Parisa Sadeghi

Balancing and Sequencing Mixed-Model Assembly Systems in the Footwear Industry

Submitted to Faculdade de Engenharia da Universidade do Porto in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Industrial Engineering and Management, Supervised by José António Soeiro Ferreira, Professor of Faculdade de Engenharia da Universidade do Porto

Department of Industrial Engineering and Management
Faculdade de Engenharia da Universidade do Porto
2018

This research was partially supported by the PhD grant SFRH/BD/91550/2012 awarded by the Portuguese Foundation for Science and Technology and by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project "POCI-01-0145-FEDER-006961"

"A person who never made a mistake never tried anything new."

Albert Einstein

Acknowledgments

By completing this thesis, a product of several years of work, I am deeply grateful to many great people who inspired and supported me markedly during my PhD study at the University of Porto.

In particular, I would like to thank my advisor, Professor José Soeiro, since all the work carried out until the present day was only possible with his strong motivation and generous help. It has been an honour to be her PhD student. His guidance helped me during the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD study. I am thankful for his constant feedback, patience and, above all, friendship.

Furthermore, Professor José Fernando Oliveira and Professor Maria Antónia Carravilla, by trusting and accepting me in the PhD program, gave me the opportunity to start a PhD in such a privileged university (FEUP) and at the same time to be in Portugal, which is a good country with nice people. I especially thank them for introducing me to my advisor.

Thanks and appreciation to INESC TEC for providing the required facilities without which I could not complete my research. I also deem it necessary to sincerely thank Engineer Rui Rebelo who strongly supported me during the hardest part of the project.

A very special thanks goes to my colleagues who did not hesitate to support me whenever I asked for assistance. I am so thankful to have Gonçalo Figueira beside me. He was always ready to help me from the beginning of my project. Thank you Gonçalo for all your supports, compassion and patience.

And finally, I would like to extend my sincerest thanks and appreciation to

my family for their unconditional support and assistance; to my parents who were a great help and without whom I could not reach this position, "Asheghetonam Maman o Baba"; to my kind and compassionate husband Hamed for his enthusiastic support, special attention, endless patience and encouragement; and to my beloved daughter Hana whose cheerful smile has been a source of strength, motivation and inspiration.

Parisa Sadeghi

Abstract

This thesis deals with Balancing and Sequencing Mixed-model Assembly Systems in the Footwear Industry. The scientific issues of balancing and sequencing problems are of great concern to Operational Research and quite relevant to many industries. This research work specially deals with the complex problems arising from new automatic transportation equipments in the stitching lines of the systems, of a large footwear industry. They will only be efficient and effective if they are well managed. A diversity of models move in the lines, in any direction, inside boxes. The operators are multi skilled and the machines are of many types.

The thesis intends to contribute scientifically in this area of optimisation but accompanying, at the same time, the real case studies to which it is associated. Therefore, the aim is also to reduce the gap between research and practice, by bringing more insights to the industry.

Two main general approaches are followed, to manage the assembly systems. First, the balancing problems and sequencing problems are considered separately and then, balancing and sequencing are addressed simultaneously. Optimisation models are provided, in connection with both approaches but, due to the dimension of the real problems, the final effective solution procedures were based on new approximate methods.

In what concerns Balancing, two methods were designed, ASBsm - inspired by Tabu Search, and RPW-VNDbal - which is based on the Variable Neighbourhood Descent metaheuristic and also using the Ranked Positional Weighted method. For Sequencing, two methods are proposed, VND-MSeq - involving Variable Neighbourhood Descent, and GA-MSeq - based on Genetic Algorithms. Different dispatching

rules are used to create initial solutions/populations. Finally, the integrated approach (Balancing and Sequencing simultaneously) pursues two directions: obtaining solutions by single and multi-objective methods. The methods developed are GA, with a single objective, and NSGA-II, a multi-objective method. The thesis is completed with the presentation and discussion of the computational results, based on generated problem instances and real data from the company. As a conclusion, it may be said that the methods are effective and that the results obtained are better that those available from the company.

Although the thesis is influenced by the mixed-model balancing and sequencing problems of a specific large company, it is believed that the proposed new methods can be applied directly to other footwear industries and extended, with adequate adaptations, to similar industrial sectors.

Resumo

A tese de doutoramento trata do Balanceamento e Sequenciamento de Sistemas de Montagem de Modelos Mistos na Indústria do Calçado. Os temas científicos associados aos problemas de balanceamento e sequenciamento são importantes no âmbito da Investigação Operacional e bastante relevantes para muitas empresas industriais. Este trabalho de investigação trata especialmente de problemas complexos que surgem com novos sistemas automáticos de transporte nas linhas de costura de uma grande empresa de calçado. Estes sistemas só serão eficientes e eficazes se forem bem geridos. Uma diversidade de modelos, dentro de caixas, move-se nas linhas, entre quaisquer postos de trabalho. Os operadores têm diferentes aptidões e as máquinas são de vários tipos.

A tese pretende contribuir cientificamente nesta área de otimização, uma meta fundamental. mas sempre acompanhando o caso de estudo empresarial a que está associada. Portanto, o objetivo é também reduzir o hiato entre a investigação e a verdadeira aplicação, transferindo mais valor para a indústria.

Destacam-se duas abordagens principais e gerais para gerir o desempenho dos sistemas de montagem. Primeiro, os problemas de balanceamento e os problemas de sequenciamento são tratados separadamente e, em seguida, o balanceamento e o sequenciamento são apreciados simultaneamente.

Formulam-se modelos de otimização matemática, em conexão com ambas as abordagens. No entanto, devido à dimensão dos problemas reais, os procedimentos finais efetivos de solução foram baseados em novos métodos aproximados. No que respeita ao Balanceamento, dois métodos foram concebidos, ASBsm - inspirado na Pesquisa Tabu, e RPW-VNDbal - baseado na meta-heurística "Variable

Neighborhood Descent" e também usando o "Ranked Positional Weighted method". Quanto ao Sequenciamento, dois métodos são propostos, VND-MSeq - envolvendo "Variable Neighborhood Descent" e GA-MSeq - baseado em Algoritmos Genéticos. Diferentes regras de despacho são utilizadas para criar soluções / populações iniciais. Finalmente, a abordagem integrada (Balanceamento e Sequenciamento simultaneamente) contempla duas linhas de investigação: otimização com um só objetivo ou multiobjectivo. Os métodos desenvolvidos são o GA, com um único objetivo, e o NSGA-II, um método multiobjectivo. A tese completa-se com a apresentação e discussão dos resultados computacionais, baseados em instâncias de problemas gerados e em dados reais da empresa. Como conclusão, pode-se afirmar que os métodos desenvolvidos são eficazes e que os resultados obtidos são melhores que os conseguidos pela empresa.

Embora o trabalho seja influenciado pelos problemas de balanceamento e sequenciamento de modelos mistos de uma grande empresa específica, acredita-se que os novos métodos propostos podem ser aplicados diretamente a outras indústrias de calçado e, possivelmente, após as adaptações adequadas, a fileiras industriais semelhantes.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Goals 4

1.3 Contributions 6

1.4 Document Structure 10

2 Footwear Industry **13**

2.1 An Overview 13

2.2 Footwear Production Process 15

2.3 Stitching Systems 17

3 Literature Review **23**

3.1 Optimisation Problems and Methods 24

3.1.1 Exact Methods 27

3.1.2 Approximate Methods 28

3.1.3 Multi-Objective Problems (MOP) 34

3.2 Balancing Problems 36

3.2.1 Classification of Balancing Problems 36

3.2.2 Solution Approaches 40

3.3 Sequencing Problems 50

3.4 Simultaneous Balancing and Sequencing Problems 64

4 Balancing Mixed-model Assembly Systems **71**

4.1 Optimisation Model 72
4.2 First Solution Method: ASBsm 75
4.2.1 ASBsm: Constructive Heuristic 75
4.2.2 ASBsm: Improvement Heuristic 81
4.3 Second Solution Method: RPW-VNDbal 84
4.3.1 RPW-VNDbal: Constructive Heuristic 85
4.3.2 RPW-VNDbal: Variable Neighbourhood Descent 88
4.4 Simulation and Conclusions 92

5 Sequencing Mixed-model Assembly Systems 97
5.1 Optimisation Model 98
5.2 Proposed Solution Method: VND-MSeq 100
5.2.1 VND-MSeq: Constructive Heuristic 101
5.2.2 VND-MSeq: Variable Neighbourhood Descent 109
5.3 Proposed Solution Method: GA-MSeq 115
5.3.1 GA-MSeq: Initial Population 115
5.3.2 GA-MSeq: Genetic Algorithm 116
5.4 Conclusions 124

6 Simultaneous Balancing and Sequencing Mixed-model Assembly Systems 127
6.1 Optimisation Model 128
6.2 Multi-Objective Solution Method: NSGA-II 133
6.2.1 NSGA-II: Non-dominated Sorting-based Approach (NS) . . . 136
6.2.2 NSGA-II: Genetic Algorithm 137
6.3 Single-Objective Solution Method: GA 149
6.4 Conclusions 150

7 Computational Results 153
7.1 Balancing Results 154
7.2 Sequencing Results 162
7.3 Balancing and Sequencing Results 170

7.4 Conclusions 176

8 Conclusions **179**

References **183**

List of Figures

1.1 Different Goals of this Work with their Objectives. 6

2.1 Stitching Workstation. 16
2.2 Lasts in the Assembly Line in the Shoes. 16
2.3 Boxes in the Stitching Line. 18
2.4 Larger Stitching System. 19
2.5 Smaller Stitching System. 19
2.6 Workstation Structure. 21
2.7 Different Systems of Stitching Lines. 21

3.1 Optimisation Problems. 25
3.2 Optimisation Methods. 26
3.3 Mathematical Models. 27
3.4 Different Category of Metaheuristic Methods. 30
3.5 Classification of ALBP Considering the Products and Processing Times. 37
3.6 Different Types of Scheduling. 50
3.7 Characteristics and Constraints for JSSP. 52
3.8 The Most Well-Known Performance Measures. 53
3.9 Basic dispatching rules. 55

4.1 Different Neighbourhood Structures. 90
4.2 VND Algorithm. 93
4.3 General Rules of Both Solution Methods. 94

4.4 Differences of Both Solution Methods. 95

5.1 Ns1, Swap Structure. 112
5.2 Ns2, Transfer Structure. 112
5.3 Ns3, Critical Tasks Transfer. 113
5.4 Ns4, Transferring Critical Tasks and Swapping Randomly. 114
5.5 Generating Initial Populations. 117
5.6 A Chromosome for the *SeqInstance* (Given Example). 118
5.7 Chromosomes of Two-Point Crossover-1. 120
5.8 Offspring of Two-Point Crossover-1. 121
5.9 Chromosomes of Two-Point Crossover-2. 121
5.10 Offspring of Two-Point Crossover-2. 122
5.11 Mutation Inversion. 123
5.12 Mutation Insertion. 123
5.13 General Rules of VND-MSeq Solution Method. 125
5.14 General Rules of GA-MSeq Solution Method. 126

6.1 The Considered MALB&SP Problems and Objectives. 129
6.2 Different Pareto Fronts (F). 134
6.3 How NSGA-II works. 135
6.4 Crowding Distance (CD) Sample. 135
6.5 Dominance Concept. 137
6.6 NS Algorithm. 138
6.7 Gene Structure. 141
6.8 Applying Crossover Sequencing-Simple. 143
6.9 Applying Crossover Sequencing-Critical Tasks Regarding Operators. 144
6.10 Applying Crossover Assignment-Simple. 145
6.11 Applying Crossover Assignment-Operators with fewer Tasks. 146
6.12 Applying Mutation Sequencing-Critical Tasks Regarding Operators. 147
6.13 Applying Mutation Assignment-Operators with Maximum Movement. 148
6.14 General Rules of NSGA-II Solution Method. 151

6.15 General Rules of GA Solution Method. 151

7.1 Comparison of the Results of the Balancing Problems for the Real Data. 163
7.2 Comparing Different Results of the Sequencing Problem for the Real Data. 170
7.3 First Front Results for the Small-size Instances, Using NSGA-II, in Simultaneously Balancing and Sequencing. 174
7.4 Comparing Different Results of the Balancing and Sequencing Problems for the Small Instances. 174
7.5 Comparing Different Results of the Balancing and Sequencing Problems for the real Instances. 178

List of Tables

3.1 Summary of More Relevant Literature to the Balancing. 48

4.1 Production Order. 76
4.2 Creating Boxes and Quantities. 76
4.3 Model Routings. 76
4.4 TasModBox Matrix. 77
4.5 Machine Type Requirements for each Task. 78
4.6 Machine Types. 79
4.7 Tasks According to Operators' Skills 79
4.8 Special Tasks and Processing Times for Ma3. 79
4.9 Special Tasks and Processing Times for Op6. 80
4.10 Initial Solution of the ASBsm. 81
4.11 Neighbourhoods and Objectives. 83
4.12 RPW Values. 87
4.13 RPW Values for Boxes considering Quantities (RPWBQ). 87
4.14 Sorted RPWBQs. 88
4.15 Initial Solution of the RPW-VNDbal. 89

5.1 Input Data and the Decisions that should be Taken. 101
5.2 Model Routings. 102
5.3 Input Data. 102
5.4 LNS Values for Different Tasks. 103
5.5 Table of Tasks - Sorted LNS Values (1st way). 104
5.6 Table of Tasks - Sorted LNS Values (2nd way). 105

5.7 Table of Tasks - Sorted LNS Values (3rd way). 105
5.8 Table of Tasks - CP Values for Different Tasks. 106
5.9 Table of Tasks - Sorted CP Values (1st way). 106
5.10 Table of Tasks - Sorted CP Values (2nd way). 107
5.11 Table of Tasks - LPT Rules for the Given Tasks. 108
5.12 Table of Tasks - SPT Rules for the Given Tasks. 108
5.13 Initial Solution Considering Table 5.5. as the Table of Tasks. 109

6.1 Model Routings. 142

7.1 Computational Results of the Balancing Problems for the Small-size Instances, Using the Optimisation Model and ASBsm. 155
7.2 Computational Results of the Balancing Problems for the Small-size Instances, Using the Optimisation Model and RPW-VNDbal. 156
7.3 Real Instance Information. 158
7.4 Computational Results of the Balancing Problems Using ASBsm for Real Instances considering LB. 159
7.5 Computational Results of the Balancing Problems Using RPW-VNDbal for Real Instances considering LB. 160
7.6 A Comparison Between ASBsm and RPW-VNDbal Methods for the Balancing Problems considering the LB. 160
7.7 Computational Results of the Balancing Problems Using ASBsm Method for Real Instances considering the Company Reality. 161
7.8 Computational Results of the Balancing Problems Using RPW-VNDbal Method for Real Instances considering the Company Reality. 162
7.9 A Comparison Between ASBsm and RPW-VNDbal Methods for the Balancing Problems considering the Company Reality. 162
7.10 Computational Results of the Sequencing Problems for the Small-size Instances, Using the Optimisation Model and VND-MSeq. 164
7.11 Computational Results of the Sequencing Problems for the Small-size Instances, Using the Optimisation Model and GA-MSeq. 165

7.12 Sequencing Problem: Real Instance Data and Number of Operators and Machines Used. 166
7.13 Computational Results of the Sequencing Problems Using VND-MSeq Method for Real Instances. 168
7.14 Computational Results of the Sequencing Problems Using GA-MSeq Method for Real Instances. 169
7.15 A Comparison Between VND-MSeq and GA-MSeq Methods for the Sequencing Problem. 169
7.16 Computational Results of the Optimisation Models for the Balancing and Sequencing Problems Separately and Simultaneously. 172
7.17 Computational Results of Simultaneously Balancing and Sequencing Problems for the Small-size Instances, Using the Optimisation Model and GA. 173
7.18 Computational Results of Simultaneously Balancing and Sequencing Problems for the Small-size Instances, Using the Optimisation Model and NSGA-II. 175
7.19 Computational Results of Simultaneously Balancing and Sequencing Problems Using NSGA-II and the Best Previously Developed Methods. 176
7.20 Computational Results of Simultaneously Balancing and Sequencing Problems Using NSGA-II Method for Real Instances considering the Company Reality. 177
7.21 Summary Results of NSGA-II. 177

Acronyms

ACO	Ant Colony Optimisation
ALBP	Assembly Line Balancing Problems
ALWABP	Assembly Line Worker Assignment and Balancing Problems
ASBsm	Assembly System Balancing Solution Method
B&B	Branch and Bound
B&C	Branch and Cut
B&P	Branch and Price
BA	Bee Algorithm
BalandSeqInstance	An Instance Related to the Simultaneous Balancing and Sequencing Problem
BalInstance	An Instance Related to the Balancing Problem
BCO	Bee Colony Optimisation
BS	Beam Search
CD	Crowding Distance
CP	Critical Path
Cpro	Constraint Programming
CT	Cycle Time
E: Time	Ending Time
EA	Evolutionary Algorithms
F	Pareto Front or Front

FFGA Fonseca and Fleming's Multi-Objective Genetic Algorithms

FJSS Flexible Job Shop Scheduling

FJSSP Flexible Job Shop Scheduling Problems

FSS Flow Shop Scheduling

GA Genetic Algorithms

GALBP Generalised Assembly Line Balancing Problems

GA-Mseq Genetic Algorithm Mixed-model Sequencing

GRASP Greedy Randomized Adaptive Search Procedure

HGA Hybridised Genetic Algorithms

HLGA Hajela and Lin's Genetic Algorithms

HPSO Hybridised Particle Swarm Optimisation

HSA Hybridised Simulated Annealing

HTS Hybridised Tabu Search

ICA Imperialist Competitive Algorithm

ILP Integer Linear Programming

IP Integer Programming

JIT Just-In-Time

JSS Job Shop Scheduling

JSSP Job Shop Scheduling Problems

LE Line Efficiency

LNS Longest Number of Successors

LP Linear Programming

LPT Longest Processing Time

MALB&SP Mixed-model Assembly Line Balancing and Sequencing Problems

MALBP Mixed-model Assembly Line Balancing Problems

Different Versions of MALBP considering the objective function:

MALBP-1, MALBP-2, MALBP-3, MALBP-4

MALBP-5, MALBP-E, MALBP-F

MALSP	Mixed-model Assembly Line Sequencing Problems
MIP	Mixed Integer Programming
MOEA	Multi-Objective Evolutionary Algorithms
MOGA	Multi-Objective Genetic Algorithms
MOP	Multi-Objective Problems
Nb1 to Nb4	Varied Neighbourhood Structure, Related to the Balancing
NEH	Nawaz, Enscore and Ham
NLP	Non-Linear Programming
NPGA	Niched-Pareto Genetic Algorithms
NS	Non-dominated Sorting-based Approach
Ns1 to Ns5	Varied Neighbourhood Structure, Related to the Sequencing
NSGA	Non-dominated Sorting Genetic Algorithms
NSGA-II	Non-dominated Sorting Genetic Algorithms II
OR	Operational Research
PMS	Parallel Machine scheduling
PROMETHEE	Preference Ranking Organization METHod for Enrichment of Evaluations
PSO	Particle Swarm Optimisation
PT	Processing Time
RPW	Ranked Positional Weighted
RPW-VNDbal	Ranked Positional Weighted method and Variable Neighbourhood Descent
S: Time	Starting Time
SA	Simulated Annealing
SALBP	Single-model Assembly Line Balancing Problems
	Different Versions of SALBP considering the objective function:
	SALBP-1, SALBP-2, SALBP-3, SALBP-4
	SALBP-5, SALBP-E, SALBP-F
SeqInstance	An Instance Related to the Sequencing Problem

SMS	Single Machine Scheduling
SPEA	Strength Pareto Evolutionary Algorithms
SPT	Shortest Processing Time
SWB	Smoothing Workloads Balance
TS	Tabu Search
VEGA	Vector Evaluated Genetic Algorithms
VND	Variable Neighbourhood Descent
VND-Mseq	Variable Neighbourhood Descent Mixed-model Sequencing
VNS	Variable Neighbourhood Search
W	Workstation
WIP	Work In Process

Chapter 1

Introduction

1.1 Motivation

In a progressively competitive world, managers of the industrial environments, to succeed and survive, need to gain operational efficiency and consumer satisfaction. The footwear industry has been remarkably improving over the last years. The high variety and quality of the products and the international competitiveness are an impressive reality. Creative design, technological leadership in the type of materials and productive equipment, the progress in management solutions and skilled labour, among other aspects, have been crucial for such improved performance. While, in the past, a few models were produced in large numbers, the circumstances changed when the industry started relying on fashion. This led to production flexibility, both in volume and variety, and fast response times for a company to succeed. The obvious consequence is that the factories need to transform or adapt themselves to face the new production paradigm and simultaneously handle a variety of mixed models.

Portugal is one of the major world players in the footwear industry. This is the case because the country has chosen to invest in technology innovation, research, manpower qualification, innovative design and internationalisation. Much has changed, from low-cost mass production to serving clients consisting of small retail chains, where orders and models are varied. Consequently, work plans vary

frequently and traditional flow lines are steadily being replaced by more flexible and sophisticated systems. That is the case of the large footwear company considered in this work, Kyaia in Portugal, whose production is almost entirely for worldwide export. The company has invested in completely new flexible automated assembly lines (and participated in their design), which transport boxes with the components of different models, from and to warehouses, that can reach any workstation with specialised operators, in any order. These flexible assembly lines offer many possibilities but certainly they require the solution to complex balancing and sequencing problems, involving large dozens of boxes and workstations.

Footwear manufacturing encompasses major processes such as cutting, stitching and assembly. This study focuses on the stitching systems, which include a network of stitching stages related to the manufacture of the upper part of the shoes. The stitching process is critical in this industry, not only because it usually determines the quality of the footwear, but also because of the difficulty in managing the associated resources, operators and machines. The stitching systems are installed in a factory that exports a wide variety of models worldwide. Moreover, the stitching process is critical in this industry since the quality of the footwear is usually determined by these systems. Furthermore, it is important to have efficient and effective production lines, and that is why this research will be focusing on line balancing and sequencing. A more detailed presentation of the situation under study and how it relates to the assembly line balancing and sequencing literature follows below. Generic references are Ghosh and Gagnon [1989], Erel and Sarin [1998], Scholl and Becker [2006], Kriengkorakot and Pianthong [2007], Boysen et al. [2007b], Boysen et al. [2009], Battaï and Dolgu [2013], Sivasankaran and Shahabudeen [2014] and Saif et al. [2014b].

As mentioned previously, the footwear industry involves the following main phases: cutting, stitching and assembly. The speed of the cutting and assembly processes is higher than stitching, and that is why this operation is considered a bottleneck in this industry. To overcome this trouble, some companies send the work pieces to other companies for stitching, some try to use more manual labour

and machines, while others implement further strategies to balance the cycle time of all production lines.

The stitching systems integrate new automatic transportation equipment that will only be effective if it is well managed. Given that production plans change rapidly, it is crucial to know and adapt the number of operators and machines required and the time table of the tasks. This study essentially focuses on balancing and sequencing the resources (operators and machines). This includes allocating tasks to a minimum of workstations, equilibrating operators' work time and minimising the total completion time of the line. The pieces of the footwear are placed inside boxes, which move through the stitching lines, in any direction, to be worked on the assigned workstations. The operators are classified according to their skills. The following points are also essential for this work:

Variety of models: The footwear industry produces shoes for men, women and children. Each of these groups requires distinct models, which also change according to the season. Clients' behaviour has also changed, which, in turn, has led to a decrease in the number of orders per model. As an effect, it is appropriate to have several models in the production lines simultaneously. The associated balancing problems are usually known as Mixed-model Assembly Line Balancing Problems ($MALBP$). Sequencing is also considered and if it is studied after balancing, then it is named as Mixed-model Assembly Line Sequencing Problems ($MALSP$) and in case they are expressed together (balancing and sequencing), the related issue is Mixed-model Assembly Line Balancing and Sequencing Problems ($MALB\&SP$). These are subjects of the present work, with the particularities that parts of a model are inside boxes, moving along the transportation systems, and operators with different skill levels and machines from various types are also involved.

Boxes in the stitching systems: Shoe components are put inside boxes, which can move to and from automatic warehouses and between workstations, due to the existing advanced transportation systems. This is an advantage of these systems, however it originates balancing and sequencing problems, distinct from the current ones in the literature - a box is not restricted to move to the closest

workstation. It can go to any available adequate workstation. Each box contains parts of the same model that are equal in size (size of a shoe) and identical in colour of different clients orders. The company prefers to create boxes with 10 sets. That is to say, if a production order of a specific model contains 35 pairs, 4 boxes can be created with 10, 10, 10 and 5 items each. However, there may be exceptions (boxes with amount more than 10 pairs).

Different tasks: To produce a pair of shoes, it is essential to divide the work needed into a set of tasks. Each task requires a certain processing time. Additionally, each model has a special production routing, which increases the difficulty of the balancing process. For example, it is not possible to sew some parts before they have been glued and assembled. Besides, other exigencies apply, such as the requirement of skilled operators. This is also a particularity of the problem considered in this work.

The practical problems considered and solved in this thesis are unique. The novelties come also from their description and balancing and sequencing modelling, in the context of new flexible stitching systems in a large footwear company. Such problems simultaneously involve operators with various skill levels and different types of machines, and the units moving in the lines are not pairs of shoes but boxes containing numerous quantities of product components. In addition, new solution methods are proposed, which are described in the proper chapters. Let us emphasise that this work also has a clear practical scope, influenced by the need to contribute to overcoming the daily difficulties of a large company with new technologies to be managed.

