 5.4, we comment on the significance of our contributions.

5.1 Main Contributions

This report describes in detail our research into how adaptation through evolution of policies can assist the design of collective adaptive systems which remain sustainable over time in the face of dynamic environments that may change unpredictably. The problem we have addressed has been to find operating policies which are approximately optimal for a system, given some performance criterion. We have modelled an energy system encompassing several integrated CESs where each community is an agent and energy is treated as a CPR. We have proposed mechanisms which enable this system model to optimise its performance over time through adaptation and evolution of its operating policy, which determines the mode of operation at each time step. The offline procedure assumes the existence of a system model upon which alternative policies may be tested beforehand and finds a policy which optimises performance during a single simulation week, following a standard GP approach. The online procedure makes no such assumption and is intended to be applied to systems which are continuously running; it is a hybrid approach which draws inspiration from GP and RL to evolve policies over time, using past performance history to promote the presence of increasingly better policies in each new generation. The results show that these optimisation procedures are useful and could lead to a better understanding of mechanisms which enable a system to remain sustainable over time. The policies evolved by our procedures *clearly outperform* the policy we have initially designed ourselves.

The representation of system policies has been a key issue throughout the modelling of the system. Representing the policies with binary decision trees has enabled us to apply GP operations when generating and evolving them. This representation is also appropriate for *drawing*

explanations about the output of our optimisation procedures. The trees can easily be translated into a sequence of potentially nested `if-then-else` rules, which may help human designers to gain insight about the system operation and what makes a good policy, enabling them to construct better policies themselves or to provide more useful "building blocks" for the procedures to find policies automatically.

5.2 Limitations

There are some limitations about our work which should be kept in mind. Firstly, the model we have created and upon which we have tested our procedures is simple. Its only goal has been to provide an example case for representing, applying, and evolving policies. The model assumes full compliance from the agents, i.e., enforcing a policy effectively sets constraints upon the possible actions which the agents may execute, as opposed to schemes which do not restrict actions and instead punish non-compliant behaviour. As a result of this, the agents lack autonomy and they also do not have cognitive abilities enabling them to learn or make complex decisions. At the system level, many aspects of political and economic science regarding self-organising EIs have not been included, such as knowledge management or notions of justice. The methods we propose have only been tested and validated against this system model and may require tuning of their parameters or other modifications when applied to more complex models and/or to other problem domains. In particular, the performance of the optimisation methods depends very much on how easy or difficult it is to find optimal (or at least good) policies and this could vary greatly among different problem domains and system models.

5.3 Future Work

We believe there are opportunities for improving and further validating the approaches we have proposed. In future work, we would like to look into increasing the complexity of the energy system model which we have created in this project. This includes assuming that the system is *open* to heterogeneous and autonomous agents. In this case, the agents may not always comply with the rules, as their actions are not constrained, which poses a need for mechanisms for detecting and punishing violations and other undesired behaviour. We also seek to incorporate an active participation of the agents in the process of adapting and evolving policies, e.g. taking into account their preferences and opinions when evaluating alternative policies, as a means to achieve some degree of self-organisation. Institutional and social concepts such as distributive justice and knowledge management may also be part of our study.

Regarding the optimisation procedures we have proposed, we would also like to apply them to other problem domains as further test cases. This would enable us to gain a better understanding of the parts of the procedures which are domain-dependent and those which are domain-independent. We could also explore other evolutionary approaches besides GP and possibly other representations for policies besides binary trees. Another important question to look into is the action that

should be taken by a system when it is faced with changes so drastic that the current policy is no longer congruent with the state of the environment. In this case, it seems straightforward to restart the online procedure from that point on so that the system is able to converge to a population of suitable policies, but there might be other options worth exploring.