1.2 Goals

This work intends to provide new insights into balancing and sequencing mixed-model assembly lines problem-solving. As mentioned before, a case study is used to motivate and identify the main challenges and also to assess the performance of the models and methods proposed.

A central interest of the industry is the optimisation of the available resources,

such as the number of workstations that are opened, and obtaining higher production results, by minimising the maximum completion time of all tasks. Another contribution of this work is smoothing the workload of the operators, in a way that operators work approximately the same time. Briefly, this research contributes both to the scientific and practical levels of the topic *Line Balancing and Sequencing*, which is achieved by engaging in two main phases: sequential balancing and sequencing and simultaneous balancing and sequencing.

Sequential balancing and sequencing: The focus of the work in this step is first on balancing mixed-model assembly lines, by considering the objectives minimising the number of workstations and smoothing operators' workload. The next step is sequencing mixed-model assembly lines. Both appoint to a decision-making process that plays a crucial role in most manufacturing industries. A proper allocation and sequence of tasks enables the company to optimise its objectives which are here minimising the number of workstations, smoothing operators' workload and minimising the maximum completion time. However, the available time is a one day production order.

Simultaneous balancing and sequencing: This step considers balancing and sequencing problems together and the objective is the combination of all the mentioned objectives, which are minimising the number of workstations, smoothing operators' workload and minimising the maximum completion time of the tasks. It is considered as both multi-objective and single-objective.

Common targets to be attained: In this context, for mentioned phases, some targets are identified, developing a mathematical model and applying approximate methods and metaheuristics to achieve good solutions in a short time. simulation is also used to evaluate the balancing solutions.

Modelling real situations is a collaborative and dynamic procedure. The aim of this activity is to understand as much as possible the balancing and sequencing problems under analysis, so that relevant aspects are not overlooked. As somehow expected, the problems may be characterised as optimisation problems and formulated as Mixed Integer Programming (MIP).

Approximate methods are used for obtaining solutions for the balancing and sequencing problems and attempt has been made that they are simple but efficient as much as possible. Metaheuristics are usually capable of generating high quality solutions and are able to cope with the multiple details and issues inherent to the balancing and sequencing problems in the footwear industry (issues which usually cannot be readily handled by simple heuristics or MIP models). Figure 1.1 depicts the mentioned goals and targets.

Goals of This Study			
Different Targets		Objectives	
Sequentially	1- Balancing	Single-Objective	Minimising the Number of Workstations + Smoothing Operators' Workload
	2- Sequencing	Single-Objective	Minimising the Maximum Completion time of the Line
Simultaneously	1- Simultaneous Balancing and Sequencing	Single-Objective	Minimising the Number of Workstations + Smoothing Operators' Workload + Minimising the Maximum Completion time of the Line
		Multi-Objective	First Part: Minimising the Number of Workstations + Smoothing Operators' Workload Second Part: Minimising the Maximum Completion time of the Line

Figure 1.1: Different Goals of this Work with their Objectives.

1.3 Contributions

This thesis has several contributions, having as background a real industrial case and the central theme of the MALB&SP. The main contributions of the work are as follows:

1. An optimisation model for balancing problems in new assembly systems; two new solution methods based on approximate and metaheuristic procedures for the balancing problems; a simulation model to evaluate the results.

2. By considering the balancing results, we contemplate: an optimisation model

for sequencing problems in new assembly systems; two new solution methods based on metaheuristic procedures for the sequencing problems;

3. Balancing and sequencing are considered together: an optimisation model for simultaneously balancing and sequencing problems in new assembly systems; single and multiple objectives are considered; the solution methods are based on a multi-objective metaheuristic approach and single-objective metaheuristic.

The work related to this thesis resulted in the following publications and presentations.

1. Journal papers (published and submitted):

   - P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Balancing Mixed-model Assembly Systems in the Footwear Industry With a Variable Neighbourhood Descent Method. (Computers & Industrial Engineering Journal, 2018), doi.org/10.1016/j.cie.2018.05.020,
     URL: `https://www.sciencedirect.com/science/article/pii/S0360835218302213`
   - P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Using a Genetic Algorithm and a Variable Neighbourhood Descent for Sequencing Mixed-model Assembly Systems in the Footwear Industry, 2018 (submitted)
   - P. Sadeghi, R. D. Rebelo, J. S. Ferreira, A multi-Objective Simultaneous Balancing and Sequencing Mixed-model Assembly Systems in the Footwear Industry, 2018 (submitted)

2. International conference paper proceedings:

   - P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Balancing a Mixed-model Assembly System in the Footwear Industry, In Proceedings of the IFIP International Conference on Advances in Production Management Systems, 2017 (APMS2017) doi.org/10.1007/978-3-319-66923-6_62,

URL: https://repositorio.inesctec.pt/handle/123456789/5872

3. Conference proceedings:

   - P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Sequencing Mixed-model Assembly Systems in a Footwear Industry, In 9th Industrial Engineering and Management Symposium, 2018 (IEMS'2019)

     URL: https://web.fe.up.pt/~degi/iems19/files/bookletIEMS19.pdf

   - P. Sadeghi, R. D. Rebelo, J. S. Ferreira, A Genetic Algorithm for Balancing and Sequencing Mixed-model Assembly Lines, In 9th Industrial Engineering and Management Symposium, 2018 (IEMS'2018),

     URL: https://paginas.fe.up.pt/~degi/iems18/files/bookletIEMS18.pdf

   - P. Sadeghi, R. D. Rebelo, J. S. Ferreira, A Variable Neighbourhood Descent Approach for the Assembly Line Balancing Problem, In 8th Industrial Engineering and Management Symposium, 2017 (IEMS'2017),

     URL: https://paginas.fe.up.pt/~degi/iems17/files/bookletIEMS17.pdf

   - P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Balancing and lot-sizing mixed-model lines in the footwear industry, In 7th Industrial Engineering and Management Symposium, 2016 (IEMS'2016),

     URL: https://paginas.fe.up.pt/~degi/iems16/files/bookletIEMS16.pdf

   - P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Mixed-Model Assembly Line Balancing in the Footwear Industry, In 6th Industrial Engineering and Management Symposium, 2015 (IEMS'2015)

     URL: https://sigarra.up.pt/fpceup/en/PUB_GERAL.PUB_VIEW?pi_pub_base_id=19000

4. Other international and national conferences:

- P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Balancing Mixed-model Assembly Systems With a Variable Neighborhood Descent Method, In the 21th Conference of the International Federation of the Operational Research Societies, 2017 (IFORS2017)

  URL: https://www.euro-online.org/conf/ifors2017/display_abstract?paperid=1952

- P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Balancing Mixed-model Assembly Systems with a Variable Neighbourhood Descent Method, In the Congress of the Portuguese Association of Operational Research, 2017 (IO2017)

  URL: http://www.norg.uminho.pt/IO2017/program/f_1_4.htm

- P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Assembly line Balancing in the Footwear Industry, In 28th European Conference on Operational Research, 2016 (Euro2016)

  URL: https://www.euro-online.org/conf/euro28/display_abstract?paperid=2887

- P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Mixed-Model Assembly Line Balancing in the Footwear Industry, In Optimization 2014 Conference, 2014

  URL: http://optimization2014.dps.uminho.pt/program/m_b_3.htm

- P. Sadeghi, R. D. Rebelo, J. S. Ferreira, Management of Assembly Lines in the Footwear Industry, In the Congress of the Portuguese Association of Operational Research, 2013 (IO2013)

  URL: http://www.io2013.ipb.pt/docs/ProgramaIO2013.pdf

1.4 Document Structure

The thesis is organised in 8 Chapters. Chapter 1 introduces the theme of the work, its goals and main contributions and describes the structure of the thesis.

Chapter 2, briefly introduces some concepts related to the footwear industry considering balancing and sequencing problems. In-depth research on the case study is conducted in this Chapter and the developments and findings are generalised to other problems in the footwear industry or other similar process industries.

Chapter 3, reviews the literature relevant to this research. Different optimisation methods are discussed. In the following, the balancing and distinct classifications are explained in this part and then sequencing and relevant issues are mentioned. The methods for solving balancing and sequencing problems on the mixed-model lines are separately and simultaneously considered in this Chapter. Besides, issues related to the footwear field are discussed.

Chapter 4 approaches balancing mixed-model assembly systems in the footwear industry. An optimisation model is proposed and solved by CPLEX only for small instances. Next, we had to devise two approximate methods. First, a constructive heuristic and an improvement heuristic, which take inspiration from Tabu Search (TS), are used. The second one is based on the Variable Neighbourhood Descent (VND) metaheuristic and integrating an adaptation of the Ranked Positional Weighted (RPW) method. The adapted RPW method is used to create initial feasible solutions. After choosing good initial solutions, VND is applied to improve their quality. Since sequencing is not considered, a simulation is used to validate the results. Different issues such as operator with various skill levels, machine with different abilities, mixed-model line and Work In Processes (WIP) are dealt with in this Chapter.

In Chapter 5, the mathematical model and solution method are revised to be incorporated in sequencing mixed-model line and also deal with a variety of issues, different priority levels for orders, boxes with different quantities, operator movement among machines and WIP. Different dispatching rules are used to generate initial solutions and then, Genetic Algorithms (GA) and VND are used as

improvement approaches.

Chapter 6 approaches balancing and sequencing simultaneously in the mixed-model assembly lines. Different dispatching rules are considered to create initial populations and a multi-objective GA based on the non-dominated sorting method which is called Non-Dominated Sorting Genetic Algorithms II ($NSGA-II$) as a solving procedure is proposed in this Chapter. A GA also developed as a single-objective procedure, besides, a mathematical model is developed.

In Chapter 7, the computational results of different methods are provided for the given problems in the case study instances. The instances from the company of the case study are related to two different stitching systems and since comparisons are made with real data, WIP are also involved. Moreover, for the given problems, a lower bound is calculated in this Chapter.

Finally, the conclusion is drawn in Chapter 8, together with future lines of research.

Chapter 2

Footwear Industry

2.1 An Overview

The footwear industry embraces a vast range of materials including plastic, textiles, leather and rubber and also products ranging from different kinds of footwear for men, women and children to more specialised products like snowboard boots and footwear (according to the Union [2018]).

In the past, models did not have much variety, but today footwear industry is constantly changing. However, given that this industry is greatly associated with fashion, they decrease orders of many different models. Considering that numerous models are available simultaneously on the line, each having a different delivery date, shoe production under such conditions should be accompanied by a lot of innovation.

As stated in the Union [2018] website, two thirds of EU footwear production mainly come from three countries: Italy, Spain, and Portugal. Thus, in today's global economy, the footwear industry is a major industry in Portugal.

According to the Portuguese-Shoes [2018] website, footwear industry in Portugal is primarily focused on the international markets that export more than 95% of its products. In recent decades, the Portuguese footwear industry has faced rapid and remarkable transformation. Footwear companies prepared themselves to face the challenge of modernising their facilities and production methods. Thus, they

invested in the less tangible aspects that gave them that competitive edge. Currently, Portuguese companies are recognised across the world for the excellence of their service and a quick response to the market needs and requirements in addition to the quality of their footwear.

In Portugal, the footwear industry still has its innovation process and its largest challenge is the conciliation of the tradition and know how to equip the generations with the most modern technology, flexibility, and excellent design. This is the challenge that is simultaneously one of the distinctive characteristics of this industry.

Based on the latest data, this industry includes over 1,450 companies, employing around 40,000 individuals. A major part of businesses that form the Portuguese footwear industry are small and medium-sized companies. These companies mainly engage in exports. Therefore, the competitiveness of the Portuguese footwear sector is mostly based on structures that distinguish themselves from their competitors since they are very flexible and extremely well-equipped in technological terms, and which present fast and effective services for customers (Portuguese-Shoes [2018]).

In 2016, Portugal has exported 81 million pairs of shoes to 152 countries in 5 continents, amounting to 1.923 million euros. The previous year (2017) was also the seventh consecutive year in which we witnessed development in sales to the foreign markets. During this short time, the Portuguese footwear industry has provided an outstanding growth dynamic and has increased its exports income by about 50% (Portuguese-Shoes [2018]).

The *FOOTure* Programme is supposed to be performed by APICCAPS, the Portuguese Footwear, Components and Leather Goods Manufacturers' Association. Its goal is that the Portuguese Footwear Industry becomes one of the most competitive and modern industries worldwide by the year 2020. The *FOOTure* programme has four main objectives including internationalisation, qualification of human resources, innovation and social responsibility. To make sure that these objectives are fulfilled, an investment of 160 million euros has been predicted until the year 2020 (Portuguese-Shoes [2018]).

2.2 Footwear Production Process

Shoes are classified into different categories including classic shoes, athletic shoes, school shoes, high heels and the like. Producing shoes can be regarded as a traditional handicraft profession. Nevertheless, it has currently been taken over mainly by industrial manufacture of footwear. Various materials are applied for making shoes.

Shoe production usually requires some processes like cutting, stitching and assembly and this industry tends to produce a variety of models because of changes in people's wishes and desires. Accordingly, there are different models at the same time in the line and each of them has different routing with a variety of tasks and different processing times.

In the cutting stage, the upper part of the shoe is made. Materials are cut in order to be prepared to be sent to the next line. Given that they are expensive if done manually, this operation requires a high level of skill. Today, there are some companies that have automatic cutting machinery and employ high quality software to determine good nestings of the pieces to be cut, and hence decreasing the costs of materials.

The next step is pre-stitching in which workpieces are prepared for the stitching line.

In the stitching line, the upper part of a shoe is made and the three dimensional upper part is sewed by the sewing machine. Various edge treatments are performed onto the leather in order to give an attractive look to the finished upper. In this stage, operator expertise is of great importance (Figure 2.1).

The upper parts after completion are molded into a shape of foot with the aid of a *Last*. Last is a plastic shape that imitates the foot shape. Later, it is removed from the finished shoe to be applied further in producing other shoes. Ultimately, an insole is attached to the bottom of the last in the assembly line (Figure 2.2).

In the end, they are polished and waxed in order to make them attractive and to make sure that the edge is waterproof. The sole bottom is often lightly buffed,

Figure 2.1: Stitching Workstation.

Figure 2.2: Lasts in the Assembly Line in the Shoes.

stained and polished and various patterns are marked on the surface to give it a craft finished look. Now, the shoe production is completed.

Entering the shoe industry is associated with competition and marketing is another aspect involving the manufacturing of footwear. Thus, there is a need for creativity in order to be successful. People mostly purchase shoes because they are fond of a particular style or function of the shoe. Still, customer service also affects a customer's decision to buy a particular kind of shoes. Therefore, formulation of a marketing strategy in connection with what sets your customer service apart can

be helpful for the company to attract business.

There are different types of clients, including purchasing from the stores and online buyers customers. They form the first and second categories; the next group consists of the shop sellers called big clients. Hence, concerning the production plan, three types of clients can be considered for the companies, namely big clients, small clients and clients from virtual stores. Clients' orders from the virtual store usually create difficulty for the production plan for two reasons: first because their quantity is less and second, due to date restriction.

The company should typically receive the orders one season earlier and after the receipt of orders from the virtual store, sufficient raw materials should be available for shoe production; otherwise, customer demands cannot be met. So, according to the prediction, the company should order the required raw material and recognise suppliers.

In this industry, both manpower and machine are of crucial importance and considering that the production plan changes at a rapid pace, it is crucial to answer the question as to how many resources are required.

This study is primarily focused on the stitching lines and resource allocation. Besides, while putting the workpieces inside boxes, which will move in the stitching lines, boxes including different quantities of workpieces, and not the pair of shoes, will be regarded as unities for planning purposes. Figure 2.3 displays boxes in a stitching line.

2.3 Stitching Systems

Two different stitching systems exist, each of which can be found in a separate place within the factory. The larger system (as shown in Figure 2.4) encompasses four parallel lines with the machines facing a transporter. Through the transporter, the boxes are transferred from the warehouses to the workstations (workstation includes a combination of an operator and a machine) and vice-versa, and between workstations, using special passages. In general, contrary to the traditional lines in which all boxes move in one direction and pass by the workstations in the same

(a)

(b)

Figure 2.3: Boxes in the Stitching Line.

sequence, in the systems under consideration, boxes can directly and in any sequence move to any workstation. The mentioned boxes are put in buffers with special designs. Afterwards, the operator receives the box from the upper part of the buffer, carries out the specified tasks and then puts it on the transporter. Buffers are not considered in the modelling process because they do not represent a bottleneck. The possible 190 stop points limit the number of machines along the transporter.

The smaller system with a ”U” shape (as displayed in Figure 2.5) is different. Here, the boxes move directly between any two workstations; but in contrast to the first system mentioned above, there are no special warehouses. The number of

Figure 2.4: Larger Stitching System.

machines placed along the transporter is restricted by the number of possible stop points, which is equal to 42.

Figure 2.5: Smaller Stitching System.

Balancing problems are classified in different ways which will be explained in the next Chapter. However, as stated previously, the most useful approach for this

new industrial case is MALB&SP.

Equipment selection: Workstations consist of both machines and operators. This work includes a process of choosing equipment and operators. If a process needs distinct equipment and/or operator skills, that will be taken into consideration.

Operators and skills: The operator' skill level is considered in line balancing and sequencing, meaning that the assignment of an appropriate operator to the right task is of great importance. Special operators can be found in the company who are the only ones capable of doing particular tasks and hence, these tasks should be assigned beforehand to those specified operators.

Machine restrictions: As previously stated, machines must be allocated to workstations. Machines can be classified into different types with respect to their capability to perform particular tasks. Further, each operator can work with different machines while each machine has only one operator. A machine can have more than one type. A singular situation may happen, if a task needs a certain type of machine and only one of these machines is available. In such conditions, this machine must be used for that task, even if that machine is characterised by other types. Afterwards, if that same machine still has sufficient free time, it may be applied for other tasks that require the other types of machine. For instance, consider the following situation: if task I needs a type B machine, and machine 5 is the only type B machine available, then machine 5 should be chosen and used for an operator, firstly as a type B machine, although it may also be characterised as a machine type B or C.

Workstations: As already pointed out, the systems under investigation are quite new and have been planned to fulfil the main requirements of this industry. Specifically, workstations' shape and organisation are such that boxes bottlenecks are avoided. The workstations' structure can be observed in Figure 2.6. Accordingly, balancing and sequencing are essential issues and given that this case is a mixed-model line as explained in Chapter 1, the balancing problem is called MALBP, and because the sequencing is taken into account, it is named as MALSP. If balancing and sequencing are addressed simultaneously, it is MALB&SP. The clas-

sification of the problem will be described in the next Chapter. This MALB&SP is unique because of the automated transportation system and the industrial sector.

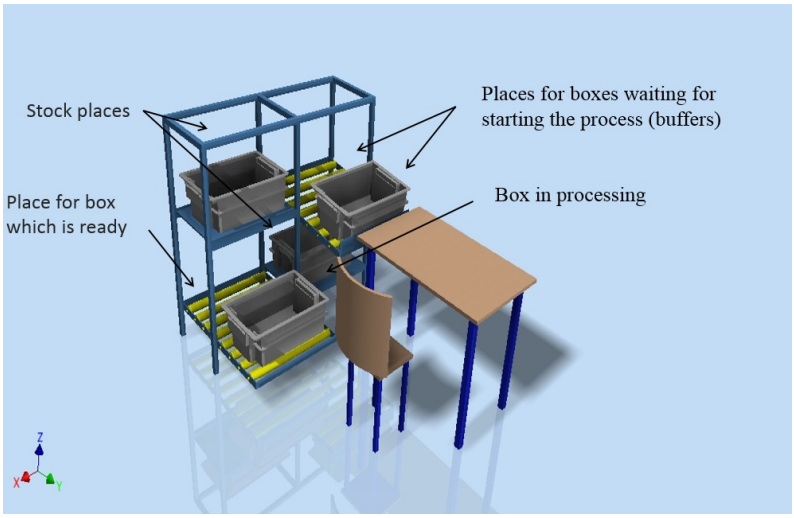

Figure 2.6: Workstation Structure.

Additionally, as mentioned previously, this work focuses on the problems in a large Portugese footwear company. This company has made investment in completely new flexible automated assembly lines, which transport boxes to any workstations, including the components of various models.

Furthermore, the general layout of both systems under investigation along with the workstations are shown in Figure 2.7.

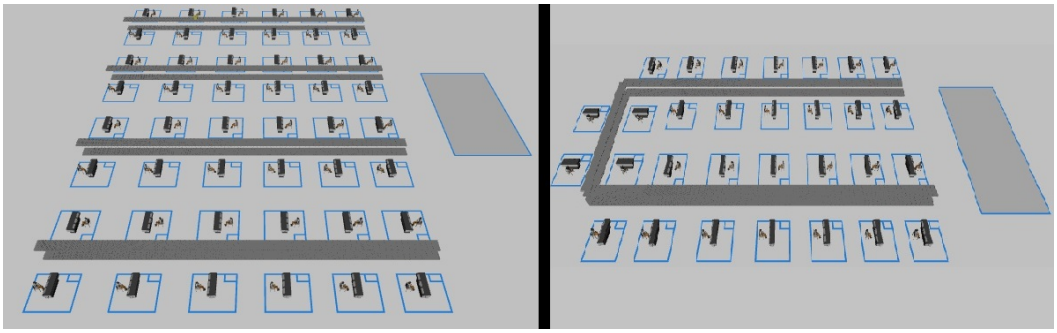

Figure 2.7: Different Systems of Stitching Lines.

Daily line balancing: This study investigates daily line balancing and sequencing. As a result, Work In Process (WIP) should be considered frequently

since there are boxes in the systems along with several tasks that have already been performed. Then, the input data (such as updating routings) is adapted. For example, if the quantities existing in the boxes, which are under processing, are summed up, they amount to approximately 3400 pairs of shoes.

This work mainly focuses on resource balancing, convenient allocation of tasks to workstations and establishment of their sequence. These flexible assembly lines provide numerous possibilities, but they definitely need a solution to complex balancing and sequencing problems. The literature associated with these problems will be examined in the next Chapter.

Chapter 3

Literature Review

The Chapter on literature review focuses on optimisation problems, solution methods and balancing and sequencing problems. We are not going to provide a comprehensive review of literature regarding optimisation and only the relevant publications, in connection with balancing and sequencing problems and their solutions, will be presented here.

The materials are provided in four main sections. In the first section, different optimisation problems and related solutions will be covered, including exact and approximate methods to deal with the balancing and sequencing problems. Next, the most applied approaches to solve multi-objective problems are presented. For this reason, in the first section, we witness a simple reference at the beginning and then, the chief methods in relation to the subject are introduced and classified. In the second section, the balancing and distinct classifications are discussed. The third section is associated with the sequencing problems. The aforesaid sections deal with balancing and sequencing separately, but in section 3.4, simultaneous balancing and sequencing is explained which has not been considered as extensively in the study. In brief, the reason why this structure is applied for the review of literature is that these types of methods are usually employed to handle the problems addressed in this work.

3.1 Optimisation Problems and Methods

Basically, decision-makers in the real world face the problems of increased complexity arising in various technical sectors. The problem to be solved can be stated as an optimisation problem (Dréo et al. [2006]). In general, the optimisation problem address various input data and decision variables and the relationship between them is characterised by constraints and different objectives for solving an optimisation problem. However, the optimisation problem can be categorised based on the type of variables and relation among them. Further, the methods used to solve the problem are various. There are different items that have been considered and defined to solve a problem, including the following:

Decision variables:

These variables consist of the numerical quantities that should be specified in order to solve the problem.

Constraints:

Constraints refer to the restrictions which are imposed by the specific features of the environment or the existing resources and should be obeyed. Indeed, in order for a given solution to be acceptable, the constraints must be observed.

Objective function:

When we want to solve an optimisation problem, solutions are found whose quality is dependent upon the selected criteria which are transformed into computable expressions of the decision variables called objective functions.

These conditions categorise the problems into discrete and continuous.

In Figure 3.1, various kinds of optimisation problems can be observed. There are some optimisation models that make sense only when decision variables assume values from a discrete set (usually a subset of integers) while we can find other models with variables taking on any real values. Under such conditions, the problems are classified into discrete and continuous types.

There is another important classification of problems: Problems with variables on which there are no constraints and problems with variables on which there are constraints. In the first category, i.e. optimisation problems with no constraints,

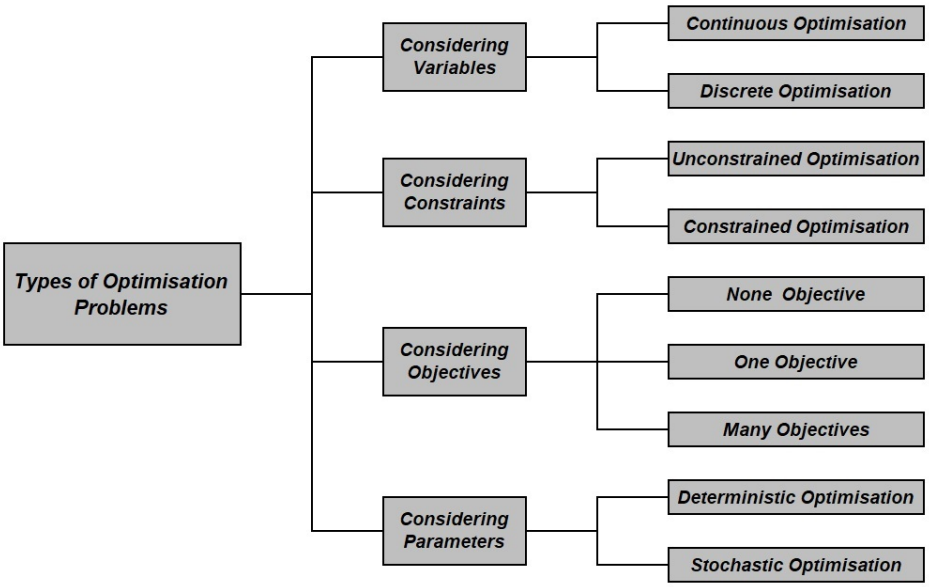

Figure 3.1: Optimisation Problems.

the constraints actually exist but are replaced by a penalty term in the objective function.

There are some problems with no objective function, whose aim is to find a feasible solution. Also, there some other problems having one objective or multi-objectives. As stated by Osyczka [1985], the problem of multi-objective optimisation (also called multi-criteria optimisation, multi-performance or vector optimisation problem) occurs naturally in many disciplines, continuing the researchers to struggle to solve them. In despite of various techniques developed in Operational Research (OR) and other disciplines to handle these problems, their solution are so complex that require alternative approaches. Specifically, as maintained by Coello et al. [2007], the goal of multi-objective problems (MOP) is to simultaneously optimise k objective functions. Here, we may need to maximise all k functions or minimise all k functions or combine the maximisation and minimisation of these k function.

In connection with the deterministic optimisation, we assume that the input data for a specific problem is accurately known. In contrast, a problem is regarded as stochastic optimisation if the data cannot be known accurately for some reasons. In the stochastic optimisation models, there is an element of uncertainty.

In order to solve balancing and sequencing problems, various methods are available that can be categorised as follows and are common for combinatorial optimisation problems: Optimum seeking algorithms or exact methods and Approximate procedures. A classification of the main optimisation methods is provided in Figure 3.2 (following Talbi [2009] and Battaï and Dolgu [2013]). The methods that are used in the thesis, differently coloured in that figure, are based on Mathematical Programming, Heuristics and Metaheuristics. However, considering the picked objective, as mentioned before there are various kinds of problems such as single- and multi-objective. Therefore, in relation to the problems given, the exact and approximate approaches as well as the multi-objective problems will be presented below.

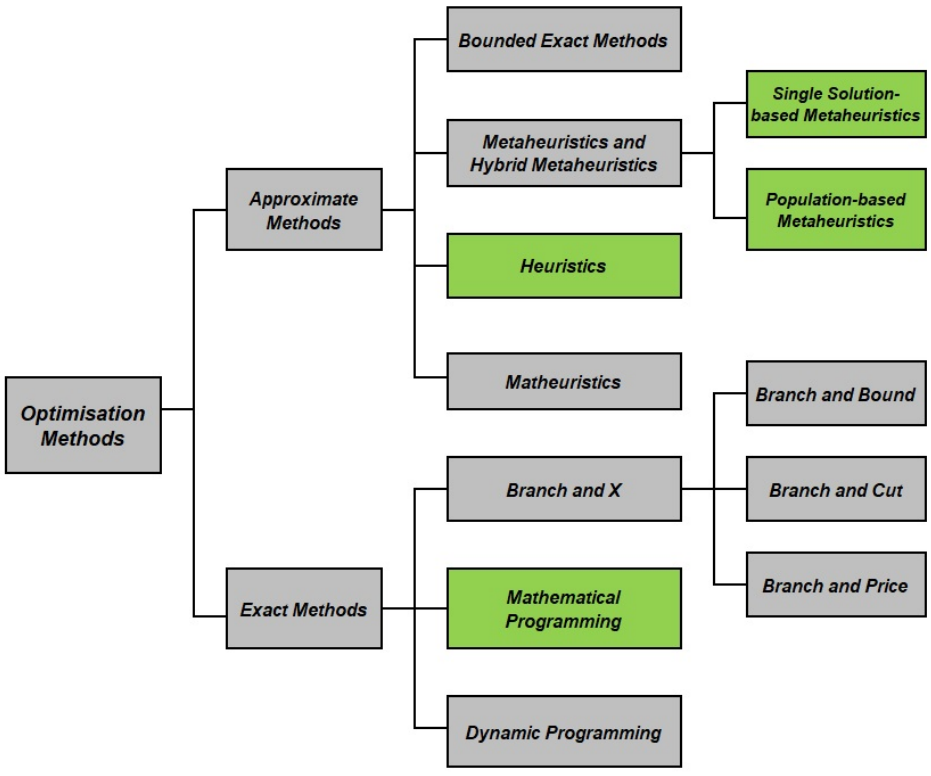

Figure 3.2: Optimisation Methods.

3.1.1 Exact Methods

To solve optimisation problems, there are different ways available. In this section, exact methods are taken into account. As Figure 3.2 displays, exact methods are classified into the following: Branch and X, Mathematical Programming and Dynamic Programming. Figure 3.3 depicts categories of Mathematical Programming, which have been obtained according to Talbi [2009]. In the following, the aforesaid methods are briefly provided:

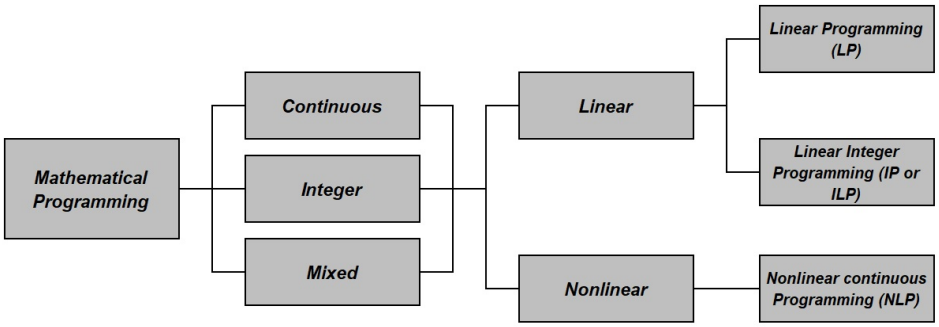

Figure 3.3: Mathematical Models.

Branch and Bound (B&B): We can observe general treatments of B&B search in Brusco and Stahl [2006]. Using this method, we can find optimal solutions. B&B is, in fact, a strategy of "*divideandconquer*". This term means the breaking of problems into sub-problems so that the problem becomes easier to solve. The idea is that the feasible region is split into more manageable subdivisions and the subdivisions can further be divided if necessary. On the whole, the feasible region can be divided through different ways, and consequently, several B&B algorithms exist as stated in Bradley et al. [1977]. However, this method has a drawback. It is very time-consuming due to the high number of nodes (Pinedo and Chao [1999]).

Branch and Cut (B&C) and Branch and Price (B&P): B&C method, as mentioned by Mitchell [2011], is an algorithm that is applied to solve different Integer Programming (IP) problems and can provide a guarantee of optimality. There are numerous problems involving non-continuous variables that possess integer values. The mentioned problems can be solved using the B&C method, which is

an exact algorithm including a combination of a cutting plane method and a B&B algorithm. Additionally, large Mixed Integer Programmings ($MIPs$) cannot apply pure cutting-plane approaches because they are not efficient. Thus, B&C comprises B&B + cutting planes; i.e. row generation. B&P is the same as B&C, with the exception that in the procedure, column generation and not row generation is focused as expressed in cited in Barnhart et al. [1998]. It can be said that pricing and cutting are considered as complementary procedures for tightening a LP relaxation.

Mathematical Programming approaches: In order to solve a problem, a mathematical model can be employed in which the relations between various variables are described and the values of each variable is then specified through recognising the input parameters and the mentioned relations. Mathematical models are divided into different categories including the following: linear and non-linear, static or dynamic, discrete or continuous.

Dynamic Programming methods: Dynamic programming is another optimisation method, often used when planning over time is fundamental. This method, as discussed by Talbi [2009], recursively divides a problem into simpler subproblems and is based on Bellman's principle of optimality, Bellman [2013], stating that "Any sub-strategy of an optimal strategy it is also an optimal strategy, with regard to initial and final states of the sub-strategy".

3.1.2 Approximate Methods

Numerous common optimisation problems are categorised as NP-hard according to Tasan and Tunali [2008]. In recent years, attempts made to effectively solve the optimisation problems have led to significant development, but there is no universal method in this regard. As a result, there is a vested interest in approximation algorithms which are able to find near-optimal solutions over a reasonable computational time.

Nonetheless, the complete solution space cannot efficiently be explored by the heuristics based on local search algorithms and there is the possibility of being trapped in the local optimum. Using a metaheuristic or repairing strategy, we can

remove this limitation (Scholl and Becker [2006]).

Constantly, constructive heuristics produce a solution that is improved by presenting a complete answer and then making efforts to achieve a better result. Overall, in local search algorithms or improvement procedures, attempt is made that a certain feasible solution is improved through iteratively transforming it into another feasible solution.

From the viewpoint of Voss et al. [2009], by applying robust tools presenting high-quality solutions to significant problems in the field of business, engineering and science in reasonable time horizons, metaheuristics can help managers in decision-making. While these applications find exact solutions, they are still considered a real challenge despite the effect of new computer technology developments and it seems that the great interactions between computer science, operations research and mathematics are the methods of choice in many applications. Metaheuristics can easily be implemented and enjoy great generality and flexibility. But contrary to most heuristics, they are likely to escape local optima. Multiple metaheuristics have been employed, some of which will be introduced in the following. Below are some of the generic references regarding this topic:Reeves [1993], Blum and Rol [2003], Talbi [2009]. Matheuristics include a combination or hybridisation of exact methods and metaheuristics and have been examined in studies recently (e.g., Maniezzo et al. [2009]).

Another division of metaheuristic methods was made by Battaï and Dolgu [2013]: Neighbourhood methods like TS, GRASP, Simulated Annealing (SA) and Variable Neighbourhood Search (VNS); Evolutionary Algorithms (EA) such as GA, differential evolution methods and Imperialist Competitive Algorithm (ICA) and swarm intelligence based metaheuristics such as Particle Swarm Optimisation (PSO) algorithm, Ant Colony Optimisation (ACO) and Bee Algorithm (BA), Hybridised Genetic Algorithms (HGA) and Beam Search (BS). The above methods have been provided in Figure 3.4. Below, the most applied methods are introduced:

Tabu Search (TS): TS is a kind of metaheuristic deterministic search technique which has been developed with the aim of escaping local optima. As stated

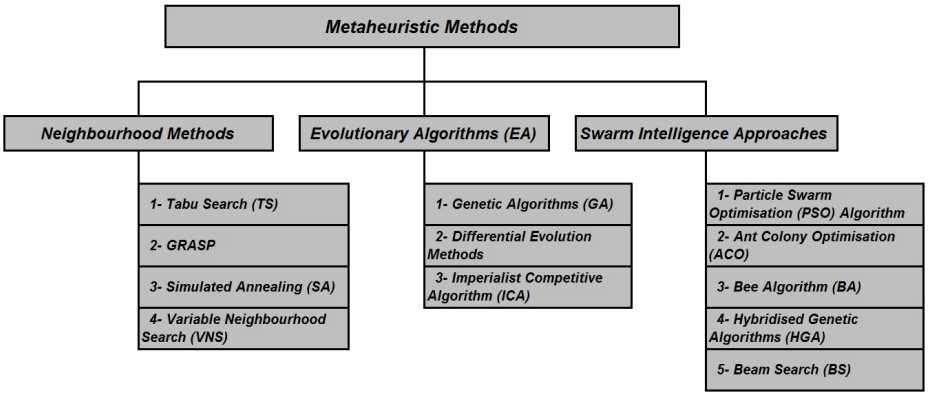

Figure 3.4: Different Category of Metaheuristic Methods.

by Michalewicz and Fogel [2013], it uses local search methods for mathematical optimisation and moves among the solutions while the next solution is always worse than the previous solution according to Pinedo and Chao [1999]. On the whole, there is a simple idea behind TS. There is a *memory* making the search explore new fields of the search space. The solutions investigated recently can be memorised and these are tabu (forbidden) points that should be avoided in making decisions concerning the selection of the next solution (Michalewicz and Fogel [2013]).

Simulated Annealing (SA): This method has been designed with the aim of escaping local optima. Its other names are Monte Carlo annealing, statistical cooling, probabilistic hill-climbing and probabilistic exchange algorithm. In each stage, the SA heuristic takes a point from a neighbourhood and selects it if it is better; otherwise, its chooses a point with some probability. Finally, these probabilities cause the system to move to the lower energy state. This stage is repeated until the system arrives at a state which is suitable for the application, or a specific computation budget has been completely used up (Michalewicz and Fogel [2013]).

Variable Neighbourhood Search (VNS): Mladenović and Hansen [1997] introduced VNS which is a metaheuristic employing a systematic change of neighbourhoods to examine various areas of the search space. VNS applies a standard local search and a stochastic shaking phase articulated with neighbourhood changes for avoiding being entrapped in local optima. Blum and Rol [2003] maintain that

VNS "is very general and many degrees of freedom exist for designing variants and particular instantiations". In Talbi [2009] opinion, VND or Variable Neighbourhood Descent may be viewed as a deterministic version of VNS, which employs successive neighbourhoods in descent to a local optimum. Its application is advantageous since different neighbourhoods may not have the same local optimum. In addition, and to the extent that VND is one of the methods used in this work, more details will be provided in Chapters 4 and 5.

GRASP: Based on the statement by Alba and Martí [2006], GRASP or Greedy Randomised Adaptive Search Procedure is mostly used in the applications because it is simple and effective. According to Andres et al. [2008], studies demonstrate that this method produces solutions with good quality for difficult combinatorial optimisation problems, specially the set covering problems, the set packing problems and the node packing problems. GRASP consists of two steps; namely, developing a solution and improving it. These two steps are repeated as prescribed. As observed by Bautista et al. [2016], GRASP is a multi-start based metaheuristic in which a feasible solution is initially constructed through a non-deterministic greedy procedure and in the improvement stage of the solutions, a given neighbourhood is examined until it is found a local optimal. The consecutive application of both stages is called iteration.

Genetic Algorithms (GA): Holland [1975] first introduced GA in the 1970s and, as stated by Tasan and Tunali [2008], became known among approximation algorithms for the ability to find near optimal solutions to big optimisation problems. GA is a stochastic search method which has been motivated by the concepts developed by Darwinian evolution theory and is related to a class of metaheuristic methods recognised as EA. GA is a solution approach with two advantages: First, GA, instead of a single point, searches a population, leading to the increased likelihood that the algorithm will not be trapped in a local optimum because a lot of solutions are considered concurrently, and second, according to Tasan and Tunali [2008], GA fitness function may assume any form and multiple fitness functions can be applied at the same time. More elucidations about GA are provided in

Chapters 5 and 6, since they are used as a solution procedure in this research work.

Particle Swarm Optimisation (PSO): PSO, as argued by Aydoğan et al. [2016], is a population-based stochastic optimisation technique introduced by Eberhart and Kennedy [1995]. PSO aims to improve the sharing of social information among particles within a swarm. Each individual in a PSO algorithm is called a particle, and each particle moves around in a multi-dimensional search space. Its velocity is continually updated by the particle's and its neighbours' experience. Zhang et al. [2015] suggest that PSO is currently one of the most widely applied optimisation techniques, such as automation control systems, electrical and electronic engineering, communication theory, OR, mechanical engineering, medicine, fuel and energy, chemistry and biology.

Ant Colony Optimisation (ACO): Blum [2004] states that the ACO was formed by Marco Dorigo and its associates in the early 1990's. These algorithms were developed as the result of the observation of ant colonies. Ants are social insects and are capable of finding good paths between their nest and a source of food. They do this by following pheromone trails that other ants have produced. With more intense trails, the probability of following it by another ant becomes greater and the next ant reinforces the trail by its own pheromone. Therefore, the activities are coordinated by the colony of ants while they do not have direct communication with each other. During this process, the information (or pheromone trails) released by another ant and then other existing information about the problem are considered by other ants. Finally, as a result of the indirect communication between the ants, suitable solutions are produced.

Bee Algorithm (BA): This kind of algorithm has been developed based on honey bees' behaviour that is viewed in the category of artificial intelligence. The current solution is selected from the best improvement of a bee's behaviour (Panigrahi et al. [2011]). BA, according to Tapkan et al. [2016], basically conducts a type of neighbourhood search together with a random search. This algorithm can be employed for both function and combinatorial optimisation. Bee algorithm has been used for several optimisation problems through its initial developers, includ-

ing: training neural networks to recognise patterns, planning jobs for a production machine, clustering of data and multi-objective optimisation. Mayteekrieangkrai and Wongthatsanekorn [2015] argue that BA is an optimisation algorithm formed based on honeybees' behaviour. Using waggle dance, they try to find a source of food in the best way. In an artificial bee colony, bees include two groups of scout bees and employed bees. The first group, i.e. scout bees, has the responsibility of finding new sources of food. These bees move around the hive on a random basis. When the bees return, they go to the dance floor if they have found a new source of food and start the waggle dance. During the waggle dance, they communicate the information on the food source to the employed bees. The latter bees join to exploit the newly found food source. Using this method, the improvement is made.

Hybridised Genetic Algorithms (HGA): GA, as previously mentioned, is applied to solve numerous combinatorial optimisation problems. Various factors can influence GA performance, including genetic operators, parameters and strategies. As a result and according to Yugay et al. [2008], many GA adaptations exist that modify the mentioned factors, resulting in various HGA. As maintained by El-Mihoub et al. [2006], another search method is employed by a HGA as a local search tool. These issues consist of various approaches to employ local search information and different mechanisms to achieve a balance between a global GA and a local search method.

Beam Search (BS): As mentioned by Ow and Morton [1988], beam search was first introduced in the mid 1970s and Lowerre was the first individual to use it in 1976. It is a fast approximate B&B method and, according to Mete et al. [2016] is structurally an adaptation of the B&B method. B&B method, through back tracking technique, examines all the existing solutions. On the contrary, BS algorithm searches a solution just on determined points with forward in order to raise the speed of the solution search. On the whole, with the control of the B&B tree growth, the BS paradigm modifies the B&B search process.

The methods discussed above are mostly employed in connection with the problem addressed in this thesis. This part dealt with the description of different prob-

lems and solution approaches and the most widely used solution approaches were provided. Nevertheless, if the studied problem turned to the multi-objective, EA normally offered better answers. Hence, EA application to solve problems with this nature has been inspired by the population-based nature of EA which paves the way for the generation of several elements of the pareto-optimal specified in a single run. In the following, some notes and solution approaches for MOP will be presented.

3.1.3 Multi-Objective Problems (MOP)

Today, MOP are attacked by computer scientists, biologists, engineers and operations researchers using EA. The MOP which is also known as multicriteria optimisation, multiperformance or vector optimisation problems is defined by (Osyczka [1985] as the problem of finding a vector of decision variables which observes constraints and optimises a vector function whose elements are representative of the objective functions. These functions mathematically describe performance criteria typically in conflict with each other. Therefore, the word "optimise" means to find a solution that offer the values of all the objective functions which are acceptable to the decision-maker.

Coello et al. [2007] state that MOP are not newly recognised problems. Indeed, several writers (e.g. Stadler [1987]) suggest that multi-objective optimisation is an inherent part of economic equilibrium. Consequently, it can be traced back to the year 1776 when Adam Smith's treatise called *The Wealth of Nations* was published. In order to solve MOP, there are various methods available.

Generally, EA can be described by following three facts:

1. A set of solution candidates is maintained;
2. Solutions go through a selection process;
3. The set of solutions is handled using genetic operators, usually recombination and mutation;

As suggested by Zitzler [1999], main issues in multi-objective search are as follows: The first issue is how to achieve fitness assignment and selection, respectively,

in order to direct the search towards the pareto-optimal set. The second issue is how to develop a diverse population for preventing premature convergence and gaining a well-distributed and well-spread non-dominated set. Below is the name of the most applied methods in this area as mentioned by Zitzler [1999].

1. Vector Evaluated Genetic Algorithms (VEGA) which is based on the Multi-Objective Genetic Algorithms (MOGA);
2. Weighting-based Genetic Algorithms which was introduced in (Hajela and Lin [1992]), so called HLGA—Hajela and Lin's Genetic Algorithms;
3. Fonseca and Fleming's MOGA (FFGA);
4. Niched-Pareto Genetic Algorithms (NPGA);
5. Non-dominated Sorting Genetic Algorithms (NSGA) and NSGA-II;
6. Strength Pareto Evolutionary Algorithms (SPEA);

Further, the most employed methods have been classified by Collette and Siarry [2013] based on the general concept of each procedure as the following.

1. Scalar procedures, including Keeney-Raiffa method and Hybrids methods;
2. Interactive procedures like Surrogate worth tradeoff method and Fandel method;
3. Fuzzy procedures, including Sakawa method and Reardon method;
4. Methods that employ a metaheuristic such as method based on SA, TS and GA;
5. Decision aid procedures; for example, ELECTRE I method and ELECTRE IS method;

Performance measures for each of the methods stated above are based on the following items:

1. Minimal distance to the pareto-optimal front;

2. Good distribution;

3. Maximum spread;

3.2 Balancing Problems

This section briefly reviews the related literature regarding Assembly Line Balancing Problems ($ALBP$). The practical problem of this work is also taken into consideration. However, it should be noted here that the writer only found a few publications about balancing problems in the footwear industry but none of them dealt with a similar case. The section is composed of two parts, including the classification of ALBP and solution approaches.

3.2.1 Classification of Balancing Problems

Because of the great diversity of features, assumptions, applications and solutions approaches, such problems are divided into different categories in the scientific literature.

Kriengkorakot and Pianthong [2007] maintain that the Single-model Assembly Line Balancing Problems ($SALBP$) are very common. As previously stated, only one type of product exists in the lines. The Generalised Assembly Line Balancing Problems ($GALBP$) are associated with multi-model, mixed-model and U-shape lines. A multi-model line generates different products, whereas a mixed-model line simultaneously produces various models of the same product. The U-shape ALBP considers the case of machines situated around a U-shaped line and operators with the capability of working on both sides of the U.

Based on the type of task processing time (either constant or variable), ALBP are divided into deterministic and stochastic problems by Ghosh and Gagnon [1989]. There is a classification of GALBP made by Boysen et al. [2007a] who observed that most of the studies on ALBP have focused on modelling and solving the SALBP. Figure 3.5 displays a classification of ALBP through inspecting products and processing times based on the work done by Ghosh and Gagnon [1989].

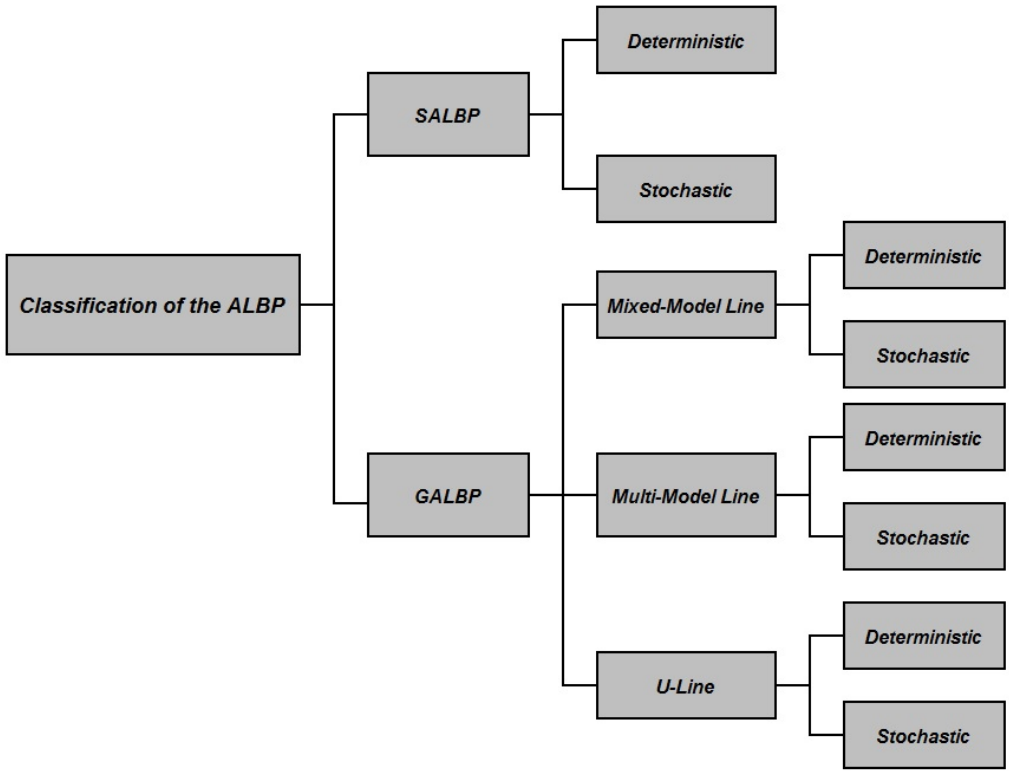

Figure 3.5: Classification of ALBP Considering the Products and Processing Times.

There is another ALBP classification with regard to the type of mechanical movement (Merengo et al. [1999]). Two distinct lines are available: mechanical and non-mechanical. Mechanical lines possess particular mechanisms for moving the work pieces and are divided into moving, paced and unpaced lines. The system moves at a constant speed in the moving line whereas in the paced line, it moves on a periodic basis and remains at each workstation for a cycle time. In an unpaced line, several buffers can be seen between the workstations, and the work pieces are removed from the buffers by the operators.

Numerous studies have been conducted on single-model lines. Below is a classification of these problems while considering objectives. This classification has been made based on the research by Becker and Scholl [2006], Tasan and Tunali [2008], Scholl and Becker [2006], Chen et al. [2012a].