5.4 Final Remarks

The methods we have presented return policies which are appropriate for a system, given some performance criterion, without a human designer's intervention. The contributions of this work are highly significant, since our proposal, for which we presented a proof of concept, could lay the foundations for the development of a new methodological paradigm for the engineering of collective adaptive systems. Our approach could be used to assist system designers, so far required to rely mostly on their own intuition, in *systematically* finding good policies, which could generally lead to better performance and provide support for adaptation mechanisms in the face of non-deterministic changes in dynamic environments. As a result of this project, we have submitted two papers which have been accepted, one at the ISoLA 2018 conference and the other at the eCAS workshop, part of the SASO 2018 conference.

References

[ADGC+07] Huib Aldewereld, Frank Dignum, Andrés García-Camino, Pablo Noriega, Juan Antonio Rodríguez-Aguilar, and Carles Sierra. Operationalisation of norms for electronic institutions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4386 LNAI, pages 163–176, 2007.

[ARAR08] Josep Lluís Arcos, Juan A. Rodríguez-Aguilar, and Bruno Rosell. Engineering autonomic electronic institutions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5049 LNAI, pages 76–87, 2008.

[Art12] Alexander Artikis. Dynamic specification of open agent systems. *Journal of Logic and Computation*, 22(6):1301–1334, 2012.

[ARVRRO15] Rubén Aguilar-Rivera, Manuel Valenzuela-Rendón, and J. J. Rodríguez-Ortiz. Genetic algorithms and Darwinian approaches in financial applications: A survey, 2015.

[AZPX14] Soha Ahmed, Mengjie Zhang, Lifeng Peng, and Bing Xue. Multiple feature construction for effective biomarker identification and classification using genetic programming. In *GECCO '14: Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 249–256, 2014.

[BBBMM14] Thomas Bartz-Beielstein, Jürgen Branke, Jörn Mehnen, and Olaf Mersmann. Evolutionary Algorithms, 2014.

[BBSS05] A. Bogdanovych, H. Berger, S. Simoff, and C. Sierra. Narrowing the gap between humans and agents in e-commerce: 3d electronic institutions. pages 128 – 37, Berlin, Germany, 2005//. e-commerce;3D electronic institutions;autonomous software agents;agent trust;3D virtual world;immersive user interface;agent behavior;.

[BD18] Scott Bucking and Vasken Dermardiros. Distributed evolutionary algorithm for co-optimization of building and district systems for early community energy masterplanning. *Applied Soft Computing*, 63(Supplement C):14–22, 2018.

[BHKW12] E.K. Burke, M. Hyde, G. Kendall, and J. Woodward. A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics. *IEEE Transactions on Evolutionary Computation*, 14(6):942 – 58, 2010/12/. genetic programming hyper heuristic approach;evolving 2D strip packing heuristics;search methodologies;evolutionary computation;.

REFERENCES

[BL04] M D Buhmann and Jeremy Levesley. Radial Basis Functions: Theory and Implementations. *Mathematics of Computation*, 73(247):1578–1581, 2004.

[BP14] Aikaterini Bourazeri and Jeremy Pitt. An agent-based serious game for decentralised community energy systems. *Lecture Notes in Computer Science*, 8861:246–253, 2014.

[Bra96] Robert B. Brandom. Making it explicit: Reasoning, representing, and discursive commitment. *Philosophical Quarterly*, 46(183):238–241, 1996.

[Bra00] Robert Brandom. *Articulating Reasons. An Introduction to Inferentialism*. 2000.

[Bra02] R Brandom. *Tales of the Mighty Dead: Historical Essays in the Metaphysics of Intentionality*. Harvard University Press, 2002.

[CAA+] Minjiang Chen, D Athanasiadis, B Al Faiya, S McArthur, I Kockar, Haowei Lu, and F de Leon. Design of a multi-agent system for distributed voltage regulation. In *2017 19th International Conference on Intelligent System Applications to Power Systems (ISAP)*, pages 6 pp. –, Piscataway, NJ, USA.

[CHC14a] Nicola Capodieci, Emma Hart, and Giacomo Cabri. Artificial Immune System Driven Evolution in Swarm Chemistry. In *International Conference on Self-Adaptive and Self-Organizing Systems, SASO*, volume 2014-December, pages 40–49, 2014.

[CHC14b] Nicola Capodieci, Emma Hart, and Giacomo Cabri. Designing self-aware adaptive systems: From Autonomic computing to cognitive immune Networks. In *Proceedings - IEEE 7th International Conference on Self-Adaptation and Self-Organizing Systems Workshops, SASOW 2013*, pages 59–64, 2014.

[CHC16] Nicola Capodieci, Emma Hart, and Giacomo Cabri. Artificial Immunology for Collective Adaptive Systems Design and Implementation. *ACM Transactions on Autonomous and Adaptive Systems*, 11(2):1–25, 2016.

[CMKP15] Minjiang Chen, Stephen D.J. McArthur, Ivana Kockar, and Jeremy Pitt. Evaluating a MAS architecture for flexible distribution power flow management. In *2015 18th International Conference on Intelligent System Application to Power Systems, ISAP 2015*, 2015.

[CO04] HL Cardoso and E Oliveira. Virtual enterprise normative framework within electronic institutions. *Proceedings of the 5th International Workshop on Engineering Societies in the Agents World (ESAW 04)*, pages 14–32, 2004.

[CO08] Henrique Lopes Cardoso and Eugénio Oliveira. A contract model for electronic institutions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4870 LNAI, pages 27–40, 2008.

[Col88] J S Coleman. Social Capital in the Creation of Human-Capital. *American Journal of Sociology*, 94:S95–S120, 1988.

[DGK05] Giovanna Di Marzo Serugendo, Marie Pierre Gleizes, and Anthony Karageorgos. Self-organization in multi-agent systems, 2005.

[Dig06] F. Dignum. Norms and electronic institutions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4048 LNAI, pages 2–5, 2006.

[dJRAGS15] Dave de Jonge, Juan A. Rodriguez-Aguilar, Bruno Rosell i. Gui, and Carles Sierra. Infrastructures to engineer open agent environments by means of electronic institutions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9068, pages 232–254, 2015.

[ERASV04] M. Esteva, J.A. Rodriguez-Aguilar, C. Sierra, and W. Vasconcelos. Verifying norm consistency in electronic institutions. volume WS-04-02, pages 8 – 14, San Jose, CA, United states, 2004. Computational approach;Electronic institutions;Illocutions;.

[EVSRA04] M. Esteva, W. Vasconcelos, C. Sierra, and J.A. Rodriguez-Aguilar. Norm consistency in electronic institutions. pages 494 – 505, Berlin, Germany, 2004//. verification;electronic institutions;norm consistency;protocols;individual goals;global goals;integrity norms;obligations;dialogues;multiagent systems;.

[FCSB14] Helenice O. Florentino, Daniela R. Cantane, Fernando L P Santos, and Bettina F. Bannwart. Multiobjective Genetic Algorithm applied to dengue control, 2014.

[For61] Jay W. Forrester. *Industrial dynamics*. 1961.

[For71] J. W. Forrester. Counterintuitive Behavior of Social Systems. *Simulation*, 16:61–76, 1971.

[FPS03] Gianluigi Folino, Clara Pizzuti, and Giandomenico Spezzano. Ensemble techniques for Parallel Genetic Programming based Classifiers. *Genetic Programming, Proceedings of EuroGP'2003*, 2610:59–69, 2003.

[Gäc07] Simon Gächter. Conditional cooperation : Behavioral regularities from the lab and the field and their policy implications about the Centre or contact. *Economics and Psychology. A Promising New Cross-Disciplinary Field*, April 2006(2006-3):19–50, 2007.

[GAVSD06] Davide Grossi, Huib Aldewereld, Javier Vázquez-Salceda, and Frank Dignum. Ontological aspects of the implementation of norms in agent-based electronic institutions. *Computational and Mathematical Organization Theory*, 12(2-3 SPEC. ISS.):251–275, 2006.

[GAZ09] Aytac Guven, H.Md. Azamathulla, and N.A. Zakaria. Linear genetic programming for prediction of circular pile scour. *Ocean Engineering*, 36(12-13):985 – 991, 2009. Adaptive neuro fuzzy inference systems (ANFIS);Circular piles;Field measurement;Linear genetic programming;Model results;Neuro-Fuzzy;Regression;Statistical errors;.