1. SALBP-1: Minimising the number of workstations with respect to the cycle time;
2. SALBP-2: Minimising the cycle time for a certain number of workstations.
3. SALBP-E: Maximising the line efficiency, minimising the number of workstations and the cycle time simultaneously;
4. SALBP-F: Finding a feasible solution;
5. SALBP-3: Maximising the workload smoothness for a specific number of workstations;
6. SALBP-4: Maximising workload relatedness;
7. SALBP-5: Multiple objectives.

It should be remembered that all types of problems stated above also exist in different MALBP versions, including MALBP-1 or MALBP-2. This work's problem aims to minimise the number of workstations and smooth the workload of the operators with respect to the available time. This will be explained in the next Chapter. Hence, it is categorised as MALBP-1 and as MALBP-4.

There is another category for ALBP which has been introduced by several researchers. This category addresses at the same time the worker assignment and balancing, the so called Assembly Line Worker Assignment and Balancing Problems ($ALWABP$). As mentioned by Pereira [2018], Miralles et al. [2007] developed this type of problem for the first time. Other references comprise Moreira et al. [2012], Mutlu et al. [2013], Borba and Ritt [2014], Moreira et al. [2017] and Oksuz et al. [2017].

The next relevant references in the following paragraph encompass those associated with the footwear industry. They give us the possibility to briefly discuss and emphasise the significant differences in connection with the problem considered in this work. Most of the publications about MALBP only take workstations into consideration and do not address operators. For example, we can refer to Haq

et al. [2006], Bai et al. [2009], Su and Lu [2007], Mamun et al. [2012], Sivasankaran and Shahabudeen [2013] and Akpınar et al. [2013]. There are other studies like Özcan et al. [2010a] and Özcan et al. [2010b] which also deal with operators but as multi-skilled and flexible operators. Operators are also addressed by Vilarinho and Simaria [2006] and Simaria and Vilarinho [2009] in two-sided assembly lines and lines with parallel workstations, but here the operators are flexible. Chen et al. [2012a] and Salehi et al. [2017] have conducted studies on skilled operators, but the problem is SALBP. Ultimately, as some works associated with ALWABP, we can mention the research by Miralles et al. [2008], Blum and Miralles [2011], Sungur and Yavuz [2015] and Pereira [2018]. These studies are intended to consider the heterogeneity introduced in the lines by the differences in the operators, which can affect task times. Nonetheless, our case study still enjoys unique characteristics because of simultaneously considering operators with diverse skills, machines with various capabilities and different quantities of mixed models inside boxes.

Concerning the footwear industry, the more generic references mentioned here. Zangiacomi et al. [2004] solved a production planning and scheduling problem for mass customisation system; Chen et al. [2012b] proposed a GA on a hybrid flow shop problem to solve a scheduling problem. Research of Guimarães et al. [2014], are about the workers macro ergonomic evaluation; Chen et al. [2014a] resort to simulation to configure the layouts of stitching lines; Chen et al. [2014b] aims at solving a balancing problem by using a hybrid GA; Ulutas and Islier [2015] work on the layout problem and Dang and Pham [2016] design an assembly line by using simulation. Moreover, Quyen et al. [2017] solving a resource constrained assembly line balancing problem in a single model line.

In the end, a voluminous literature is available on ALBP, but there is only a few papers addressing the footwear applications. Besides, no similar case was observed in the literature review. There are two reasons in this regard: one because of the new dedicated assembly systems and the other is the fact that the work simultaneously consider mixed models, skilled operators and machines.

3.2.2 Solution Approaches

As previously stated, the approaches for solving ALBP are classified into two main groups: Optimum seeking or exact methods, and Approximate methods.

As maintained by Becker and Scholl [2006] and Sivasankaran and Shahabudeen [2014], most researchers have attempted to solve SALBP. Thus, there are a variety of optimisation models for this type of problem. It was reported by Boysen et al. [2008] that Salveson [1955] published the first optimisation model in this area. IP (ILP), LP, NLP, dynamic programming and B&B are the most common exact methods for solving ALBP.

Altekin et al. [2008] employed a MIP formulation and also determined lower bounds for the problem to solve it. Another definition for the ALBP indicates that the ALBP can be solved by applying the bin packing problem. Hence, the lower bound of the bin packing problem is equivalent to the ALBP lower bound. Various exact methods including Erel and Gokcen [1999] turn the problem to bin packing problem and solve it. Another MIP model was designed by Pastor [2011] to optimally solve the ALBP. Additionally, a MIP formulation was developed by Özcan et al. [2010a] for two-sided assembly line problem. Albert et al. [2008] used MIP for balancing assembly line with skilled and unskilled workers and Pinnoi and Wilhelm [1997] applied it to minimise the maximum idle times for a certain number of stations in order to balance workloads allocated to all stations. Researchers such as Gökçen and Erel [1998] and Oliveira et al. [2012] provided binary integer approaches for MALBP. Additionally, ILP was employed by Miralles et al. [2007] and Sungur and Yavuz [2015] in order to solve SALBP.

Also, the ALBP can be transferred to the shortest path problem. In this graph, each path matches a feasible solution and each shortest-path corresponds to an optimal solution. An algorithm was designed by Gutjahr and LNemhauser [1964] which is based on finding a shortest-route in a finite directed network provided to the ALBP. A balancing problem was solved by Dolgui et al. [2006] for paced production lines with workstations in series and blocks of parallel operations at the workstations. This is done based on the transformation of the problem to

a constrained shortest path one. Battaïa and Dolgui [2012] introduced an exact method to solve a GALBP. Their purpose is to decrease the size of the problem in order to solve and hence lessen the solution time needed to exactly solve the problem.

The B&B procedures are usually applied in the literature, and various ways are available to solve this problem through the application of the B&B method. In the following, some of the mentioned procedures will be provided. Hoffmann [1992] employed a quick B&B to solve SALB. Some researchers (e.g. Bukchin and Rubinovitz [2003] and A.Dolgui and I.Ihnatsenka [2004]) addressed SALBP with an equipment selection process and utilised a B&B algorithm. Pastor and Pastor and Ferrer [2009] introduced an improved mathematical program. This method belongs to the B&B technique with LP relaxation. Other researchers including Johnson [1983] and Sewell and Jacobson [2012] took advantage of B&B on the ALBP.

The B&C procedure is also utilised in several papers and this method is associated with the vertical balance. Computational results obtained in the study by Pinnoi and Wilhelm [1998] reveal that the B&C is a useful and powerful approach to solve assembly system design problems. Pinnoi and Wilhelm [1997] exploited B&C for the purpose of workload smoothing on the assembly lines. The obtained result indicates considerable improvement in running times.

According to Bard [1989], dynamic programming is employed to solve complex problems and divide them into a set of simpler sub-problems. It is also applied by Nicosia et al. [2002] for the problem of assigning operations to an ordered sequence of non-identical workstations, observing precedence relationships and cycle time limitations. Carraway [1989] addressed the problem of minimising the number of workstations required on an assembly line for a specific cycle time and made dynamic programming formulation for solving the balancing problem. Baybars [1986] has stated that when the number of tasks is great, all exact algorithms fail, meaning that the CPU time increases quickly.

The approximate methods are associated with heuristic, metaheuristic, whose goal are to obtain reasonable solutions over a short time. They do not guarantee the

achievement of an optimal solution. MALBPs are NP-hard optimisation problems. Therefore, it is expected that the approximate methods are used in large-scale cases. Two different priority rules exist which are based on constructive procedures for ALBP, namely station-oriented procedures and task-oriented procedures, which have been discussed in the work by Scholl and Becker [2006].

Workstation-oriented procedures: start with the first workstation and the following workstations are successively taken into account. In every iteration, a task with the greatest priority is assigned to the current workstation until it is maximally loaded. A station being loaded maximally becomes closed but the next workstation is opened. In the research carried out by Scholl and Voß [1997], a heuristic method is employed for SALBP-1 and SALBP-2 and the general attitude is workstation-oriented and an improvement procedure is performed with TS.

Task-oriented procedures: Out of the existing tasks, a task with the greatest priority is selected and assigned to the first workstation to which it can be designated. This procedure is applied more than the previous one and is repeated over and over and when all available tasks are assigned, it stops.

Multiple manufacturers are changing their production lines from single product or batch production to mixed-model production. Various models of a product are generated on the same line in the mixed-model production. In this way, manufacturers can present a variety of products to their customers in a timely and cost-effective manner (Sparling and Miltenburg [1998]). The MALBP is based on the same fundamental assumptions as the SALBP. One of the basic approaches that can be used to model and solve the MALBP is reduction to single-model problem (Becker and Scholl [2006]). By considering the assumption that similar tasks should be designated to the same station for all models, the mixed-model problem is divided into P independent SALBP instances.

With regard to to Figure 3.4 and as stated by Battaï and Dolgu [2013], the approximate methods to solve ALBP can be categorised into neighbourhood methods like TS, GRASP, SA and VNS; evolutionary approaches, such as GA, differential evolution methods and ICA; and swarm intelligence based metaheuristics such as

PSO algorithms, ACO and BA, HGA and BS. In the following, the most applied approaches in this field will be presented:

Tabu Search: Some scientists applied TS. However, developing TS for SALBP with the objective of minimising the number of workstations (SALBP-1) is more than SALBP with the objective of minimising the cycle time (SALBP-2), but an elementary TS procedure is raised in the study by Heinric [1994] for SALBP-2. In the research conducted by Heinric [1994], the researcher examines and shows the similarities and the performance of SA and TS. A variety of modern search methods utilised for combinatorial problems possess a common structure. The accomplishment of both methods is displayed for the ALBP and their performance is compared. The problem set comprises famous literature problems. In this comparison, TS becomes superior to other methods and is more successful.

Chiang [1998] also employed TS; four varied versions of algorithms have been built and the obtained results demonstrate that the TS always finds optimal solutions with some exceptions. This reinforces the belief that the TS can be a very promising and useful instrument to solve problems in the field of manufacturing.

Lapierre et al. [2006] provided another TS algorithm in which its performance on the SALBP-1 was assessed, and the algorithm was examined on a real set of industrial data. Computational results concerning the benchmark instances and real world problems from industry revealed that the method is effective when it is compared with other methods.

TS was also utilised by Özcan and Toklui [2009] for a two-sided assembly line. Further, Özcan et al. [2009] chose it for conducting a research on parallel ALBP; and Lapierre et al. [2006] referred to the differences between their new TS algorithm and other algorithms in the literature. Buyukozkan et al. [2016] made a TS named lexicographic bottleneck ALBP for a new type of problem. They compared the obtained results with the Bee Colony Optimisation (BCO) algorithm mentioned previously.

Simulated Annealing: Suresh and Sahu [1994] addressed the ALBP with stochastic task processing times. Their paper provides an approach to solve the

problem of the SA technique application. The suggested approach is intended to achieve the global optimum through not getting trapped at various local optimum points.

SA was applied by Vilarinho and Simaria [2002] and Özcan et al. [2010a] for MALBP with parallel workstations. Cakir et al. [2011] investigated a hybridised SA (HSA) in order to solve a multi-objective optimisation problem with stochastic times approach. The objectives are as follows: First, the minimisation of the smoothness index and second, the minimisation of the design cost. A multinomial probability mass function was put into effect in the proposed SA so that multi-objectives are taken into consideration and the decision-maker is able to assess a large number of alternative solutions. The achieved result suggests that this algorithm can be applied for multi-objective single-model / mixed-model, U-shaped deterministic / stochastic ALBP. Roshani et al. [2012] also employed SA approach for a two-sided line. In their study, the proposed SA approach creates balancing solutions for the problem with respect to task priority.

Variable Neighbourhood Search: Polat et al. [2016] raised a two-step VNS and VND algorithm for minimising the cycle time and assigning workers to workstations. It finds optimal or near optimal solutions. The obtained results demonstrate that the algorithm is more effective and powerful relative to the methods existing in the literature. As mentioned by Kellegöz [2016], VNS is utilised for multi-objective ALBP with multi-manned stations; literature instances are applied to compare the VNS performance.

VNS and VND were not used as much as the other methods, in the resolution of ALBP. This is a suitable opportunity for a practical assessment. On the other hand, VND seems to provide a good potential for diversification through different neighbourhood structures based on the type of problem. The recent work done by Polat et al. [2016] which has previously been mentioned also affected out selection of a VND-based solution method. The simple selection of a metaheuristic and its efficient implementation is not a precise decision-making process. But, as maintained by Ferreira [2013], multimethodology in metaheuristics can also be helpful

in addressing problem solving. In Chapter 4, greater details are provided about VND to the extent that it will be applied to solve the MALBP.

Genetic Algorithms: In connection with ALBP, GAs were usually used according to Tasan and Tunali [2008]. For example, we can refer to the studies by Kim et al. [1996], Simaria and Vilarinho [2004] and Sivasankaran and Shahabudeen [2013]. These researchers looked for a set of pareto-optimal solutions for MOP. The works of Simaria and Vilarinho [2004] and Sivasankaran and Shahabudeen [2013] are focused on MALBP.

Falkenauer and Delchambre [1992] introduced an effective GA for SALBP-1. They showed that the algorithm has the ability to tackle real world data sizes. Another study in which GA for solving SALBP-2 has been examined is related to Anderson and Ferris [1994]. The writers aimed to indicate the efficient use of GA to solve combinatorial optimisation problems. Additionally, in their study, Stockton et al. [2004] investigated GA application to the SALBP-1. Their research makes a comparison between GA performance and a traditional solution method. In the study by Kim et al. [2009], the writers applied the GA for the two-sided assembly line. It is a kind of production line in which tasks are carried out in parallel on both sides of the line. Kim et al. [2009] have provided a mathematical model and a GA for the SALBP-2 problem. Mutlu et al. [2013] employed it for SALBP and addressed worker assignment process and the problem under investigation is ALWABP-2.

A Grouping GA was utilised to solve grouping optimisation problems. Here, it is aimed to group members of a set into a small number of families for the purpose of optimising objective functions under certain constraints (seeBrown and Sumichrast [2005]). In their paper, Chen et al. [2012a] intend to maximise the workload smoothness and it is a type of SALBP-3. Their work creates a grouping GA for ALBP of sewing lines with various levels of labour skill in the garment industry.

Haq et al. [2006] suggested a HGA, aimed at minimising the number of workstations of a MALBP. It applied the solution from the modified ranked positional method for the initial solution for decreasing the search space in the global space,

leading to reduced search time. A HGA was employed by Quyen et al. [2017] for solving resource constrained ALBP in a footwear industry. In their research, the desired solution is selected with regard to the priority rule based method and GA.

GA and Simulation were combined for the MALBP by Su and Lu [2007]. Their study aimed to smooth the workload balance in each workstation. In their study, Mamun et al. [2012] introduced a modified procedure for solving MALBP-1 while applying GA framework. The standard encoding scheme directly designates the tasks to the workstations in a balancing solution. The overall number of workstations is minimised by the proposed model. This gives the user the possibility to control the replication process.

Particle Swarm Optimisation: It has previously been stated that PSO is an evolutionary meta-heuristic motivated by the swarming behaviour seen in flocks of birds. It was applied by Nearchou [2011] on a SALBP in order to maximise the production rate and the workload smoothing. Some researchers, including Jiansha et al. [2009] and Jianping et al. [2011], took advantage of Hybridised PSO ($HPSO$). In the first study, the proposed problem was mathematically described and in the second research, a new permutation encoding method was created in order to make the particle correspond to a feasible operation sequence which observes the precedence constrains of assembly tasks.

Further, a PSO was developed by Chutima and Chimklai [2012] with the aim of solving a multi-objective two-sided MALBP. Also, Hamta et al. [2013] utilised it for multi-objective optimisation of a SALBP in which the operation times of tasks are unknown variables and the only known information available includes the lower and upper bounds for operation time of every task.

A modified approach was introduced by Delice et al. [2014] for solving MALBP by PSO. This approach integrates selection mechanism and decoding procedure. The new procedures strengthen the solution capability of the algorithm and enable it to effectively search at various points of the solution space.

Ant Colony Optimisation: In their study, Bautista and Pereira [2002] deal with the SALBP-2. The problem applied ACO metaheuristics with different charac-

teristics and achieved good results. Then, an adaptation of the previous execution is applied to solve a real case problem observed in a bike assembly line with a hierarchical multi-objective function. The research by McMullen and Tarasewich [2003] provides an approach which is based on ACO techniques to efficiently examine the ALBP with the complicating factors of parallel workstations, stochastic task durations, and mixed-models.

In their work, Vilarinho and Simaria [2006] described an ACO algorithm for balancing MALBP. The intended algorithm accounts for zoning constraints and parallel workstations and its goal is to minimise the number of operators in the assembly line for a specific cycle time. Besides, a flexible agent-based ACO algorithm was provided by Kucukkoc and Zhang [2016] in order to solve a two-sided MALBP. The obtained results demonstrate that the TS algorithm outperforms six other heuristics applied.

Sabuncuoglu et al. [2009] employed another ACO optimisation for the SALBP, U-shaped line. The writers carried out an experimental study on a large scale and compared the performance of the proposed algorithm with best known algorithms available in the literature. Moreover, ACO algorithms were applied by Simaria and Vilarinho [2009] and Akpınar et al. [2013] with the aim of minimising the number of workstations and smoothing workloads, in MALBP.

Bee Colony Optimisation (BCO): A two-sided assembly line was addressed by Tang et al. [2016]. They utilised a BCO with the purpose of minimising the cycle time, given a number of workstations. In addition, Oksuz et al. [2017] provided BCO and GA for maximising the efficiency of the line in a MALBP.

In Tables 3.1 and 3.2.2, a summary of the more relevant literature has been presented. In the mentioned tables, Ws represents Workstation, Smoothing Workload Balances has been displayed by $SWBs$, CT stands for Cycle Time and LE indicates Line Efficiency.

Table 3.1: Summary of More Relevant Literature to the Balancing.

Publications	Problems Structures	Objectives	Solving Approaches	Test Problems
Gökçen and Erel [1998]	MALBP	Min N.er Ws	Binary IP	Generated
Simaria and Vilarinho [2004]	MALBP	Min CT + SWBs	GA	Generated
Haq et al. [2006]	MALBP	Min N.er of Ws	HGA	Generated
Vilarinho and Simaria [2006]	MALBP	Min N.er Ws + SWBs	ACO	Generated
Miralles et al. [2007]	SALBP	Min CT	ILP	Case Study
Su and Lu [2007]	MALBP	SWBs then Min CT	GA	Generated
Miralles et al. [2008]	SALBP	Min CT	B&B	Generated
Bai et al. [2009]	MALBP	Min N.er Ws + Max LE	GA joined with SA	Generated
Simaria and Vilarinho [2009]	MALBP	Min N.er Ws + SWBs	ACO	Benchmarked
Özcan et al. [2010b]	MALBP	Min N.er Ws	TS	Generated
Özcan et al. [2010a]	MALBP	Min N.er Ws + SWBs	SA	Benchmarked
Blum and Miralles [2011]	SALBP	Min CT	BS	Benchmarked
Mamun et al. [2012]	MALBP	Min N.er Ws + SWBs	GA	Benchmarked

Publications	Problems Structures	Objectives	Solving Approaches	Test Problems
Moreira et al. [2012]	SALBP	Min CT	HGA + Heuristic	Benchmarked
Chen et al. [2012a]	SALBP	SWBs	GA	Case Study
Sivasankaran and Shahabudeen [2013]	MALBP	Min N.er Ws	GA	Benchmarked
Akpınar et al. [2013]	MALBP	Min N.er Ws + SWBs	ACO joined with GA	Benchmarked
Mutlu et al. [2013]	SALBP	Min CT	Iterative GA	Benchmarked
Borba and Ritt [2014]	SALBP	Min CT	Heuristic + B&B	Benchmarked
Sungur and Yavuz [2015]	SALBP	Min Cost	ILP	Generated
Polat et al. [2016]	SALBP	Min CT	VNS + VND	Case Study
Moreira et al. [2017]	MALBP	Min CT + SWBs	Heuristic	Benchmarked
Oksuz et al. [2017]	MALBP	Max LE	BCO + GA	Case Study
Salehi et al. [2017]	SALBP	1- Min N.er of Ws; 2- Min Cost; 3- Min total Disqualification Rate	Fuzzy Programming	Case Study
Pereira [2018]	SALBP	Min CT	Exact and Heuristic	Benchmarked

3.3 Sequencing Problems

Sequencing problems form a basic research area of OR because of their relevance to real world applications and their great complexity. A sequencing problem is aimed at constructing an order of tasks in defined resources so that one or several objectives are optimised. Resources and tasks in a grouping can be representative of a lot of different things. For example, resources may include workstations, machines and operators, while tasks are the operations. Techniques of mathematical programming and approximate methods, including those discussed in the review of literature, are applied to solve them. This includes the assignment of sequences of tasks to specific workstations while optimising some functions. The researchers of this field usually apply the terms *Sequencing and Scheduling* interchangeably. This may also occur in this thesis particularly when referring to other studies.

Different types of scheduling are available with regard to the resources and tasks. They are displayed in Figure 3.6.

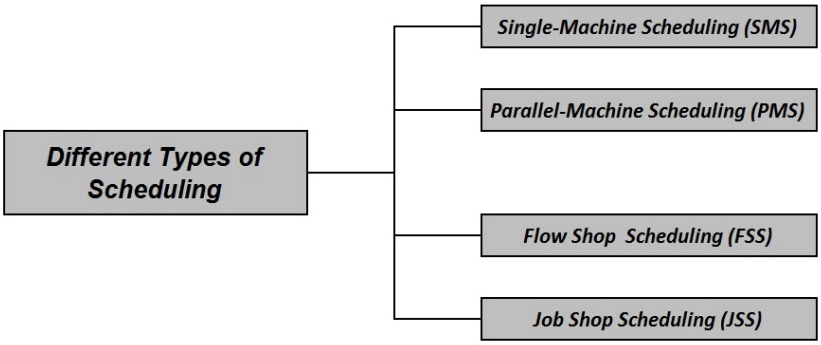

Figure 3.6: Different Types of Scheduling.

Single Machine Scheduling (SMS): As observed by Biskup [1999], the SMS is the process of designating a set of tasks to a single machine or resource so that the sum of flow-times of all tasks is reduced to the smallest possible amount.

Parallel Machine Scheduling (PMS): We can see n tasks and m machines in the classical PMS problem. Each task should be done on one of the machine. Thus, the goal here is to create a schedule that minimises a defined performance measure (Mokotoff [2001]).

Flow Shop Scheduling (FSS): Pinedo [2016] argues that there are m machines in series and each task should proceed on each of the m machines. A similar route should be followed by all tasks.

Job Shop Scheduling (JSS): In a JSS, if we have m machines available, then each job or task possesses its own route to follow. This route is predetermined. Pinedo [2016] state that there are various types of JSS. Each task can visit a machine once or more than once, whose name is recirculation. Here, it is intended to achieve a schedule so that the tasks are allocated to time intervals on the machines with the minimum duration needed for completing all tasks, according to Baker [1974],

Flexible Job Shop Scheduling (FJSS): The Flexible Job Shop Scheduling Problems ($FJSSP$) is indeed a generalisation of the classical JSS Problems ($JSSP$), in which jobs or tasks are processed on a machine from among a set of the machines available. Also, FJSSP is more difficult than the classical JSSP. The reason is that FJSSP introduces a further decision level in addition to the sequencing one. According to Pezzella et al. [2008], by specifying the task routes, it means to make decisions about what machine must process a given task among the available ones.

With respect to these categories, the case examined in this work is closer to the JSSP. In these conditions, jobs are called tasks with different routings because they come from different models. Besides, workstations which are a combination of operator and machine are taken into account instead of machines. In connection with JSSP, there are some features and constraints, the most important of which has been displayed in Figure 3.7, according to Pinedo and Chao [1999].

The situation under investigation is a make-to-order environment. Moreover, there are precedence constraints among tasks and operator and machine scheduling restrictions. There is another issue in this work which is about the lots, in the stitching line. The problem interfaces with scheduling of boxes with various quantities but not a pair of shoes.

Furthermore, as maintained by Xiaobo and Ohno [1994], mixed-model assembly lines are mostly applied in Just-In-Time (JIT) production systems. Based on their study, if a suitable mixed-model balance is assumed to be already achieved,

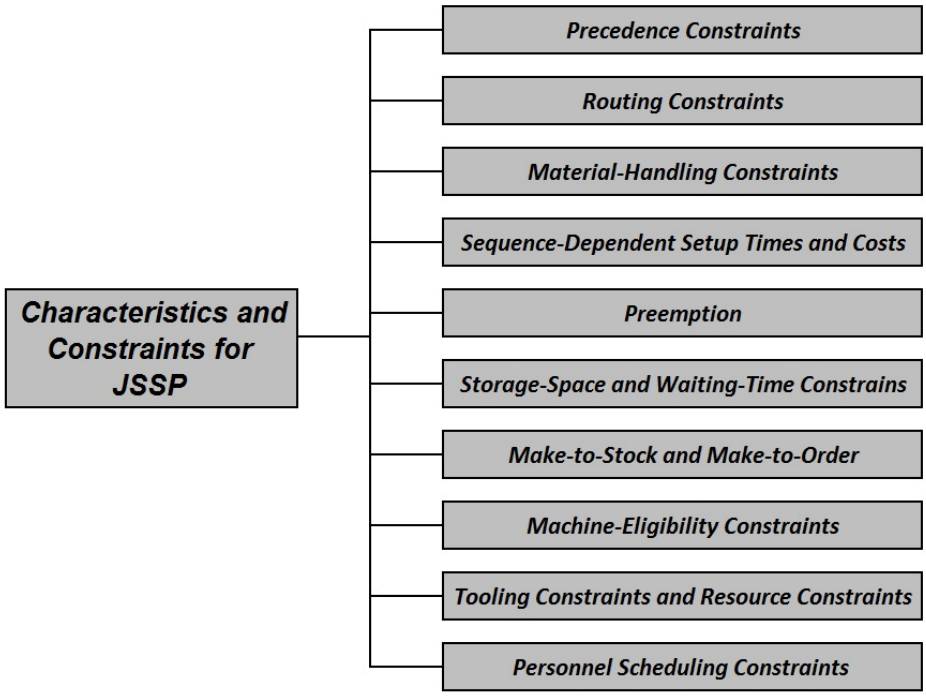

Figure 3.7: Characteristics and Constraints for JSSP.

concentrating on the sequencing problem, then the investigated problem is mixed-model sequencing. This is exactly the case in this research (MALSP- Mixed-model Assembly Line Sequencing Problems). Thus, the rest of this section will address a review of literature mostly associated with MALSP and JSSP.