[GCRASV09] Andrés García-Camino, Juan A. Rodríguez-Aguilar, Carles Sierra, and Wamberto Vasconcelos. Constraint rule-based programming of norms for electronic institutions. *Autonomous Agents and Multi-Agent Systems*, 18(1):186–217, 2009.

[GDC+17] Julian Pedro Garbiso, Ada Diaconescu, Marceau Coupechoux, Jeremy Pitt, and Bertrand Leroy. Distributive justice for fair auto-adaptive clusters of connected vehicles. In *Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017*, pages 79–84, 2017.

[GSFB10] Markus Gärtner, Ingo Seidel, Josef Froschauer, and Helmut Berger. The formation of virtual organizations by means of electronic institutions in a 3D e-Tourism environment. *Information Sciences*, 180(17):3157–3169, 2010.

[HLZ+15] Yong Hu, Kang Liu, Xiangzhou Zhang, Lijun Su, E. W.T. Ngai, and Mei Liu. Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review, 2015.

[HS16] Emma Hart and Kevin Sim. A Hyper-Heuristic Ensemble Method for Static Job-shop Scheduling. *Evolutionary computation*, 24(4):609–635, 2016.

[HSGK17] Emma Hart, Kevin Sim, Barry Gardiner, and Kana Kamimura. A Hybrid Method for Feature Construction and Selection to Improve Wind-damage Prediction in the Forestry Sector. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 1121–1128, New York, NY, USA, 2017. ACM.

[HY14] Zishuo Huang and Hang Yu. Approach for integrated optimization of community heating system at urban detailed planning stage. *Energy and Buildings*, 77:103–111, 2014.

[HYCP17] Zishuo Huang, Hang Yu, Xiangyang Chu, and Zhenwei Peng. A goal programming based model system for community energy plan. *Energy*, 134:893–901, 2017.

[HYPF17] Zishuo Huang, Hang Yu, Zhenwei Peng, and Yifu Feng. Planning community energy system in the industry 4.0 era: Achievements, challenges and a potential solution, 2017.

[HYPL15] Zishuo Huang, Hang Yu, Zhenwei Peng, and Zhiyuan Liu. Two-stage Optimization Model Used for Community Energy Planning. In *Energy Procedia*, volume 75, pages 2916–2921, 2015.

[HYPZ15] Zishuo Huang, Hang Yu, Zhenwei Peng, and Mei Zhao. Methods and tools for community energy planning: A review, 2015.

[IKS03] I. K. Ibrahim, G. Kotsis, and W. Schwinger. Mapping abstractions of norms in electronic institutions. In *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, volume 2003-January, pages 30–35, 2003.

[KKF+16] Binod Prasad Koirala, Elta Koliou, Jonas Friege, Rudi A. Hakvoort, and Paulien M. Herder. Energetic communities for community energy: A review of key issues and trends shaping integrated community energy systems, 2016.

[Koz92] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[KP17a] David Burth Kurka and Jeremy Pitt. Smart-CPR: Self-organisation and self-governance in the sharing economy. In *Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017*, pages 85–90, 2017.

[KP17b] David Burth Kurka and Jeremy Pitt. The principled violation of policy: Norm flexibilization in open self-organising systems. In *Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017*, pages 33–38, 2017.

[KPR+16] Hirushie Karunathilake, Piyaruwan Perera, Rajeev Ruparathna, Kasun Hewage, and Rehan Sadiq. Renewable energy integration into community energy systems: A case study of new urban residential development, 2016.

[Kra02] Krzysztof Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4):329–343, 2002.

[KS86] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.

[LB05] Yingqiang Lin and Bir Bhanu. Evolutionary feature synthesis for object recognition. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 35(2):156–171, 2005.

[LZH+17] Q.G. Lin, M.Y. Zhai, G.H. Huang, X.Z. Wang, L.F. Zhong, and J.W. Pi. Adaptation planning of community energy systems to climatic change over Canada. *Journal of Cleaner Production*, 143:686–698, 2017.

[MG17] Goran Mauša and Tihana Galinac Grbac. Co-evolutionary multi-population genetic programming for classification in software defect prediction: An empirical case study. *Applied Soft Computing Journal*, 55:331–351, 2017.

[MIF11] Gonçalo Mendes, Christos Ioakimidis, and Paulo Ferrão. On the planning and analysis of Integrated Community Energy Systems: A review and survey of available tools, 2011.

[MM13] Julian F. Miller and Maktuba Mohid. Function optimization using cartesian genetic programming. pages 147 – 148, Amsterdam, Netherlands, 2013. Benchmark functions;Cartesian genetic programming;Fitness functions;Function Optimization;Multi-modal;Real number;.

[MPSB12] Sam MacBeth, Jeremy Pitt, Julia Schaumeier, and Didac Busquets. Animation of self-organising resource allocation using Presage2. In *International Conference on Self-Adaptive and Self-Organizing Systems, SASO*, pages 225–226, 2012.

[NZA12] K Neshatian, Mengjie Zhang, and P Andreae. A Filter Approach to Multiple Feature Construction for Symbolic Learning Classifiers Using Genetic Programming. *Evolutionary Computation, IEEE Transactions on*, 16(5):645–661, 2012.

[OA03] Elinor Ostrom and Tk Ahn. Foundations of social capital. *Science*, pages 1–24, 2003.

[Ost15] Elinor Ostrom. *Governing the commons: The evolution of institutions for collective action*. 2015.

[PBM14] Jeremy Pitt, Dídac Busquets, and Sam Macbeth. Distributive Justice for Self-Organised Common-Pool Resource Management. *ACM Transactions on Autonomous and Adaptive Systems*, 9(3):1–39, 2014.

[PBR13] Jeremy Pitt, Dídac Busquets, and Régis Riveret. Procedural justice and 'fitness for purpose' of self-organising electronic institutions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8291 LNAI, pages 260–275, 2013.

[PBR15] Jeremy Pitt, Dídac Busquets, and Régis Riveret. The pursuit of computational justice in open systems. *AI and Society*, 30(3):359–378, 2015.

[PD16] Jeremy Pitt and Ada Diaconescu. Interactive self-governance & value-sensitive design for self-organising socio-Technical systems. In *Proceedings - IEEE 1st International Workshops on Foundations and Applications of Self-Systems, FAS-W 2016*, pages 30–35, 2016.

[PDB17] Jeremy Pitt, Ada Diaconescu, and Aikaterini Bourazeri. Democratisation of the SmartGrid and the active participation of prosumers. In *IEEE International Symposium on Industrial Electronics*, pages 1707–1714, 2017.

[PH17] Jeremy Pitt and Emma Hart. For flux sake: The confluence of socially-and biologically-inspired computing for engineering change in open systems. In *Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017*, pages 45–50, 2017.

[PHCG] B Prasad Koirala, R A Hakvoort, J P Chaves Avila, and T Gomez. Assessment of Integrated Community Energy Systems. In *2016 13th International Conference on the European Energy Market (EEM)*, pages 6 pp. –, Piscataway, NJ, USA.

[Pit17] Jeremy Pitt. Interactional Justice and Self-Governance of Open Self-Organising Systems. In *Proceedings - 11th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2017*, pages 31–40, 2017.

[POD] J Pitt, J Ober, and A Diaconescu. Knowledge Management Processes and Design Principles for Self-Governing Socio-Technical Systems. In *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W). Proceedings*, pages 97–102, Los Alamitos, CA, USA.

[PPB17] Patricio E. Petruzzi, Jeremy Pitt, and Dídac Busquets. Electronic Social Capital for Self-Organising Multi-Agent Systems. *ACM Transactions on Autonomous and Adaptive Systems*, 12(3):1–25, 2017.

[PSA12] Jeremy Pitt, Julia Schaumeier, and Alexander Artikis. Axiomatization of Socio-Economic Principles for Self-Organizing Institutions. *ACM Transactions on Autonomous and Adaptive Systems*, 7(4):1–39, 2012.