In their research, Boysen et al. [2009] made a good classification for this problem (MALSP) with a tuple-notation, which takes into account the characteristics such as station boundaries, reaction on immediate work overload, setups, processing time, parallel stations, concurrent work, number of stations, launching discipline, homogeneity of stations, return velocity, line layout and objectives.

The JSSP is considered as one of the most crucial and complex problems which is known as an NP-hard. According to Zhang et al. [2013], it has been a very challenging combinatorial optimisation problem since 1950s. As mentioned by Chaudhry and Khan [2016], the most recognised performance measures of this problem have been displayed in Figure 3.8.

There are various ways to solve the given problems. Below, the most applied

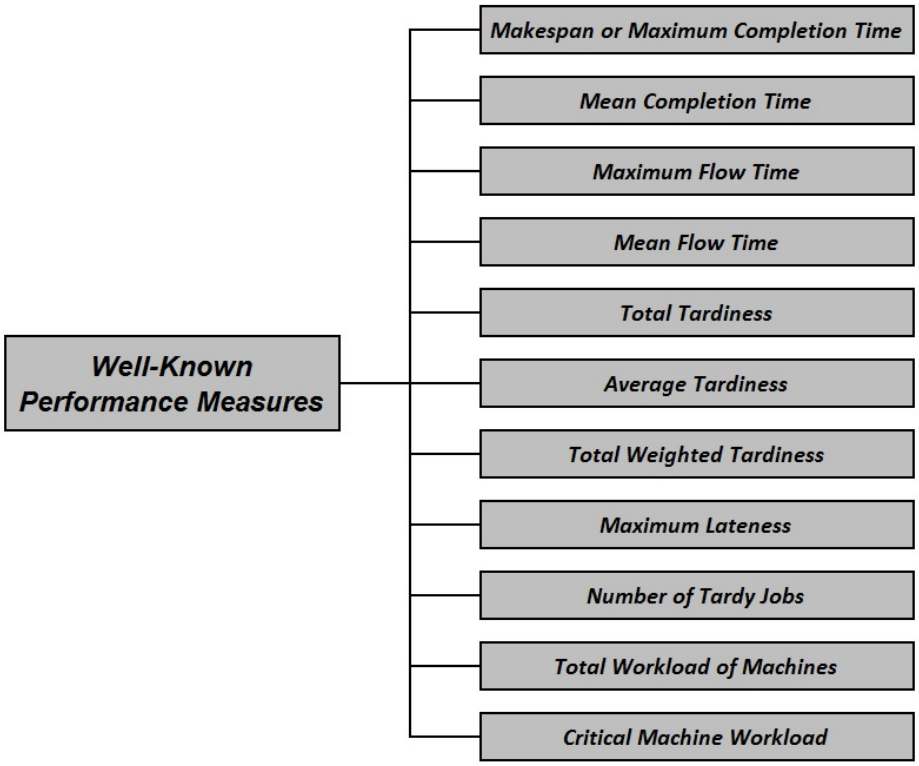

Figure 3.8: The Most Well-Known Performance Measures.

approaches are provided.

Fan and Zhang [2010] suggest that the methods to solve scheduling problems are called scheduling optimisation algorithms, which can be classified into exact and approximate algorithms. The accurate algorithm consists of analytical method and enumerative method. The approximate algorithm encompasses the following: constructive method, local search method and artificial intelligence. As stated by Fan and Zhang [2010], the analytical method comprises Moore algorithm, Johnson algorithm, and so on. The mentioned two algorithms were hardly ever studied and used in recent years. The reason is that they are only used to solve simple small-scale scheduling problems and researchers started to examine some complex scheduling problems. In relation to the enumerative method, the researchers suggested that this method is suitable only for small scale problems and cannot solve large size problems. B&B and mathematical methods are related to this category.

For example, dynamic programming was employed by Miltenburg et al. [1990] for a mixed-model JIT problem. The same problem existing in the assembly line production environment has been investigated by Ng and Mak [1994]. According to Ng and Mak [1994], the model for this problem was not linear. Also, it is difficult to solve particularly when we are dealing with a great number of products and high production quantity. In their study, a B&B algorithm is raised as a solution methodology. Additionally, a simple heuristic is applied to find an upper bound of the optimal objective function value of problem. Xiaobo and Ohno [1994] aim to minimise line stopping time through the application of a B&B method. Their paper addresses the sequencing problem of finding an optimum sequence that decreases the total line stoppage time for the mixed-model assembly line in a JIT production system. Besides, Bolat et al. [1994] employed a B&B method to schedule jobs in a mixed-model assembly line. MALSP was also considered by Bard et al. [1994]. The obtained model of their work assumes the form of a mixed integer NLP. It is solved using heuristics combined with B&B. In his study, Tsai [1995] applied the model properties and built an exact approach for a MALSP.

In the following, Chen and Dong [1999] applied an IP method for an industrial problem. Bautista et al. [2015] have provided an extension of the MALSP with the minimisation of work overload. In their study, two new optimisation models have been introduced for the given problem. But, as cited in the Bautista et al. [2015], the general approach to solve the problem is IP. Ku and Beck [2016] took advantage of MIP for JSSP and compared the methodology performance through the use of CPLEX, GUROBI and SCIP.

Dispatching Rules: Dispatching rules by prioritising all the jobs that are waiting for processing on the machines create solutions. The prioritisation may consider the jobs attributes and the machine attributes in addition to the current time. When a machine is freed, a dispatching rule examines the waiting jobs and chooses the job with the greatest priority. According to Pinedo and Chao [1999], there are numerous basic dispatching rules as shown in Figure 3.9.

Holthaus and Rajendran [1997] has introduced a dispatching rule for scheduling

	Rules	Objectives
Rules dependent on release dates and due dates	Earliest Release Date first (ERD)	Variance in throughput times
	Earliest Due Date first (EDD)	Maximum lateness
	Minimum Slack first (MS)	Maximum lateness
	Longest Processing Time first (LPT)	Load balancing over parallel machine
Rules dependent on processing times	Weighted Shortest Processing Time first (WSPT)	Weighted sum of completion times, WIP
	Critical Path (CP)	Makespan
	Largest Number of Successors (LNS)	Makespan
Miscellaneous	Service In Random Order (SIRO)	Ease of implementation
	Shortest Setup Time first (SST)	Makespan and throughput
	Least Flexible Job first (LFJ)	Makespan and throughput
	Shortest Queue at the Next Operation (SQNQ)	Machine idleness

Figure 3.9: Basic dispatching rules.

in a JSSP. These rules are created based on the additive combination of the following cases: process time, total work-content of jobs which are waited for the next operation, arrival time and slack of a job. These new rules are obtained from the combination of the other rules. Costa and Ferreira [1999] tested various dispatching rules in a mixed-model line through the application of the simulation method.

According to Fan and Zhang [2010], there are 116 papers between 2000 and 2009 which are about the dispatching rules. In the study by Chiang and Fu [2007], sixteen dispatching rules are chosen from the literature and applied as the benchmarks. Their characteristics and design concepts are also taken into account. Afterwards, a dispatching rule is suggested with the purpose of gaining good and balanced performance when more than one objective is simultaneously considered. Their paper demonstrates that the rules usually concentrate on one objective and cannot deliver good performance on multiple objectives simultaneously. In order to solve FJSSP, genetic programming and dispatching rules were combined by Tay and Ho [2008]. The writers created a genetic programming tree with the aim of building comprehensive dispatching rules. In their work, five composite rules are analysed through different performance measures, including makespan, average tardiness, average flow time and percentage of tardy jobs. The composite rules have better performance than other rules and can be easily executed. In the research by Huang

and Süer [2015], a dispatching rule based GA was employed. The method put forward addresses a JSSP with the goals of minimising makespan, average flow time, maximal tardiness and total tardiness. Heger et al. [2016] examined the studies about dispatching rules, but generally stated that they are limited to their local information and there is no rule that outperforms other rules in different objectives, scenarios and system conditions.

Below, the most applied approximate approaches for this problem are presented.

Tabu Search: According to Brandimarte [1993], TS is a useful instrument for real world application because of its generality and capability to solve complicated problems. From the perspective of Nowicki and Smutnicki [2005], exact algorithms are not able to solve instances greater than 250 operations over a reasonable time (hours, days). In their work, Nowicki and Smutnicki [2005] employed a developed approximate algorithm which is based on the big valley phenomenon. Next, a tabu list is produced and ultimately, results in a quick time are achieved.

In the investigation made by Saidi-Mehrabad and Fattah [2007], a TS is utilised to reduce the makespan. The suggested TS algorithm comprises two parts: First, a procedure searching for the best sequence of job operations and the second, a procedure seeking the best selection of machine alternatives. Randomly produced test problems are applied to assess the performance of the proposed algorithm with a suitable solution. A mathematical model is solved by the traditional optimisation technique which is B&B method, and they indicate that the proposed algorithm produces proper solutions.

Zhang et al. [2007], in their study, introduced a new neighbourhood structure and employed it to solve the JSSP through the use of TS approach. Based on their article, neighbourhood structures and move evaluation strategies have a crucial role in the effectiveness and efficiency of the TS for the JSSP.

As maintained by Zhang et al. [2008], the quality of solutions obtained by TS approach is dependent upon the initial solution. To remove this problem and present a powerful and effective methodology for the JSSP, the heuristics search approach which integrates SA and TS strategy is formed in the research conducted by Zhang

et al. [2008]. The fundamental of this approach is that in order to obtain the good solutions inside big valley, SA is applied such that TS is able to re-intensify search from the promising solutions. The researchers tested this hybrid algorithm on the standard benchmark sets and compared it to the other approaches. According to their computational results, the proposed algorithm can gain the high-quality solutions over reasonable computing times.

Li et al. [2010] performed a study and put forth an efficient Hybridised TS (HTS) algorithm for solving the FJSSP. Three minimisation objectives of the makespan, the total workload of machines and the workload of the critical machine are simultaneously taken into consideration. In their research, a TS algorithm was formed with an effective neighbourhood structure. This algorithm combines two adaptive rules and makes improved local search in the machine assignment module. A VNS algorithm is provided which integrates three insert and swap neighbourhood structures based on public critical block theory. The aim is to perform local search in the operation scheduling component.

Artificial BCO and TS are two efficient metaheuristic approaches introduced by Buyukozkan et al. [2016] for a MALSP. They evaluated the performances of both algorithms with the aid of well-known test problems and both algorithms were found to be promising solutions.

Simulated Annealing: A SA based heuristic was proposed by McMullen and Frazierss [2000]. It takes into account both setups and the stability of parts usage rates for a MALSP at the same time. The researchers compared the results with the TS obtained from the literature, then the SA offers near-optimal solutions for smaller problems.

In their research, Suresh and Mohanasundaram [2006] examined the JSSP with the goals of minimising the makespan and the mean flow time of jobs. The multi-objective optimisation problem is the simultaneous consideration of these objectives. A metaheuristic procedure is suggested to find non-dominated solution sets for the JSSP. It is based on the SA algorithm called pareto archived SA.

Loukil et al. [2007] attempted to solve a multi-objective problem in a production

scheduling through the SA approach while taking into account the makespan and the maximal tardiness. Other researchers like Zhang and Wu [2010] and Shivasankaran et al. [2015] developed HSA. The first study is JSSP with single-objective and the second study is FJSSP with multi-objective. SA was utilised by Ramalingam and Subramanian [2016] in order to solve level schedules in a mixed-model assembly line.

Variable Neighbourhood Search: Yazdani et al. [2010] presented a parallel VNS algorithm. This algorithm tries to solve the FJSSP to minimise the makespan. Application of the parallel system in the mentioned algorithm is based on the use of multiple independent searches increasing the exploration in the search space. The suggested method takes advantage of different neighbourhood structures. These structures are responsible for making changes in assignment and sequencing of operations for producing neighbouring solutions. In another work Amiri et al. [2010], used VNS while applied six types of neighbourhood structures for the FJSSP. They also did benchmark on the computational study.

Gao et al. [2008] provided a FJSSP with three objectives, minimising makespan, maximal machine workload and total workload. The solution approach includes a HGA. The GA is improved by a VND, consisting of two local search procedures. The performance of this approach is displayed by a broad computational study about 181 benchmark problems.

In the paper by Roshanaei et al. [2009], VNS solved a JSSP with setup times. Based on their article, the VNS algorithms were proved to be capable of solving scheduling problems to optimal or near-optimal schedule.

In the study by Driessel and Mönch [2011], job scheduling on the parallel machines is addressed. Arrival times of the jobs, precedence constraints, and sequence-dependent setup times are present in their paper and the solving approach applied, is a VNS approach.

The focus of the study conducted by Chen et al. [2013] is on a multi-stage and parallel-machine scheduling problem. A hybrid approach based on the VNS and PSO is created by them to achieve the near-optimal solution.

Zhang et al. [2018] took advantage of a VNS based GA to increase the search capability and to balance the intensification and diversification. The purpose of FJSSP is to minimise the maximum completion times of operations or makespan.

Genetic Algorithms: This metaheuristic is usually employed in sequencing problems. GA was utilised by Bierwirth and Mattfeld [1999] in order to reduce the mean flow-time. According to these researchers, the running time increases through increased size of the problem. By the use of GA, Chryssolouris and Subramaniam [2001] attempted to minimise the mean job tardiness and mean job costs. They compared their findings with many dispatching rules and stated that the GA is superior and preferred. In order to solve jointly the assignment and JSSP, Kacem et al. [2002] provided two approaches. The first one is that the problem of resource allocation is solved through localisation and in the second one, a multi-objective approach employs GA for solving JSSP.

Esquivel et al. [2002] solved a single- and multi-objective JSSP with a EA GA. Besides, Cochran et al. [2003] used a multi-population GA to solve a multi-objective PMS problem. Watanabe et al. [2005] conducted a study in which a modified GA was applied to minimise the makespan and the obtained results were benchmarked. MOGA approaches were provided by Mansouri [2005] and Chiang and Fu [2006] for solving a JIT MALSP (The first work) and JSSP (the second work).

The makespan of a multi-factory and multi-product environment was minimised by Chen et al. [2008] through the use of an adaptive GA. A new approach, based GA was developed by Wong et al. [2009]. This process was proposed to solve resource constrained assembly JSSP with lot streaming technique. In their work, the goal is to minimise the total lateness. On the whole, Lei [2009] carried out a study about multi-objective production scheduling. He stated that the methods involved include EA, GA, TS, SA, Genetic Programming, ACO, heuristic, local search, programming method and so on. Regrading these methods, EA and GA are selected in more than 40 papers among 90 articles.

In connection with the scheduling problem, Rebai et al. [2012] presented a new metaheuristic method in which GA, local search, and B&B Algorithm were com-

bined. According to the computational results, the deviation of the metaheuristics solutions from the optimal solution is very small. In another research, Huang et al. [2012] employed GA, adaptive learning, and heuristics integrated into a sequential GA. Hence, highly satisfactory schedules were gained over a short period of time. As observed by Gen and Lin [2014], they are varied in terms of the implementation elements and the nature of the particular problem employed. Among the EA different version of the GA has attracted a lot of attention, as maintained by Gen and Lin [2014]. Further, GA was used by May et al. [2015] and Zhou et al. [2017] for the multi-objective JSSP and multi-objective FJSSP.

Particle Swarm Optimisation: In order to solve the scheduling optimisation problem of Jerald et al. [2005] successfully executed four approaches, including GA, SA, memetic algorithm and PSO. As stated by these researchers, PSO algorithm is preferred to other algorithms and offers the minimum value of the objective function. As mentioned by Jerald et al. [2005], the procedures created in their study can properly be modified to any kind of flexible manufacturing systems with a great number of machines and components.

A HPSO algorithm was provided by Sha and Hsu [2006] for the JSSP. In their research, TS has been utilised to improve the quality of the solution. Another example of the HPSO can be seen in the study kun Lin et al. [2006]. These researchers applied two other methods called part optimisation and minute variation in the developed algorithm for the MALSP.

Rahimi-Vahed et al. [2007] conducted a study in which a multi-objective MALSP was provided. In their research, PSO was applied to solve the problem. For validating the performance of the proposed algorithm, different test problems are solved and the reliability of the proposed algorithm is compared to three distinguished multi-objective GA with regard to some comparison metrics. In another investigation, Rahimi-Vahed and Mirzaei [2007] and Mirghorbani et al. [2007] exploited two different hybrid algorithms for the MALSP. These algorithms were based on the PSO and TS. Also, PSO was used for the FJSSP in the study by Nouiri et al. [2018]. The researchers maintained that according to the experimental results, the

algorithm is highly efficient and can play a significant role in real production.

Ant Colony Optimisation: An original heuristic called ant system was provided in the study by Colorni et al. [1994]. The researchers applied it to the JSSP. In McMullen [2001] study, a production sequencing problem was solved with two objectives: minimisation of the setup times and optimisation of the stability of material usage rate. The solution is compared to other approaches including SA and TS and it is based on the ACO. As another example of applying ACO in JSSP, we can refer to the study by Blum and Sampels [2004]. Huang and Liao [2008] proposed a hybrid algorithm which combines ACO and tabu in order to minimise the makespan in the JSSP. The researchers offered a definition of a specific pheromone trail motivated by shifting bottleneck and analysed a dynamic greedy heuristic in order to improve over a straightforward ACO algorithm. This is a new method of defining the pheromone trails which can also be used in other scheduling problems.

In their study, Zhu and Zhang [2011] addressed a MALSP that reduces the variation of part usage rate. The developed ACO algorithm applies five ant search steps. The TS based local search process was added to the ants' solution construction step. Mehrabad et al. [2015], in their research, concentrated on the integrated model of JSSP and conflict-free routing of automated guided vehicles. In their paper, a MIP was developed and a ACO algorithm was used to solve the model. A mathematical model was also introduced by Zhao et al. [2018] for the multi-objective JSSP and their solving procedure is ACO.

Bee Colony Optimisation: A hybrid algorithm for the FJSSP was provided in the research performed by Li et al. [2011]. It is based on the BCO method. In their paper, the researchers stated that the experimental results concerning the benchmark instances and comparisons made with other recently published algorithms demonstrate the efficiency of the proposed algorithm. In another study done by Wang et al. [2012], an artificial BCO algorithm was provided to solve the same problem (FJSSP) with the purpose of minimising the makespan. Other references include Li et al. [2014] and Wang et al. [2017]. These researchers solved FJSSP by the use of BCO method.

A discrete FJSSP was solved in the study by lin LI et al. [2011] through a modified algorithm based on artificial BCO method. A a method for solving the scheduling problem was employed with the aim of optimising the quality of the solutions. A variable neighbourhood region searching mechanism with SA algorithm was developed in their study in order to improve the searching precision and scope.

Researchers like Chong et al. [2006], Asadzadeh [2016], Choo et al. [2016] and Sundar et al. [2017] solved a JSSP by the use of the BCO method. Asadzadeh [2016] employed the parallel BCO algorithm whereas Choo et al. [2016] applied the modified version which comprises two-enhancement scheme with neighbourhood perturbation. A hybrid BCO was also used by Sundar et al. [2017] with the aim of minimising the makespan.

Other Approaches: Dar-El [1978] argued that two basic objectives exist for the MALSP: The first objective is to find the sequence that can minimise the overall facility length and the second objective is to obtain the sequence that can minimise the throughput time. In order to solve a MALSP, Duplaga and Bragg [1998] introduced six sequencing heuristics. The sixteen different production scenarios were compared using large problems representing realistic situations.

Pinedo and Chao [1999] reported that one of the most successful and useful heuristic procedures created to minimise the makespan is the shifting bottleneck heuristic. The structure of the shifting bottleneck heuristic indicates the relationship between the bottleneck concept and more combinatorial concept including longest path and maximum lateness. In the study by Mason et al. [2002], a modified shifting bottleneck heuristic has been formed to minimise the total weighted tardiness in a semiconductor wafer fabrication facility.

Erel et al. [2007] developed six BS algorithms for the objectives of part-usage variation and load levelling performance measures. The results obtained from extensive experiments suggest that generally, two of BS are superior to the well-known heuristics on both the benchmark problems and the data sets produced in their research.

In the research performed by Rahimi-Vahed and Mirzaei [2007], the following

three objectives were taken into consideration: the minimisation of total utility work, total production rate variation and total setup cost. Because of the complexity of the problem, finding an optimum solution for this type of problems by means of traditional methods is very difficult. Thus, a hybrid multi-objective algorithm based on shuffled frog-leaping algorithm and bacteria optimisation are utilised. The researchers then compare the performance of the proposed hybrid algorithm with three well-known GA.

The MALSP was also considered by Jayaprakash et al. [2015]. The aim is to maximise the percentage of utilisation and minimise the makespan to enhance productivity in an assembly line. In their investigation, four different methods of line sequencing have been taken into account and then the simulation is applied to choose the best method.

In the study by Mosadegh et al. [2017], the MALSP was modelled as a shortest path problem and a new heuristic motivated by Dijkstra's algorithm was created to deal with it. Based on the numerical results, the performance of the proposed method is better than other algorithms.

As maintained by Rauf et al. [2017], assembly line balancing and model sequencing are two main problems. MALB addresses the assignment of task to the stations. MALSP considers the determination of the order in which the product should be assembled and in this way, the objectives are optimised. In their research, MALSP has been examined and a multi-objective optimisation, with the objectives of flow time and makespan, is taken into account. To this end, a NEH (Nawaz, Enscore and Ham) algorithm was integrated with $PROMETHEE - II$ (Preference Ranking Organisation METHod for Enrichment of Evaluations) which is a multi-criteria decision-making technique.

Fattahi and Askari [2018] solved a MALSP with stochastic operation time by the use of a multi-objective harmony search algorithm. Afterwards, it is compared with NSGA-II to assess the performance of the proposed algorithm which is a strong and well-known algorithm in this field. NSGA-II is a method to solve the scheduling problem. Ultimately, numerical examples are provided to compare these two

algorithms with some comparing metrics.

3.4 Simultaneous Balancing and Sequencing Problems

Balancing and sequencing are considered as two short-term planning issues which are of great importance. Here, the balancing aim is to specify an allocation of assembly tasks for a blend of products among the assembly stations with limited work space in order to balance the station workloads. On the contrary, the sequencing goal includes the establishment of the detailed schedule and timing of all assembly tasks for each single product in order to increase the line's efficiency, which may be defined in terms of any objectives such as throughput, makespan, etc according to Pinedo and Chao [1999]. Many researches have separately or sequentially examined balancing and sequencing. Nevertheless, the balancing and sequencing decisions in a simultaneous approach are made together or try to optimise both simultaneously which is a difficult and challenging work. As maintained by Karabatı and Saıin [2003], in a mixed-model assembly line, the workload at a workstation is influenced by the mixed model operations schedule. Hence, for having an efficient line management, the task assignment and sequencing problems should be considered simultaneously.

In the study conducted by Thomopoulos [1967], it was revealed that single-model line balancing techniques can be adapted to mixed-model schedules. In this regard, a case study from the automotive industry was implemented and a daily production plan was taken into account.

In the research performed by Sawik [2002], a two level balancing and scheduling and then a simultaneous procedure were used for a Flexible Assembly Line (FAL). Here, it is intended to specify an assignment of tasks to workstations and schedule products in order to satisfy the production plan in least possible time. In his work, MIP models are employed as the solving procedure. In a study carried out by Battin et al. [2009], a balancing-sequencing approach was introduced. This approach intends to optimise the assembly line performance. The desired method in their research is step-by-step B&B methodology. In their work, the objective

includes the minimisation of the idle time and overload time in workstations both in the balancing and in the sequencing problems.

In their study, Ozturk et al. [2010] provided the formulations of MIP and Constraint Programming ($CPro$) for solving a flexible MALB&SP while considering the setup times. The experiments in their work demonstrate that the MIP models are not able to solve relatively small instances, whereas the CPro approach seems more useful and bright. For solving a flexible MALB&SP, Öztürk et al. [2013a] provided another MIP model and a CPro. Furthermore, in the study by Öztürk et al. [2013b], the same problem but with station parallelism was examined and then a MIP and decomposition approaches were applied in order to solve the problem. Ultimately, the researchers stated that the CPro approach outperforms the other approaches.

Disassembly line balancing and sequencing problem discussed by Bentaha et al. [2014], the task times were considered as random variables with known probability distributions. The solution method is based on a stochastic binary programming and all instances were solved to reach optimality. In the following, we present the most applied approximate methods to solve the MALB&SP.

Simulated Annealing: In the research conducted by Kara et al. [2007a], the simultaneous problem in a U-shape line was discussed with multi-objectives. The goal is to minimise the absolute deviations of workloads in workstations, part usage rate and setup costs. A SA with a neighbourhood generation method was applied as the solving procedure. In their study,Kara et al. [2007b] investigated the same problem with the purpose of minimising the number of workstations. Again, a SA method was employed for solving the problem. A mathematical programming and a SA were utilised to solve MALB&SP of JIT production systems in the research performed by Kara [2008]. This researcher argued that it is important to have a smoothed workload distribution among workstations.

Mosadegh et al. [2012b] employed the SA in their study. They taken into account the station-dependent assembly times. The obtained results benchmarked with a co-evolutionary GA. Besides, in another work a SA was developed by Hamzadayi and Yildiz [2012] along with a priority-based GA for the MALB&SP with parallel

workstations and zoning constraints. The findings indicated that the proposed approach is capable of solving practical-sized problems with reasonable CPU time.

A SA algorithm was used by Hamzadayi and Yildiz [2013] in order to solve a U-shape MALB&SP. They minimised the number of required workstations and smoothing workloads balances. Also, a tabu list was inserted into the proposed algorithm to decrease the running times. Dong et al. [2014] examined a stochastic U-shape integer problem. Their study was intended to reduce the expectation of work overload time for a certain combination of cycle time and number of workstations. For the purpose of solving the problem, researchers applied a SA based algorithm. As observed by Nilakantan et al. [2017], the MALB&SP increasingly become popular due to the flexibility of generating customised products. In their article, the researchers developed two MIP models and a SA with new encoding, decoding and neighbourhood procedures. The first method of robotic MALB&SP was provided by Li et al. [2018]. This problem includes three sub-problems: task assignment, model sequencing and robot allocation. Additionally, a new MIP model was created in order to minimise the makespan and a SA and a co-EA procedure were exploited for solving the problem.

Evolutionary Algorithms: Kim et al. [2000a] introduced a new method which applies a co-EA. This method is able to solve the two problems of balancing and sequencing simultaneously. In this algorithm, it is necessary to enhance population diversity and search efficiency. Also, a localised interaction within and between populations is adapted. The experimental results obtained show that the proposed algorithm is better than other available approaches. Kim et al. [2000b] conducted an investigation and evaluated the simultaneous problem in a U-shape line. In their study, the researchers employed a co-EA for solving the problem.

An EA was proposed by Kim et al. [2006] in order to perform a U-shape MALB&SP. their study aimed to minimise the summation of the deviations between average workload in each cycle and the amount of work allocated to each workstation. This algorithm includes a parallel search, which increases its capability and power compared to hierarchical approaches.Mosadegh et al. [2012a] carried

out a study and attempted to minimise total utility work. They developed a MIP model in order to gain the exact solution. Moreover, in their study, an EA was applied due to the NP-hardness of the problem. In a study, the developed method and modified version of the co-evolutionary GA were compared. The efficiency of the proposed EA was statistically obtained from two aspects, the first is the quality of solutions and and the second is the running times for finding the best solution.

Genetic Algorithms: Miltenburg [2002] performed a study in which a JIT U-shape MALB&SP was examined. Their research revealed that two important problems occur routinely with this line: the first problem includes the allocation of tasks to workstations and the second problem is the choice of the sequence where models will be generated. The study introduced a GA with different crossovers and mutations. Cao and Ma [2008] employed MOGA for solving MALB&SP. In their work, the aim is to minimise the idle work time, and level part usage of sequencing.

A multi-objective problem and an amelioration structure with a GA were applied by Hwang and Katayama [2010]. The study was performed with the purpose of improving a variance of workload on mixed-model assembly line production systems. The applied method chose the priority-based coding rule to solve the U-shape MALB&SP. The actual data were obtained from a case study of a boiler manufacturer in Japan. Özcan et al. [2011] conducted a study in which a U-shape line was taken into account. They solved a MALB&SP with stochastic task times. To this end, a GA was used to solve the problem and to evaluate the efficiency of the proposed algorithm. Besides, a computational study was performed for both deterministic and stochastic versions of the problem.

Particle Swarm Optimisation: In their study, Rabbani et al. [2017] suggested a multi-objective problem with different limitations. The two approximate methods called multi-objective PSO and NSGA-II are utilised for solving the developed model. According to the results, NSGA-II performance is better than the multi-objective PSO algorithm in most criteria applied in their investigation.

Ant Colony Optimisation: Agrawal and Tiwari [2008] suggested balancing and sequencing of a U-shape line with stochastic task completion times for disas-

sembly operations. In their study, an ACO algorithm was employed for solving the problem. The researchers ran the algorithm over a series of benchmark problem instances. Kucukkoc and Zhang [2014b] conducted a study in which a parallel two-sided MALB&SP was discussed. In order to solve the problem, an agent-based ACO algorithm was suggested. Their study aims to minimise the number of workstations through reducing idle times and combining the advantages of both parallel and two-sided assembly lines. As stated by Kucukkoc and Zhang [2014b], there is a need for more powerful solution approaches. The aim is to incorporate model sequencing problem into balancing of a complex line system which enhances the complexity of the problem and needs much more CPU time. In their study, Kucukkoc and Zhang [2014a] introduced the same problem and proposed a mathematical model for it. Also, a novel agent based ACO approach was introduced as the solution method. In Their study, the researchers reviewed 15 articles associated with this simultaneous problem. It was found that EA and SA are the most used approach.

Bee Colony Optimisation: In their research, Saif et al. [2014a] considered a MALB&SP and a multi-objective BCO method. In their paper, the data were obtained from standard assembly line problems, from OR library. A multi-objective MALB&SP was examined by Saif et al. [2017]. The study was intended to minimise the variation in material usage, the maximum makespan among the multi-lines and the penalty cost of the late delivery models from various orders at the same time. Besides, a new mix-minimum part sequencing method was discussed and a multi-objective artificial BCO algorithm was suggested to find a solution for the desired problem. Standard assembly line data were obtained from OR library and were tested.

Other Approaches: Uddin et al. [2010] emphasised the important assembly line design features and presented a combined approach to solve the MALB&SP. Uddin and Lastra [2011] conducted a study and initially performed the balancing for a single model line and then used the defined rule for the mixed-model problem. Ultimately, they integrated the sequencing problem. The solving approach employed in their research is step by step.

Lian et al. [2012] conducted a study in which a U-shape MALB&SP was examined and a modified metaheuristic called the colonial competitive algorithm was applied for solving the problem. In the end, Manavizadeh et al. [2015] investigated a multi-objective MALB&SP with the following three objectives: Minimisation of the cycle times, the wastages in each workstation, and the work overload. Also, a new heuristic algorithm was applied for solving the problem.

Chapter 4

Balancing Mixed-model Assembly Systems

This Chapter provides the solution of the Mixed-model Assembly Balancing Problems ($MALBP$), considered in the thesis. An optimisation model is presented. It helps to clarify the situation under study. It has been tested and solved for small instances and programmed into the CPLEX Optimization Studio, but for large company instances it was not possible to get optimal solutions. Consequently, due to the complexity of the problems under study, we decided to resort to approximate solution methods. Two methods were devised: developed and tested.

The first is inspired by Tabu Search (TS) and named $ASBsm$- Assembly System Balancing Solution Method. It integrates a constructive heuristic and an improvement heuristic, based on TS. The second approximate method ($RPW - VNDbal$) integrates an adaptation of the Ranked Positional Weighted (RPW) method and the Variable Neighbourhood Descent (VND) metaheuristic. The adapted RPW method is used to create initial feasible solutions while preassigning special operators and machines. VND is applied to improve the quality of the solutions. The similarities and differences between the methods and their evaluations are portrayed in section 4.4. However, their detail evaluation and comparison are in Chapter 7. *Simulation* is also implemented to contribute to evaluate the solutions and their practicability.

4.1 Optimisation Model

This study contemplates the daily balance of lines. For this reason, WIP has to be taken into consideration quite often, as there are boxes in the systems with some tasks that have already been executed. Consequent adaptations of the input data (such as updating routings) are performed. For instance, if the quantities of the boxes, which are in process are summed up, they can represent around 3400 pairs of shoes.

Objective: A good line balance solution should open a reduced number of workstations, in a way that the right operators and machines are assigned to the correct tasks and the operators' workloads are balanced. This is translated into the objective function of the optimisation model. The following indices, parameters and decision variables are considered in the mixed integer optimisation model.

Indices:

j: index for model, $j \in [J]$, $[J] = \{1, 2, \ldots, J\}$

i: index for task, $i \in [I]$, $[I] = \{1, 2, \ldots, I\}$

k: index for number of boxes, $k \in [K]$, $[K] = \{1, 2, \ldots, K\}$

o: index for operator, $o \in [O]$, $[O] = \{1, 2, \ldots, O\}$

m: index for machine, $m \in [M]$, $[M] = \{1, 2, \ldots, M\}$

t: index for type of machine, $t \in [T]$, $[T] = \{1, 2, \ldots, T\}$

Input parameters:

P_{ij} = processing time of task i for model j

$OT_{oi} = 1$: if operator o can perform task i; 0: otherwise

$MTY_{mt} = 1$: if machine m is of type t; 0: otherwise

$TRM_{it} = 1$: if task i needs machine type t; 0: otherwise

Q_{jk} = quantity of model j in box k

MW = minimum number of needed workstations

Av = available time

Υ = weight in the objective function

Decision variables:

$X_{ijko} = 1$: if task i of model j of box k is assigned to operator o; 0: otherwise

$Y_{ijkmt} = 1$: if task i of model j of box k is assigned to machine m from type t; 0: otherwise

Auxiliary variables:

$Z_{om} = 1$: if operator o is assigned to machine m; 0: otherwise

W_o = workload of each operator o

$\bar{W}$ = average workload of the operators

Optimisation model:

The objective function is the minimisation of the number of workstations (the first part) and smoothing the operators' workload (the second part).

$$Minimise \left(\sum_{o=1}^{O} \sum_{m=1}^{M} Z_{om} + \frac{\Upsilon}{O} \sum_{o=1}^{O} |W_o - \bar{W}| \right)$$

Parameter Υ should be adapted in order to give a relative lower weight to the second part of the objective function. Here, Υ is selected so that the smoothing part has one less digit than the number of workstations. The value of Υ for different instances regarding the size of them is changed and this issue will be explained in Chapter 7.

The **constraints** are presented and explained below:

1. Task i of model j of box k should be assigned to the required operator o:

$$OT_{oi} - X_{ijko} \geq 0, \quad i \in [I], \quad j \in [J], \quad k \in [K], \quad o \in [O]$$

2. Task i of model j of box k should be assigned to exactly one operator:

$$\sum_{o=1}^{O} X_{ijko} = 1, \quad i \in [I], \quad j \in [J], \quad k \in [K]$$

3. Task i of model j of box k should be assigned to the required type t of machine m:

$$TRM_{it}MTY_{mt} - Y_{ijkmt} \geq 0, \quad i \in [I], \quad j \in [J], \quad k \in [K], \quad m \in [M], \quad t \in [T]$$

4. Task i of model j of box k should be assigned to exactly one machine m of type t:

$$\sum_{m=1}^{M} \sum_{t=1}^{T} Y_{ijkmt} = 1, \quad i \in [I], \quad j \in [J], \quad k \in [K]$$

5. Available time:

$$\sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} X_{ijko} Q_{jk} P_{ij} \leq Av, \quad o \in [O]$$

6. Operators' workload:

$$W_o = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} X_{ijko} Q_{jk} P_{ij}, \quad o \in [O]$$

7. If task i of model j of any box k is assigned to operator o and to machine m of type t, then operator o is allocated to machine m:

$$X_{ijko} + Y_{ijkmt} - 1 \leq Z_{om}, \quad i \in [I], \quad j \in [J], \quad k \in [K], \quad o \in [O], \quad m \in [M], \quad t \in [T]$$

8. Each machine m can only be allocated to one operator o:

$$\sum_{o=1}^{O} Z_{om} \leq 1, \quad m \in [M]$$

9. Average operator workload:

$$\bar{W} = \tfrac{1}{MW} \sum_{o=1}^{O} W_o$$

The 'minimum' number of workstations (MW) required is calculated using the following expression, where D_j represents the demand of model j:

$$MW = \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} D_j P_{ij}}{Av}$$

The optimisation model was programmed into the CPLEX optimisation Studio, which applies an exact optimisation method. As expected, and due to the large dimension of the real industrial instances, it was not possible to attain optimal solutions. However, the development of the model was quite valuable in order to provide a better understanding of the situation, to solve small size instances and to evaluate the approximate methods, ASBsm and RPW-VNDbal, which will be explained next.

4.2 First Solution Method: ASBsm

The Assembly System Balancing Solution Method (ASBsm) proposed to solve the balancing problems integrates two heuristics: a constructive heuristic to generate a feasible initial solution and an improvement heuristic, which is a neighbourhood search procedure inspired by TS. The venture into this approximate approach (ASBsm) to deal with the real problem is an obvious consequence of various types of considerations: a new assembly system composed of various stitching lines, not discussed in the literature; reaching a good equilibrium between practical solutions and real modelling limitations; the intrinsic combinatorial complexity of the MALBP; the large dimension of the real instances and their unclear data.

By taking into account the reality of the factory, ASBsm considers boxes of shoe components as unity, and assigns tasks of different models, in various boxes, to suitable operators and machines, in order to balance the lines. The details are presented in the following sections.

4.2.1 ASBsm: Constructive Heuristic

Production orders of various footwear models are known in advance. The first phase is creating boxes with the shoe parts, usually containing a maximum of 10 sets. The following example with three models to be produced, including models A, B and model C, with order quantities of 10, 20 and 14 pairs as shown in Table 4.1, respectively, will be used to illustrate the heuristic. Let us name this example as $BalInstance$. It will be used a few times in this thesis.

Table 4.1: Production Order.

Model	Order Q.ty
A	10
B	20
C	14

A possible set of corresponding boxes is represented in Table 4.2. Inside each box there are items of the same model, the same size (size of a shoe) and equal colour. Furthermore, each box has two identifications: one of them is the box counter (k) and the other is the model type inside (j); for instance, box 2 of model C (Box_{32}) has 4 pairs inside.

Table 4.2: Creating Boxes and Quantities.

	Box_{jk}	
Model(j)	Box (k=1)	Box (k=2)
A (j=1)	10	0
B (j=2)	10	10
C (j=3)	10	4

Each model has a routing of different tasks, depicted in Table 4.3 with the processing time of the tasks and the next phase assigns the different tasks of each box to the suitable operators and machines, which will be explained below.

Table 4.3: Model Routings.

Model	Task	Sequence	Processing Time
A	10	2	7
A	20	1	8
A	30	3	4
A	40	0	0
B	10	1	3
B	20	2	5
B	30	4	5
B	40	3	3
C	10	3	4
C	20	2	7
C	30	0	0
C	40	1	3

The heuristic will sort all the tasks of all the boxes. For that, at the beginning,

boxes of all models are ordered. Afterwards, the prior model of the first box is collected (in the example of model A of box 1, with the quantity of 10) and then the second and so on until all the models/boxes are concluded. Then the boxes undergo different tasks according to their sequence and model type as mentioned before. After that, the next model of the first box is selected (box 1 of model B with the quantity of 10, in the example) and, when all the tasks of all the first boxes are listed, then the procedure is repeated for the next boxes until all the tasks are sorted in a matrix. Table 4.4 shows the matrix, which refers to $BalInstance$ (see beginning of this section), with all the tasks of each model in each box which named it as $TasModBox$. Processing Time is shown by (PT). The entrance order of the tasks in this Table is also based on the sequence of the tasks in each routing. Moreover, the assignment of the tasks to the operators and machines will be done by considering the given order in the mentioned Table (4.4).

Table 4.4: TasModBox Matrix.

Box	Model	Task	Sequence	Box Q.ty	Box Processing Time
1	A	20	1	10	80
1	A	10	2	10	70
1	A	30	3	10	40
1	B	10	1	10	30
1	B	20	2	10	50
1	B	40	3	10	30
1	B	30	4	10	50
1	C	40	1	10	30
1	C	20	2	10	70
1	C	10	3	10	40
2	B	10	1	10	30
2	B	20	2	10	50
2	B	40	3	10	30
2	B	30	4	10	50
2	C	40	1	4	12
2	C	20	2	4	28
2	C	10	3	4	16

For some tasks, there is only one resource available, which leads to a pre-assignment (before designating other tasks). There are two types of pre-assignment, the first taking into account the operators and the second the machines, as explained below:

1. If a task requires a specific type of machine and only one of those machines exists, it must be selected for that task, even if other types characterise that machine. Afterwards, if that machine still has enough free time, it may be assigned to other tasks requiring the types such machine represents. As an example, if task J requires a type B machine and machine 4 is the only one of type B, although it may also be characterised by other types (e.g. A and C), machine 4 should be selected and assigned to an operator, firstly as a type B machine.

2. If only one operator has the ability to perform a specific task, such task must be assigned to them, the workload being updated taking into consideration the processing time of the picked task multiplied by the amount of boxes. Other tasks can be assigned to that operator if there is enough free time, regarding the available time or a used percentage of the available time;

We continue with the example $BalInstance$ in Table 4.5, which presents machine type requirements. Table 4.6 describes machine types and Table 4.7 shows the tasks that the operators can execute, according to their skills.

Table 4.5: Machine Type Requirements for each Task.

Task	Type of Machine
10	Type1
20	Type2
30	Type3
40	Type4

Table 4.5 shows that task 40 needs machine $Type4$ for processing and, according to Table 4.6, there is only one $Type4$ machine available, which is machine $Ma3$. This machine has one other type; therefore, specificity occurs and a workstation should be created in advance by assigning task 40 to machine $Ma3$. Table 4.8 provides an example of this specificity and includes minimum occupied times of $Ma3$.

As there is only one operator to execute task 30 (operator $Op6$), then a second kind of specificity occurs. This means, that task should already be assigned to the selected operator and the operator's workload should be updated taking into

Table 4.6: Machine Types.

Machine	Type
Ma1	Type1
Ma1	Type2
Ma2	Type1
Ma3	Type2
Ma3	Type4
Ma4	Type3
Ma5	Type2
Ma6	Type3
Ma7	Type2
Ma8	Type2
Ma8	Type3
Ma9	Type1
Ma9	Type2
Ma10	Type2

Table 4.7: Tasks According to Operators' Skills

Operator	Task
Op1	20
Op2	10
Op2	20
Op3	20
Op4	10
Op4	40
Op5	10
Op6	20
Op6	30
Op7	20
Op7	40
Op8	10
Op8	20
Op8	40

Table 4.8: Special Tasks and Processing Times for Ma3.

Special Task	Model	Box N.er	Box Q.ty	Box Processing Time
40	B	1	10	30
40	C	1	10	30
40	B	2	10	30
40	C	2	4	12
		Sum		102

account the processing time of the selected task, multiplied by the box quantity. Other tasks may be assigned to this operator if she or he has enough free time. It is necessary to verify the available time restriction (constraint number 5). Table 4.9 provides an example of this specificity and includes minimum occupied times of $Op6$.

Table 4.9: Special Tasks and Processing Times for Op6.

Special Task	Model	Box N.er	Box Q.ty	Box Processing Time
30	A	1	10	40
30	B	1	10	50
30	B	2	10	50
		Sum		140

After allocating special tasks to the machines and/or operators capable of executing them, the heuristic allocates the remaining tasks. It starts by assigning tasks of the $TasModBox$ matrix according to their order and respecting the constraints below which were mentioned in the previous part (optimisation model).

1. Each task of a model of a box should be assigned to a required operator;
2. Each task of a model of a box should be assigned to exactly one operator;
3. Each task of a model of a box should be assigned to a required machine type;
4. Each task of a model of a box should be assigned to exactly one machine of the required type;
5. Each operator should not work more than an available period of time or defined percentage of the available time;
6. Each machine can only be allocated to one operator;

After picking a task, the heuristic finds the first operator with the required skill and available time to work on the components in the box; otherwise, it searches for another operator. In the next step, the heuristic searches for a required type

machine, not yet allocated, repeating the procedure until all the tasks are routed. When a task is assigned to an operator and a machine, then a corresponding workstation is created.

Quite often, WIP has to be taken into consideration, as there are boxes in the lines, with some tasks already executed. Consequent adaptations of the input data are performed, such as updating routings. In order to achieve a better initial solution, two different initial sequences of operators are contemplated. Moreover, to deal with the eventual infeasibility, the sequence of operators is changed and, if after a limited number of iterations, no feasible initial solution is obtained, then the production plan should be changed. The results obtained (initial solution) are provided in Table 4.10.

Table 4.10: Initial Solution of the ASBsm.

Task	Model	Box N.er	Operator	Machine	Time
20	A	1	Op1	Ma1	80
10	A	1	Op2	Ma2	70
30	A	1	Op6	Ma4	40
10	B	1	Op2	Ma2	30
20	B	1	Op1	Ma1	50
40	B	1	Op4	Ma3	30
30	B	1	Op6	Ma4	50
40	C	1	Op4	Ma3	30
20	C	1	Op1	Ma1	70
10	C	1	Op2	Ma2	40
10	B	2	Op2	Ma2	30
20	B	2	Op3	Ma5	50
40	B	2	Op4	Ma3	30
30	B	2	Op6	Ma4	50
40	C	2	Op4	Ma3	12
20	C	2	Op1	Ma1	28
10	C	2	Op2	Ma2	16

4.2.2 ASBsm: Improvement Heuristic

ASBsm integrates an Improvement Heuristic, which is evolved by using different problem adequate neighbourhoods and local search. The heuristic takes some inspiration from TS, one of the most well-known metaheuristics, which has been successfully used in many combinatorial optimisation problems. It has also been

applied to solve ALBP (Özcan et al. [2009]), but not specifically in the footwear industry. The heuristic improves the solutions, not only by local search, but also by promoting diversification by varying neighbourhood structures. The concept of tabu list shows up when these structures are introduced along the iterative procedure.

In order to create appropriate solutions with the Improvement Heuristic, a careful attempt is made to devise effective neighbourhood structures (Nbi), close to the industrial reality, by transferring and swapping a task, or by performing some tasks at once between workstations. All neighbours of a current solution are evaluated and the best one, regarding the objective function, is chosen. The movement among various neighbourhood structures are based on the Improvement Heuristic, which will be explained. The dimension of each neighbourhood structure is related to the instances used, in the different types of problems. The essentials are explained below.

Nb1: Transfer all tasks of machine m of operator o to machine m' and operator o';

In this neighbourhood, a machine that is already in use is selected and checked for types. If the machine is of one type only, another used machine of the same type is locked; after this, only if the operator of the second machine is able to perform all the tasks of the first machine, and if there is enough free time, considering the available time, the task can be transferred.

If the first machine contemplates different types, then another operational machine is chosen, regardless of the types. Then, the operators assigned to each of the machines are identified and all assigned tasks of the first machine and operator, one by one, are considered. If the second machine and operator are able to carry out all of the tasks and have enough free time, tasks can be transferred as well.

Nb2: Transfer one task of machine m of operator o to machine m' and operator o';

The process is the same as in $Nb1$ except that in $Nb2$ only one task will be transferred in each step. Using $Nb2$ may reduce the number of workstations, but it especially favours the smoothing part of the objective function.

Nb3: Transfer tasks of machine m with operator o to other machines and operators (could be more than one machine and operator);

The process of finding machines and operators is the same as in $Nb1$. Tasks are transferred one by one, and, if the selected operator and machine do not have enough time, another machine and operator will be chosen to perform the remaining tasks. Therefore, tasks assigned to a machine and operator may be transferred to different machines and operators. This transfer could be for one task or all tasks, depending on the problem and the possibility of transferring tasks.

Nb4: Swap all tasks of machine m of operator o with machine m' and operator o' if operator o is assigned to more than one machine (e.g: machines m and m''). The working time of an operator who works on more than one machine is calculated by adding the working times of the different machines. Changes in an operator's working time may contribute to changes in the smoothing part of the objective function. The process of finding machines and operators is the same as in the previous neighbourhoods but, in this case, all tasks of both machines will be tested to verify if each machine and operator have the ability and enough time for the new tasks. At the beginning, operators who work on more than one machine are searched.

Other structures were tested, in particular swapping a task among workstations. However, they were discarded because they increased running times and did not significantly improve the quality of the solution.

The computational experiments showed that the neighbourhood structures might have different impacts on both objectives: reducing the number of workstations and smoothing workload. Although the structures can be used for both, Table 4.11 illustrates the best impact for each objective.

Table 4.11: Neighbourhoods and Objectives.

Reducing N.er Worksations	Smoothing Workload
Nb1	–
Nb2	Nb2
Nb3	–
–	Nb4

Tabu list and Diversification: ASBsm includes a diversification phase to drive the search into new regions by changing the neighbourhood types. After a particular neighbourhood is explored and the best solution is found, such neighbourhood structure enters a tabu list, being forbidden for a number of neighbourhood iterations. In the next step, a new neighbourhood is searched. It is expected that other neighbourhood structures, with diverse effects, may have the chance to improve the current solution. Given the number of neighbourhood structures, which is 4, the length of the tabu list includes 2 iterations.

Aspiration criterion: If a neighbourhood structure (non-tabu) does not improve the current solution, then an aspiration criterion is employed, overriding a tabu state by resorting to the older element of the list. If again an improvement is not met, then another tabu neighbourhood structure (recently becoming tabu) is applied and examined.

Termination condition: A termination condition is 30 iterations or no improvement occurs during these iterations; then the algorithm ends sooner.

4.3 Second Solution Method: RPW-VNDbal

Another proposed solution method (RPW-VNDbal) is based on VND and resorts to a constructive heuristic to generate initial solutions. The constructive heuristic is, itself, an adaptation of the RPW method to the more general and complex problem. VND improves the initial solutions by considering different neighbourhood spaces. The details are presented ahead.