[RBB+02] Rosaldo J.F. Rossetti, Rafael H. Bordini, Ana L.C. Bazzan, Sergio Bampi, Ronghui Liu, and Dirck Van Vliet. Using BDI agents to improve driver modelling in a commuter scenario. *Transportation Research Part C: Emerging Technologies*, 10(5-6):373–398, 2002.

REFERENCES

[Res82] N Rescher. *Distributive Justice*. G - Reference,Information and Interdisciplinary Subjects Series. University Press of America, 1982.

[RLCB02] Rosaldo J F Rossetti, Ronghui Liu, Helena B B Cybis, and Sergio Bampi. A multi-agent demand model. *Proceedings of the 13th Mini-Euro Conference and The 9th Meeting of the Euro Working Group Transportation*, pages 193–198, 2002.

[SB05] Matthew G. Smith and Larry Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6(3):265–281, 2005.

[Sch03] M.G. Schuster. A multiobjective genetic programming approach for pricing and hedging derivative securities. pages 77 – 84, Piscataway, NJ, USA, 2003//. multiobjective genetic programming approach;finance;contingent claim pricing;symbolic differentiation;hedging;derivative securities;stock market;.

[Sea95] John R Searle. *Construction of Social Reality*, volume 1. 1995.

[SHP15] Kevin Sim, Emma Hart, and Ben Paechter. A Lifelong Learning Hyper-heuristic Method for Bin Packing. *Evolutionary Computation*, 23(1):37–67, 2015.

[Smi80] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.*, 29(12):1104–1113, December 1980.

[SPHGV] E Segredo, B Paechter, E Hart, and C I Gonzalez-Vila. Hybrid parameter control approach applied to a diversity-based multi-objective memetic algorithm for frequency assignment problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1517 – 24, Piscataway, NJ, USA.

[SRAN+04] Carles Sierra, Juan Antonio Rodriguez-Aguilar, Pablo Noriega, Marc Esteva, and Josep Lluis Arcos. Engineering Multi-Agent Systems as Electronic Institutions. *European Journal for the Informatics Professional*, 4(4):33–39, 2004.

[SRK+15] David Sanderson, Svetan Ratchev, Emma Kelly, Dídac Busquets, and Jeremy Pitt. Self-organising electronic institutions and flexible manufacturing systems. In *IFAC-PapersOnLine*, volume 28, pages 2071–2076, 2015.

[SVS+16] D Strickland, M A Varnosfederani, J Scott, P Quintela, A Duran, R Bravery, A Corliss, K Ashworth, and S Blois-Brooke. A review of community electrical energy systems. In *2016 IEEE International Conference on Renewable Energy Research and Applications (ICRERA)*, pages 49–54, nov 2016.

[TFLBP16] F. Torrent-Fontbona, B. López, D. Busquets, and J. Pitt. Self-organising energy demand allocation through canons of distributive justice in a microgrid. *Engineering Applications of Artificial Intelligence*, 52:108–118, 2016.

[TL10] Hsing-Chih Tsai and Yong-Huang Lin. Predicting high-strength concrete parameters using weighted genetic programming. *Engineering with Computers*, 27(4):347 – 55, 2011/10/. high strength concrete parameters prediction;weighted genetic programming;evolutionary algorithm;binary tree topology;WGP optimization;.

[VL14] Margarita Vázquez and Manuel Liz. Simulation models of complex social systems a constructivist and expressivist interpretation of system dynamics. In *Studies in Applied Philosophy, Epistemology and Rational Ethics*, volume 8, pages 563–582, 2014.

[XJJ+15] Xiandong Xu, Xiaolong Jin, Hongjie Jia, Xiaodan Yu, and Kang Li. Hierarchical management for integrated community energy systems. *Applied Energy*, 160:231–243, 2015.

[YZV17] Dayong Ye, Minjie Zhang, and Athanasios V. Vasilakos. A survey of self-organization mechanisms in multiagent systems, 2017.