The section 4.1 refers to an optimisation model but, as stated, the large dimension of the industrial balancing problems made its practical resolution impossible, due to the number of workstations, tasks and operators with diverse skills. This is the reason for the development and application of such an approximate but applicable method. Also, in the previous section, the used method was based on an approximate procedure and heuristic, which in this section we use a metaheuristic approach. Therefore, in view of the literature review, and given the new type of problem, we decided to use VND, expecting that it may promote a good solution

diversification via various neighbourhood structures. More details about VND are presented in this section.

4.3.1 RPW-VNDbal: Constructive Heuristic

RPW-VNDbal considers the boxes of components moving in the systems as unities and assigns the tasks of the different models in the boxes to the proper operators and machines. The goal of the constructive heuristic is to find initial solutions. It is the result of a convenient adaptation of the RPW method.

RPW was introduced by Helgeson and Birnie [1961] for a single model line. RPW contemplates two elements: firstly, the addition of task times that follow a selected task in the model routing, called RPW values; secondly, the position of the selected task in the routing diagram. The tasks are assigned to the workstations according to their RPW values. Consequently, and regarding constraints 2 and 4 discussed in the section 4.1, all tasks should be allocated to an operator and a machine; otherwise, it will not be feasible regardless of whether the sequencing constraint is met or not. Therefore, the boxes are sent to workstations taking into account task sequencing constraints, but not considering any sequence in the lines, because they are multi-directional.

As stated, this study covers MALBP in which operators have different skills; machines are of various types; boxes with components move in the lines; and pre-assignment is accomplished. Consequently, the procedure verifies all models and tasks in the production plan. Then, it considers the types of machines that are required to perform these tasks. If only one machine is available for a type and that machine is characterised by different types, then, before any other action, it should be assigned to the most adequate operator. Otherwise, the solution may not be feasible. A simple example will help clarify the procedure.

In the given *BalInstance* in the previous section, Table 4.1 referred to 3 models to be produced and Table 4.2 showed that the boxes (maximum number of 10) that move along a line have been created.

The Constructive heuristic is as follows:

1. The RPW value for each task of each model is calculated by adding the processing time of the selected task and the times of all tasks that follow the selected task in the sequencing diagram;

2. After that, RPW values are multiplied by the box quantities, obtaining the values called RPWBQ;

3. Then, the RPWBQs are sorted from the largest to the smallest value;

4. Next step takes into account task specificities. If there is only one operator to perform a task, the selected task is assigned to that operator and his/her workload is updated, considering the processing time of the assigned task, multiplied by the box quantity;

5. After this, a machine is allocated to the selected operator and selected task, according to machine requirements.

Frequently, as mentioned in the previous section, WIP in the systems must be considered. One way to address that is to allocate the remaining tasks (of the WIP) to proper workstations before considering new orders. Therefore, the first boxes are WIP, and the boxes that follow are created to complete the new orders; another way of handling WIP is turning it into new models.

Let us consider again $BalInstance$ and get back to Table 4.3, including model routings. Table 4.5 presented machine type requirements. Table 4.6 described machine types and Table 4.7 showed the tasks that the operators can execute, according to their skills (as mentioned in the previous section). The two mentioned specialities in the the past section also occurred here.

However, the steps that follow are adapted from the RPW method (Tables 4.12, 4.13 and 4.14).

Finally, and as mentioned about the previous method (ASBsm), two different sequences of operators are considered in the input data. The one with the best objective will be chosen as the initial solution. However, in real world instances, it is not always easy to generate a feasible solution only by applying the rules

Table 4.12: RPW Values.

Model	Task	RPW
A	10	11
A	20	19
A	30	4
A	40	0
B	10	16
B	20	13
B	30	5
B	40	8
C	10	4
C	20	11
C	30	0
C	40	14

Table 4.13: RPW Values for Boxes considering Quantities (RPWBQ).

Model	Task	Box N.er-Box Q.ty	RPW	RPWSBQ
A	10	1-10	11	110
A	20		19	190
A	30		4	40
A	40		0	0
B	10	1-10	16	160
B	20		13	130
B	30		5	50
B	40		8	80
C	10	1-10	4	40
C	20		11	110
C	30		0	0
C	40		14	140
B	10	2-10	16	160
B	20		13	130
B	30		5	50
B	40		8	80
C	10	2-4	4	16
C	20		11	44
C	30		0	0
C	40		14	56

Table 4.14: Sorted RPWBQs.

Model	Task	Box N.er	Sorted RPWSBQ
A	20	1	190
B	10	1	160
B	10	2	160
C	40	1	140
B	20	1	130
B	20	2	130
A	10	1	110
C	20	1	110
B	40	1	80
B	40	2	80
C	40	2	56
B	30	1	50
B	30	2	50
C	20	2	44
A	30	1	40
C	10	1	40
C	10	2	16
A	40	1	0
C	30	1	0
C	30	2	0

mentioned above. In fact, the sequential task assignment to an ordered list of operators and machines may not be feasible. To overcome this problem, different orders of operators are considered, as new input data. If after a limited number of changes, no feasible solution is obtained, then the production orders must be changed. Considering the mentioned rules, the initial solution of the $BalInstance$ in this section is shown in Table 4.15. The next subsection describes the VND, the substantial part of RPW-VNDbal.

4.3.2 RPW-VNDbal: Variable Neighbourhood Descent

The metaheuristic VND has been chosen for the improvement part of the solution method. Chapter 3 includes a brief description of VND, which has been applied to solve various combinatorial optimisation problems and, in particular, to some ALBP (Polat et al. [2016] and Kellegöz [2016]). In order to obtain good solutions, suitable neighbourhood spaces are defined. Then, all neighbours of a current solution are evaluated by transferring and swapping tasks between workstations. The best one in

Table 4.15: Initial Solution of the RPW-VNDbal.

Task	Model	Box N.er	Operator	Machine	Time
20	A	1	Op1	Ma1	80
10	B	1	Op2	Ma2	30
10	B	2	Op2	Ma2	30
40	C	1	Op4	Ma3	30
20	B	1	Op1	Ma1	50
20	B	2	Op1	Ma1	50
10	A	1	Op2	Ma2	70
20	C	1	Op3	Ma5	70
40	B	1	Op4	Ma3	30
40	B	2	Op4	Ma3	30
40	C	2	Op4	Ma3	12
30	B	1	Op6	Ma4	50
30	B	2	Op6	Ma4	50
20	C	2	Op1	Ma1	28
30	A	1	Op6	Ma4	40
10	C	1	Op2	Ma2	40
10	C	2	Op2	Ma2	16

terms of objective function is chosen and moving between different neighbourhood structures is based on the VND. The dimension of each neighbourhood structure is related to the instances used.

The four neighbourhood structures are used in this section and they are the same as the four previous neighbourhood structures ($Nb1 - Nb4$). They have, purposely, diverse impacts on the objective function. Figure 4.1 illustrated neighbourhood applications with more visual insight, through small examples of allocation of tasks and machines to operators. In the case of $Nb3$, two possibilities are described, which may depend on the constraints, for example, the impossibility of transferring all tasks. In each neighbourhood structure, all possible incumbent neighbour solutions are searched.

$Nb1$ and $Nb3$ have the capacity to reduce the number of workstations at each step. When a workstation works only on one task, $Nb2$ also reduces the number of workstations; otherwise, as mentioned earlier, it is more effective in the smoothing part of the objective function. $Nb4$ only improves the solution relative to the smoothing part. When transferring or switching from one workstation to another (or others) is no longer possible, the objective value is calculated. If the new solution is better, it is saved to compare with other neighbour objectives. Diversification visibly

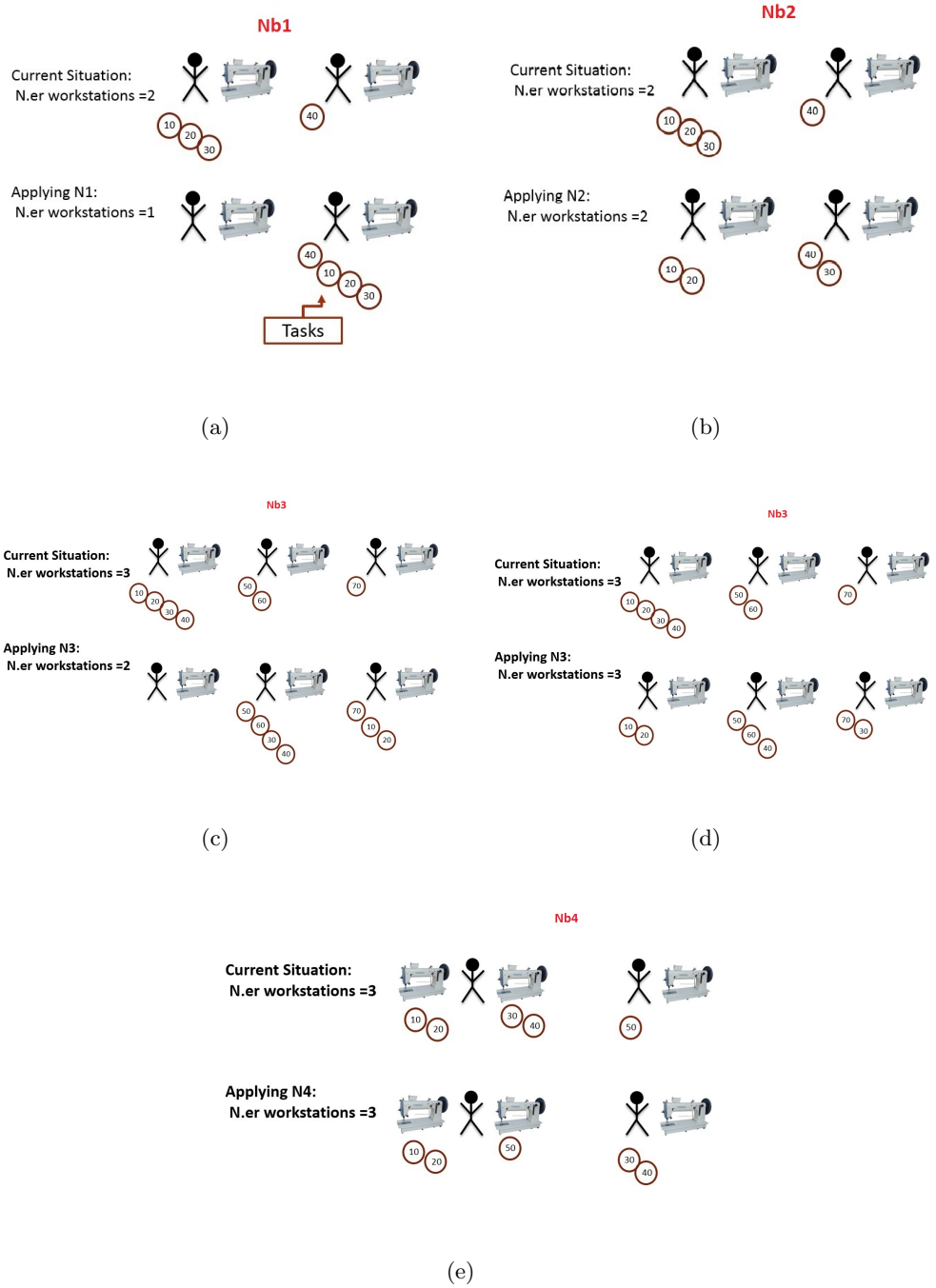

Figure 4.1: Different Neighbourhood Structures.

takes place when all the neighbourhoods are applied; the way of the movement from each neighbourhood structure to another, which is a VND attribute, helps the

intensification.

The VND part of RPW-VNDbal improves the initial solutions by searching several areas of the solution space, as it is its purpose. According to Polat et al. [2016], it is possible to implement a sequential and a nested analysis. The latter uses more running time. Here, the neighbourhoods are sequenced in a way that swaps are completed after transfers. These end after 25 iterations, for the transfers, and 10 runs, for the swap structure. Please see Figure 4.2, for a detailed explanation of the VND algorithm.

During the start of the procedure, the incumbent solution is equal to the initial solution. The defined neighbourhood structures then search all neighbours of the incumbent solution and the best one in each space is chosen. These are saved in a matrix row called $ObjectiveValueforNeighbours$ (line 3). Each element of this matrix will be compared in line 9, and, if an improvement occurs by a special neighbour, then the search will choose the superior neighbour and the incumbent solution will be updated to the best one (lines 8 to 14). Then, the VND algorithm will update the $ObjectiveValueforNeighbours$ matrix and all its elements will be equal to the objective of the incumbent solution. Other information related to the incumbent solution will be updated and finally, as stated, it will remain in the chosen neighbour by making $CheckForImprovement$ equal to false (lines 26 to 31).

In another phase, between lines 18 and 24, if there is no improvement in the current neighbourhood, then the procedure will control all other neighbourhood spaces, and the search will shift to the best one. The new results of each neighbourhood search will be saved in the $ObjectiveValueforNeighbours$ matrix, which will be updated.

VND ends after a limited number of running times, which is defined by a loop in line 7. This number was initiated in line 2. Lines 15 to 17 are defined to finalise the search process, if there is no improvement in the objective function.

$ChosenNeighbour$ shows the number of the neighbourhood that is chosen at each iteration and $CheckForImprovement$ changes to true, if after comparison, at each iteration, there is an improvement in the elements of the $ObjectiveValuefor$

Neighbours matrix.

When the condition for process termination is attained, the swap neighbourhood structure is applied to the new founded incumbent solution for a maximum of 10 iterations. If no other improvements occur, the procedure ends earlier.

4.4 Simulation and Conclusions

This section explains the use of *Simulation* and draws some conclusions. The two previous sections comprised two new solution methods, ASBsm and RPW-VNDbal, for the balancing problems, modelled in section 4.1. The similarities and differences between both methods are portrayed in Figures 4.3 and 4.4.

It was decided to complement the work done relative to the solution methods with simulation of the results obtained, in order to gain more confidence about the completion of the programmed plans, to illustrate them and to help company managers in the decision processes. Therefore, a simulation model is developed for this reason.

In fact, Chapter 4 focuses on Balancing, not specially considering task sequencing constraints for the models of shoes in the lines. Simulation contemplates sequencing and evaluates diverse scenarios according to the available line production times. The available time can decrease up to 40% and in intervals of 5%. Then, ASBsm and RPW-NVDbal are run for each scenario and the results found are tested by simulation, to verify when all boxes created can be produced. For instance, in the first step, the initial available time is considered and if the production plan cannot be met, in this time period, then it will be reduced by 5% and, therefore, 95% of the available time will be taken up in the next step. In case the production plan is executed, no more reductions are required. Otherwise, a further reduction up to 90% will take place and the entire process is repeated.

Another rule followed in the simulation is related to the entrance order of the boxes in the lines. The number of the tasks in the routings of the boxes are calculated. Then, they are sorted according to these values, from the largest to the smallest. Afterwards, based on that ordering, the tasks of each box are picked for

Figure 4.2: VND Algorithm.

```
initialization;
MaxRunningforVND = DefinedNumber;
Save the calculated objective value of each neighbour in
  ObjectiveValueforNeighbours matrix;
ChosenNeighbour = −1;
CheckForImprovement = false;
IncumbentSolution = InitialObjectiveValue;
for Counter1 = 0 to Counter1 ≤ MaxRunningforVND do
    for Counter2 = 0 to Counter2 ≤ NumberofNeighbours do
        if ObjectiveValueforNeighbours[Counter2][0] ≤ IncumbentSolution then
            IncumbentSolution = ObjectiveValueforNeighbours[Counter2][0];
            ObjectiveValue = ObjectiveValueforNeighboors[Counter2][0];
            ChosenNeighbour = Counter2;
            CheckForImprovement = true;
        end
        if ChosenNeighbour = −1 then
            Counter1 = MaxRunningforVND;
        end
        if ChosenNeighbour = NumberofNeighbours − 1 and
          CheckForImprovement = false then
            if ChosenNeighbour = A Special Neighbour then
                Other neighbours will be considered (not A Special Neighbour
                  which is the current one);
                Update bjectiveValueforNeighbours matrix;
                ChosenNeighbour = −1;
            end
        end
    end
    if ChosenNeighbour = A Special Neighbour then
        Make ObjectiveValueforNeighboors matrix equal to the
          IncumbentSolution;
        Update all information related to the IncumbentSolution;
        Stay in the current neighbourhood region;
        CheckForImprovement = false;
    end
end
```

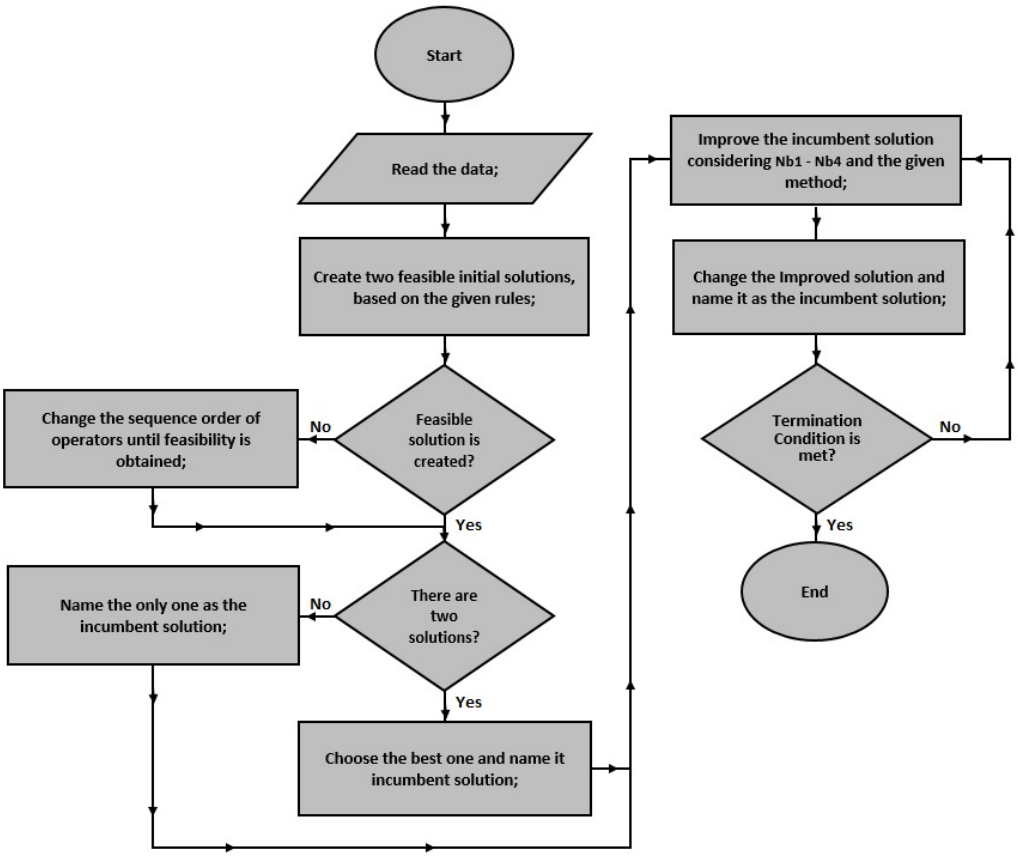

Figure 4.3: General Rules of Both Solution Methods.

workstation allocation and, if there are more than one task in queue for a workstation, they will be assigned according to their arrival times. For example, if there are 3 boxes to enter one line and the number of tasks in their routings includes: box-1, 10 tasks; box-2, 5 tasks and box-3, 6 tasks, then their entrance order is: box-1, box-3 and box-2 and their tasks will be assigned according to this order while also considering arrival times, as mentioned.

The methods were compared in various ways and on different types of instances, mainly based on real data. Moreover, the optimisation model was also tested on small instances. Such results may be found in Chapter 7. Nevertheless, we can already say that, in general, the second solution method (RPW-NVDbal) works better than the first method (ASBsm).

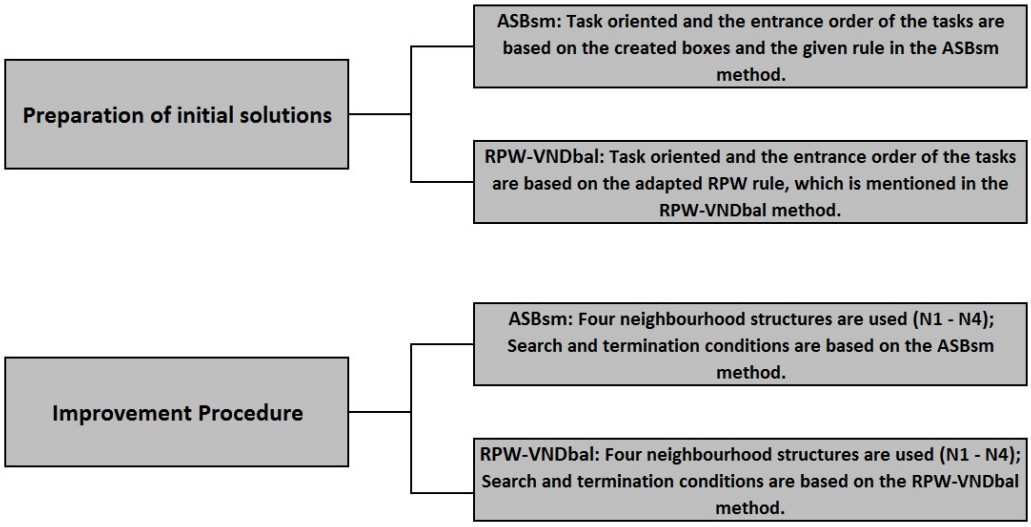

Figure 4.4: Differences of Both Solution Methods.

Chapter 5

Sequencing Mixed-model Assembly Systems

This Chapter contemplates Mixed-model Assembly Line Sequencing Problems (MA -LSP). It is presumed that the balancing problems are solved in advance. Therefore, the sequencing of the mixed-model systems will take into account that the assignments of tasks are already established. An optimisation model for the problems under consideration is developed. However, due to its type and dimension of the problems, two approximate solution methods are developed.

One of these methods is based on the VND metaheuristic, called Variable Neighbourhood Descent Mixed-model Sequencing ($VND-MSeq$). Different dispatching rules are used to construct the initial solutions and the best one is chosen as the initial answer. Varied neighbourhood structures are used in the VND-MSeq method which will be explained below. The other approach is based on GA which is called Genetic Algorithm Mixed-model Sequencing ($GA-MSeq$). For generating the initial populations, diverse dispatching rules are devised and then, GA is used as the improvement procedure. The main aspects of both methods are mentioned at the end of this chapter in Section 5.4. Moreover, their detailed evaluations and comparisons are addressed in Chapter 7.

5.1 Optimisation Model

In this section, an optimisation model for real world mixed-model assembly sequencing problem is proposed. As already mentioned, it is assumed that the results of the balancing problems are available; therefore, being known the values of X_{ijko} and Y_{ijkm} these variables are no longer considered. Operators and machines are considered simultaneously. As in the previous chapter, WIP is also considered. For example, in large instances the number of the available boxes on the line is around 3400 pairs of shoes. Once again, the unity is the box and not a pair of shoes.

Objective: The objective is the minimisation of makespan, i.e. the minimisation of the last completion times of all the tasks (included in all boxes) of all operators and machines. The following indices, parameters and decision variables are considered in the mixed integer optimisation model.

Indices:

j: index for model, $j \in [J]$, $[J] = \{1, 2, \ldots, J\}$

i: index for task, $i \in [I]$, $[I] = \{1, 2, \ldots, I\}$

k: index for number of boxes, $k \in [K]$, $[K] = \{1, 2, \ldots, K\}$

o: index for operator, $o \in [O]$, $[O] = \{1, 2, \ldots, O\}$

m: index for machine, $m \in [M]$, $[M] = \{1, 2, \ldots, M\}$

Input parameters:

P_{ij} = processing time of task i for model j

S_{ij} = sequence of task i for model j

Q_{jk} = quantity of model j in box k

EC_{ij} = earliest completion time of task i for model j

$X_{ijko} = 1$: if task i of model j of box k is assigned to operator o; 0: otherwise

$Y_{ijkm} = 1$: if task i of model j of box k is assigned to machine m; 0: otherwise

Av = available time

GM = great number

Decision variables:

C_{ijk} = completion time of task i of model j of box k

$U_{ijki'j'k'} = 1$: if completion time of task i' of model j' of box k' is greater than completion time of task i of model j of box k; 0: otherwise

$Cmax$: the largest completion time

Optimisation model:

The objective function is the minimisation of the makespan.

$$Minimise \quad (Cmax)$$

The **constraints** are presented and explained below:

1. The sequence of any task i for each model j in box k should be respected:

$$C_{i'jk} \geq C_{ijk} + P_{i'j}Q_{jk}, \quad i, i' \in [I], \quad j \in [J], \quad k \in [K], \quad S_{i'j} > S_{ij}$$

2. Each operator o must work on only one task i of model j of box k, at any time (tasks of the same operator do not overlap in time):

$$C_{ijk} \geq C_{i'j'k'} + P_{ij}Q_{jk} - GM(U_{ijki'j'k'}),$$

$$i, i' \in [I], \quad j, j' \in [J], \quad k, k' \in [K], \quad o \in [O] \quad X_{ijko}X_{i'j'k'o} = 1 :$$

$$(j \neq j') \quad \vee \quad (j = j' \quad \wedge \quad k \neq k')$$

$$C_{i'j'k'} \geq C_{ijk} + P_{i'j'}Q_{j'k'} - GM(1 - U_{ijki'j'k'}),$$

$$i, i' \in [I], \quad j, j' \in [J], \quad k, k' \in [K], \quad o \in [O] \quad X_{ijko}X_{i'j'k'o} = 1 :$$

$$(j \neq j') \quad \vee \quad (j = j' \quad \wedge \quad k \neq k')$$

3. Each machine m must work on only one task i of model j of box k, at any time (tasks of the same machine do not overlap in time):

$$C_{ijk} \geq C_{i'j'k'} + P_{ijk} - GM(U_{ijki'j'k'}),$$

$$i, i' \in [I], \quad j, j' \in [J], \quad k, k' \in [K], \quad m \in [M] \quad Y_{ijkm} Y_{i'j'k'm} = 1 :$$

$$(j \neq j') \quad \vee \quad (j = j' \quad \wedge \quad k \neq k')$$

$$C_{i'j'k'} \geq C_{ijk} + P_{i'j'k'} - GM(1 - U_{ijki'j'k'}),$$

$$i, i' \in [I], \quad j, j' \in [J], \quad k, k' \in [K], \quad m \in [M] \quad Y_{ijkm} Y_{i'j'k'm} = 1 :$$

$$(j \neq j') \quad \vee \quad (j = j' \quad \wedge \quad k \neq k')$$

4. The completion time of each task i of model j of box k should be less than the available time:

$$C_{ijk} \leq Av, \quad i \in [I], \quad j \in [J], \quad k \in [K]$$

5. The completion time of each task i of model j of box k cannot be less than the earliest completion time:

$$C_{ijk} \geq EC_{ij} Q_{jk}, \quad i \in [I], \quad j \in [J], \quad k \in [K]$$

6. Cmax (MakeSpan) is the longest completion time:

$$C_{ijk} \leq Cmax, \quad i \in [I], \quad j \in [J], \quad k \in [K]$$

As in Chapter 4, the optimisation model was developed with CPLEX optimisation Studio. The model provides a better understanding of the situation. Moreover, due to the large dimension of the real industrial instances, it was not possible to attain optimal solutions. But, they were used to solve small-size instances and to evaluate the approximate methods i.e. VND-MSeq and GA-MSeq, which will be explained in the following.

5.2 Proposed Solution Method: VND-MSeq

In this section, the proposed solution method ($VND - MSeq$) to solve the MALSP is based on the VND metaheuristic. Via this method, we get the advantage of having diversification by testing different neighbourhood structures. This method is

composed of two parts. The first part is a constructive heuristic, which is developed to generate the initial solution; in the second part, VND is used to improve the solution.

The input data for the given problem, as mentioned earlier, are from the previous chapter, which are the results of the balancing problems. The assignment of the tasks from different boxes, and related to diverse models, to the operators and machines is determined by following the approach provided in Chapter 4. Now, the issue is about sequencing the tasks/boxes in the lines. VND-MSeq will be applied to various stitching lines of both assembly systems.

5.2.1 VND-MSeq: Constructive Heuristic

The production orders of different models are previously known. Further, according to the results of the balancing problems, the assignment of different tasks of varied boxes of different models to the operator and machines is determined and the proposed question, as mentioned, is to find the sequence of all the available tasks of operators and machines while minimising the makespan by satisfying the given constraints of the optimisation model. In general, the shape of the input data and the decision that should be made are presented in Table 5.1. In this Table, the $S : Time$ and $E : Time$ are Starting time and Ending time of a given task, which are the questions of this chapter.

Table 5.1: Input Data and the Decisions that should be Taken.

Task	Model	Box N. er	Operator	Machine	S: Time	E: Time
20	A	1	Op1	Ma1	?	?

To better understand the problem, an instance is provided named $SeqInstance$. In the given example ($SeqInstance$), there are 3 models available. Different tasks in each model routing are shown in Table 5.2. Table 5.3 also depicts the assignment of the tasks of varied boxes and models to the operators and machines.

To produce possible initial solutions, different dispatching rules are used so that VND-MSeq needs one initial solution. Thus, the one with the best objective

Table 5.2: Model Routings.

Model	Task	Sequence	Processing Time	Order Q.ty
A	10	2	7	20
A	20	1	8	20
A	30	3	4	20
B	10	1	3	10
B	20	2	5	10
B	30	4	5	10
B	40	3	3	10
B	50	5	6	10
C	20	2	7	29
C	40	1	3	29

Table 5.3: Input Data.

Task	Model	Box N.er	Box Q.ty	Operator	Machine
10	A	1	10	Op2	Ma2
20	A	1	10	Op1	Ma1
30	A	1	10	Op6	Ma4
10	A	2	10	Op2	Ma2
20	A	2	10	Op1	Ma1
30	A	2	10	Op6	Ma4
10	B	1	10	Op2	Ma2
20	B	1	10	Op1	Ma1
30	B	1	10	Op6	Ma4
40	B	1	10	Op4	Ma3
50	B	1	10	Op4	Ma5
20	C	1	10	Op1	Ma1
40	C	1	10	Op4	Ma3
20	C	2	10	Op6	Ma4
40	C	2	10	Op4	Ma3
20	C	3	9	Op6	Ma4
40	C	3	9	Op4	Ma3

function is chosen. According to Pinedo and Chao [1999] and as mentioned in Chapterr 3, there are different dispatching rules but the ones applied in this section are as follows:

1. Largest Number of Successors (LNS);

2. Critical Path (CP);

3. Longest Processing Time (LPT);

4. Shortest Processing Time (SPT);

The first rule, which is the *Largest Number of Successors*, is used in 3 distinctive ways and the *Critical Path* method is applied in 2 different manners. Therefore, 7 *Tables of Tasks* with different arrangements, and consequently 7 possible initial solutions, are constructed and, as mentioned before, the one with the best objective function will be chosen as the initial solution. Moreover, the *Table of Tasks* comprises all the available tasks; by changing the position of a task in that table, new solutions be achieved. Next, different applied dispatching rules are explained for the given example.

Largest Number of Successors (LNS): The LNS gets a task and calculates the number of tasks which are next to it in the model routing of that task. As an instance, the LNS value for the task 10 of model A is 1 because there is only one task after task 10 in the routing diagram of that model. This value for task 20 of the same model (A) is 2. Table 5.4 shows LNS values for all tasks of *SeqInstance*. This table is an example of a table of a task for the given example.

Table 5.4: LNS Values for Different Tasks.

Task	Model	Box N.er	LNS Value
10	A	1	1
20	A	1	2
30	A	1	0
10	A	2	1
20	A	2	2
30	A	2	0
10	B	1	4
20	B	1	3
30	B	1	1
40	B	1	2
50	B	1	0
20	C	1	0
40	C	1	1
20	C	2	0
40	C	2	1
20	C	3	0
40	C	3	1

After calculating these values, they are sorted in a descending manner, considering the LNS values from the higher to the lower ones. This sorting process is done in 3 different ways and consequently, they could create 3 different solutions. The first one is a simple sort as in Table 5.5. The second one is displayed in Table

5.6. In this way, tasks are sorted with regard to the LNS values inside each model and then, all models are sorted according to the LNS values. As an example in the *SeqInstance*, model *B* has a higher LNS value than others and therefore, it is placed at the top of the table of tasks which is shown in Table 5.6. In the third type of sorting, tasks of all boxes are sorted according to their sequence in the routing diagram, from sequence one which is the first sequence. Next, a task with the same sequence is sorted according to its LNS values in a descending way. Table 5.7 shows the case explained above.

Table 5.5: Table of Tasks - Sorted LNS Values (1st way).

Task	Model	Box N.er	LNS Value
10	B	1	4
20	B	1	3
20	A	1	2
20	A	2	2
40	B	1	2
10	A	1	1
10	A	2	1
30	B	1	1
40	C	1	1
40	C	2	1
40	C	3	1
30	A	1	0
30	A	2	0
50	B	1	0
20	C	1	0
20	C	2	0
20	C	3	0

Critical Path (CP): The CP rule for each task is calculated by summing the processing time of the selected task and the tasks which are next to it in the routing diagram of the picked task. The CP values for the *SeqInstance* are calculated in Table 5.8. Then, they are sorted in a descending mode but in 2 different ways. Therefore, they create 2 different sequences of tasks and as a result, 2 different initial solutions are generated. Generally, CP method gives priority to the tasks with higher CP values.

In the first way, tasks are sorted based on the CP values from the higher to the lower values (see Table 5.9). In the second way, they are sorted inside each model

Table 5.6: Table of Tasks - Sorted LNS Values (2nd way).

Task	Model	Box N.er	LNS Value
10	B	1	4
20	B	1	3
40	B	1	2
30	B	1	1
50	B	1	0
20	A	1	2
10	A	1	1
30	A	1	0
20	A	2	2
10	A	2	1
30	A	2	0
40	C	1	1
20	C	1	0
40	C	2	1
20	C	2	0
40	C	3	1
20	C	3	0

Table 5.7: Table of Tasks - Sorted LNS Values (3rd way).

Task	Model	Box N.er	LNS Value
10	B	1	4
20	A	1	2
20	A	2	2
40	C	1	1
40	C	2	1
40	C	3	1
20	B	1	3
10	A	1	1
10	A	2	1
20	C	1	0
20	C	2	0
20	C	3	0
40	B	1	2
30	A	1	0
30	A	2	0
30	B	1	1
50	B	1	0

(from the higher to the lower values). Then, all models are sorted according to the CP values. A model with greater CP values is placed at the top of the table of tasks (in *SeqInstance*). Table 5.10 depicts this process.

Longest Processing Time (LPT) and Shortest Processing Time (SPT): These two rules consider the processing time of each task without any attention to

Table 5.8: Table of Tasks - CP Values for Different Tasks.

Task	Model	Box N.er	Box Q.ty	Box Processing Time	CP Value
10	A	1	10	70	110
20	A	1	10	80	190
30	A	1	10	40	40
10	A	2	10	70	110
20	A	2	10	80	190
30	A	2	10	40	40
10	B	1	10	30	220
20	B	1	10	50	190
30	B	1	10	50	110
40	B	1	10	30	140
50	B	1	10	60	60
20	C	1	10	70	70
40	C	1	10	30	100
20	C	2	10	70	70
40	C	2	10	30	100
20	C	3	9	63	63
40	C	3	9	27	90

Table 5.9: Table of Tasks - Sorted CP Values (1st way).

Task	Model	Box N.er	Box Q.ty	Box Processing Time	CP Value
10	B	1	10	30	220
20	A	1	10	80	190
20	A	2	10	80	190
20	B	1	10	50	190
40	B	1	10	30	140
10	A	1	10	70	110
10	A	2	10	70	110
30	B	1	10	50	110
40	C	1	10	30	100
40	C	2	10	30	100
40	C	3	9	27	90
20	C	1	10	70	70
20	C	2	10	70	70
20	C	3	9	63	63
50	B	1	10	60	60
30	A	1	10	40	40
30	A	2	10	40	40

its sequence in the related model routing diagram. They are sorted in the LPT from the higher to the lower values and SPT method is in the opposite manner; the sorting process is from the lower to the higher values. Since the sequence of the tasks

Table 5.10: Table of Tasks - Sorted CP Values (2nd way).

Task	Model	Box N.er	Box Q.ty	Box Processing Time	CP Value
10	B	1	10	30	220
20	B	1	10	50	190
40	B	1	10	30	140
30	B	1	10	50	110
50	B	1	10	60	60
20	A	1	10	80	190
10	A	1	10	70	110
30	A	1	10	40	40
20	A	2	10	80	190
10	A	2	10	70	110
30	A	2	10	40	40
40	C	1	10	30	100
20	C	1	10	70	70
40	C	2	10	30	100
20	C	2	10	70	70
40	C	3	9	27	90
20	C	3	9	63	63

in these two approaches is not considered, then the VND-MSeq changes the position of the tasks in the table of tasks to create a feasible sequence of tasks. For example, if task 20 has 70 LPT value and this value for task 40 is 27 in a table of task, task 20 should first be allocated and then task 40. However, according to the routing diagram, task 40 is first considered and then task 20. Therefore, their position in the table of tasks with regard to LPT method should be changed according to their sequence in the routing diagram. This rule should also be observed in the SPT approach. Moreover, all these issues are shown in Tables 5.11 and 5.12 which are related to the LPT and SPT rules. In these two tables, if the position of the tasks are changed with respect to the sequence of the tasks in the routing diagram, it is mentioned Yes in the column *Position Changed*? ; otherwise, it is mentioned NO.

After creating a table of tasks by changing the position of tasks in 7 different ways, the VND-MSeq starts to create the initial solutions. By knowing the operator and machines to which each task should be allocated and considering the earliest starting time of each task, a feasible solution is created. Operators and machines should not overlap in time to perform different tasks. In the $SeqInstance$, by con-

Table 5.11: Table of Tasks - LPT Rules for the Given Tasks.

Task	Model	Box N.er	Box Q.ty	Box Processing Time	Position Changed?
20	A	1	10	80	No
20	A	2	10	80	No
10	A	1	10	70	No
10	A	2	10	70	No
40	C	1	10	30	Yes
20	C	1	10	70	No
40	C	2	10	30	Yes
20	C	2	10	70	No
40	C	3	9	27	Yes
20	C	3	9	63	No
10	B	1	10	30	Yes
20	B	1	10	50	Yes
40	B	1	10	30	Yes
30	B	1	10	50	Yes
50	B	1	10	60	No
30	A	1	10	40	No
30	A	2	10	40	No

Table 5.12: Table of Tasks - SPT Rules for the Given Tasks.

Task	Model	Box N.er	Box Q.ty	Box Processing Time	Position Changed?
40	C	3	9	27	NO
10	B	1	10	30	NO
20	B	1	10	50	Yes
40	B	1	10	30	NO
40	C	1	10	30	NO
40	C	2	10	30	NO
20	A	1	10	80	Yes
10	A	1	10	70	Yes
30	A	1	10	40	NO
20	A	2	10	80	Yes
10	A	2	10	70	Yes
30	A	2	10	40	NO
30	B	1	10	50	NO
50	B	1	10	60	NO
20	C	3	9	63	NO
20	C	1	10	70	NO
20	C	2	10	70	NO

sidering Table 5.5, the initial solution will be as in Table 5.13. Finally, after creating the initial solutions, the one with the shortest completion time or makespan will be chosen as the initial answer and improved in the next section by VND metaheuris-

tic. At this moment, it is not important if the initial answer is feasible considering the available time since in the improvement section, it becomes feasible. This happens because the input data is the output data of the balancing problems and the feasibility of the results were checked in the balancing part with the simulation.

Table 5.13: Initial Solution Considering Table 5.5. as the Table of Tasks.

Task	Model	Box N.er	S: Time	E: Time
10	B	1	0	30
20	B	1	30	80
20	A	1	80	160
20	A	2	160	240
40	B	1	80	110
10	A	1	160	230
10	A	2	240	310
30	B	1	110	160
40	C	1	0	30
40	C	2	30	60
40	C	3	110	137
30	A	1	230	270
30	A	2	310	350
50	B	1	160	220
20	C	1	240	310
20	C	2	160	230
20	C	3	350	413

In the next section the improvement process will be explained.

5.2.2 VND-MSeq: Variable Neighbourhood Descent

The metaheuristic VND is now applied as an improvement procedure. Details about VND are included in Chapter 3. According to Driessel and Mönch [2011] VNS approaches often outperform other metaheuristics for scheduling problems. Additionally, according to Roshanaei et al. [2009], VNS, a recently proposed metaheuristic technique, has quickly gained widespread success. Many papers have attempted to improve the solutions by using different local search improvement heuristics. According to Amiri et al. [2010], the VNS is one of the famous metaheuristics which has been successfully applied to solve optimisation problems. It is able to escape from the local optima through systematic changes of the neighbourhood structures during the search process. Therefore we applied VND as the improvement ap-

proach, which follows the major structures of VNS. Five neighbourhood structures ($Ns1 - Ns5$) are used for this purpose in this part.

The solution shape, as mentioned in the previous section, is a table of tasks. The position of each task in the table shows its priority to start the process on the related operator and machine. By changing the position of a task in a solution, it is possible to create a new answer. Therefore, the concept of neighbourhood structures may be utilised to change task positions, in any incumbent solution. By changing the position of tasks, it is possible to make the table of tasks infeasible. Consequently, each neighbourhood structure should be applied in a way to prevent any infeasibility. The dimension of each neighbourhood structure is firstly based on the given problem and, secondly, related to the defined values of the number of search iterations and the number of neighbour solutions, which should be searched in each iteration. These aspects will be explained in more details in the following. Additionally, in this section the problems are divided into the small and large ones. If the number of used operators is less than 11, then it is considered as small; otherwise, it is large. This deviation is only crucial in the fixed values such as the number of search iteration and number of neighbours' solutions.

The key issue considered in each neighbourhood structure is related to the feasibility which should be satisfied in all $Ns1$ to $Ns5$. By randomly (or by any other rule) changing the sequence of a task, an infeasible table of tasks may be obtained. One possibility to avoid this situation is to change the position of the tasks in the table so that, tasks which are prior to the task selected in the routing diagram are considered first. This rule is called *Precedence Preserving* and used in the whole Chapter to avoid infeasibility concerning model routings. Finally, the objective function will be calculated for the created sequence of tasks.

Ns1: This neighbourhood is based on the *Swap* structure and works as the following. Firstly, random numbers between zero and number of available tasks that show the position of each task are generated. For small instances, two random numbers are considered for one swap and for large instances, eight random values are considered for four swaps. Next, the tasks are swapped by considering the random

values two by two and as a result, a new sequence of tasks is created. Besides, the issue of feasibility should be checked and the position of tasks should be changed if necessary. In the $SeqInstance$ which is a small one, by considering Table 5.13 as the incumbent solution and generating two random values, a new table of tasks is created as shown in Figure 5.1. Generally, in this neighbourhood space, three steps are taken as follows, and displayed in Figure 5.1.

1. The first step involves the swapping of the selected tasks by changing their positions in the table of tasks.

2. The second step includes a change in the positions of the related tasks to the selected tasks. The related tasks are the ones with a smaller sequence compared to the picked tasks in the routing diagram. This means that they should be assigned first (the precedence preserving rule).

3. The third step requires the creation of a sequence by adding the remaining tasks (tasks that have not been considered and changed before) in the table of tasks.

In Ns1 for large instances, the number of solutions to be searched in each iteration is 70 and for the small ones, it is 10.

Ns2: This neighbourhood structure is similar to the previous one (Ns1), but instead of having swaps, tasks are transferred; therefore, it is based on $Transfer$. As in Ns1, two random numbers are considered for small instances and eight random numbers are chosen for the large ones. Tasks from the first position (the first random value) are transferred to the second position (the second random value). After that, to satisfy the sequencing constraint, the tasks which are before the transferred task should also be transferred (second step, the precedence preserving rule). The last step, as previously mentioned, is adding the remaining tasks. The number of searching new solutions is the same as Ns1 which is 70 (for large instances) and 10 (for the small ones). Figure 5.2 depicts Ns2 with regard to the given example with two random numbers.

Tasks' Position	Task (Task Name-Model Name-Box Number)	First Step (Swapping Tasks in Positions 5 and 10)	Second Step (Changing Positions of the Related Tasks)	Third Step (Adding the Remaining Tasks)
1	20-A-1			20-A-1
2	10-A-1		10-B-1	10-B-1
3	30-A-1		20-B-1	20-B-1
4	20-A-2		40-B-1	40-B-1
5	10-A-2	30-B-1	30-B-1	30-B-1
6	30-A-2			10-A-1
7	10-B-1			30-A-1
8	20-B-1			20-A-2
9	40-B-1			50-B-1
10	30-B-1	10-A-2	10-A-2	10-A-2
11	50-B-1		30-A-2	30-A-2
12	40-C-1			40-C-1
13	20-C-1			20-C-1
14	40-C-2			40-C-2
15	20-C-2			20-C-2
16	40-C-3			40-C-3
17	20-C-3			20-C-3

Figure 5.1: Ns1, Swap Structure.

Tasks' Position	Task (Task Name-Model Name-Box Number)	First Step (Transferring Task in Position 5 to Position 10)	Second Step (Changing Positions of the Related Tasks)	Third Step (Adding the Remaining Tasks)
1	20-A-1			20-A-1
2	10-A-1			10-A-1
3	30-A-1			30-A-1
4	20-A-2			20-A-2
5	10-A-2			10-B-1
6	30-A-2			20-B-1
7	10-B-1			40-B-1
8	20-B-1			30-B-1
9	40-B-1			50-B-1
10	30-B-1	10-A-2	10-A-2	10-A-2
11	50-B-1		30-A-2	30-A-2
12	40-C-1			40-C-1
13	20-C-1			20-C-1
14	40-C-2			40-C-2
15	20-C-2			20-C-2
16	40-C-3			40-C-3
17	20-C-3			20-C-3

Figure 5.2: Ns2, Transfer Structure.

Ns3: The Ns3 is based on the Transfer structure while using *Intelligence*, which takes advantage of the critical path. The following steps are considered in Ns3. Firstly, the critical path is recognised and tasks of the critical path are transferred one by one to the beginning of the table of tasks. In this way, we give them priority for allocation. Afterwards, the other two steps (2 and 3) explained before are taken into account. This structure is depicted in Figure 5.3. In this Figure, two created neighbours of a current solution based on this structure are generated. The first neighbour in this Figure is formed by transferring all the critical tasks. But the next neighbour is created by transferring some of the critical tasks (all the critical tasks but not the last one). The next neighbour could be another sequence of the critical tasks which are transferred to the beginning of the table of tasks. This process is repeated until all neighbours of a current solution are considered. The number of searched solutions in Ns3 is dependent on the dimension of each problem. But in general, for large instances, it is around 150 solutions on average. Moreover, after transferring tasks in this structure, steps 2 and 3 are also applied to create a feasible sequence of tasks in the related table of tasks.

Tasks' Position	Task (Task Name-Model Name-Box Number)	Tasks of the Critical Path	A Neighbour Solution by Transferring all Critical Tasks and the Related Tasks to Them, then Considering the Remaining Tasks (Steps: 1, 2 and 3)	Another Neighbour Solution by Transferring Some Critical Tasks and the Related Tasks to Them, then Considering the Remaining Tasks (Steps: 1, 2 and 3)
1	20-A-1		20-A-1	20-A-1
2	10-A-1		10-A-1	10-A-1
3	30-A-1	30-A-1	30-A-1	30-A-1
4	20-A-2		20-A-2	20-A-2
5	10-A-2		10-A-2	10-A-2
6	30-A-2	30-A-2	30-A-2	30-A-2
7	10-B-1		10-B-1	10-B-1
8	20-B-1		20-B-1	20-B-1
9	40-B-1		40-B-1	40-B-1
10	30-B-1	30-B-1	30-B-1	30-B-1
11	50-B-1		50-B-1	50-B-1
12	40-C-1		40-C-2	40-C-2
13	20-C-1		20-C-2	20-C-2
14	40-C-2		40-C-3	40-C-1
15	20-C-2	20-C-2	20-C-3	20-C-1
16	40-C-3		40-C-1	40-C-3
17	20-C-3	20-C-3	20-C-1	20-C-3

Figure 5.3: Ns3, Critical Tasks Transfer.

Ns4: The Ns4 *Transfers Critical Path Tasks and Swaps Other Tasks Randomly*. The structure of the transferred critical tasks in this neighbour structure is different from Ns3. This structure transfers two by two the tasks of the critical path to the beginning of the table of tasks and steps 2 and 3 are also applied here as in all other neighbourhood structures. Moreover, by choosing four random numbers in large instances, swamping tasks are considered for four different tasks. For small instances, swap is not applied. This structure searches all the neighbour solutions of a current solution. Figure 5.4 depicts this structure in the given problem.

Tasks' Position	Task (Task Name-Model Name-Box Number)	Tasks of the Critical Path	A Neighbour Solution by Transferring Critical Tasks (Two by Two) and the Related Tasks to Them, then Considering the Remaining Tasks (Steps: 1, 2 and 3)	Another Neighbour Solution by Transferring Critical Tasks (Two by Two) and the Related Tasks to Them, Then Considering the Remaining Tasks (Steps: 1, 2 and 3)
1	20-A-1		20-A-1	20-A-1
2	10-A-1		10-A-1	10-A-1
3	30-A-1	30-A-1	30-A-1	30-A-1
4	20-A-2		10-B-1	40-C-2
5	10-A-2		20-B-1	20-C-2
6	30-A-2	30-A-2	40-B-1	20-A-2
7	10-B-1		30-B-1	10-A-2
8	20-B-1		50-B-1	30-A-2
9	40-B-1		20-A-2	10-B-1
10	30-B-1	30-B-1	10-A-2	20-B-1
11	50-B-1		30-A-2	40-B-1
12	40-C-1		40-C-1	30-B-1
13	20-C-1		20-C-1	50-B-1
14	40-C-2		40-C-2	40-C-1
15	20-C-2	20-C-2	20-C-2	20-C-1
16	40-C-3		40-C-3	40-C-3
17	20-C-3	20-C-3	20-C-3	20-C-3

Figure 5.4: Ns4, Transferring Critical Tasks and Swapping Randomly.

Ns5: The Ns5 *Transfers Critical Path Tasks and Transfers Other Tasks Randomly*. This structure is mainly based on the transferring of tasks and the transfer process is the same as Ns2, but the only difference is associated with the tasks chosen for this process, which are the tasks of the critical path. For this reason, we didn't provide a Figure. However, for small instances, only one transfer is applied and as mentioned earliest which is just for the tasks of the critical path. In large instances, two transfers are done, one for the critical tasks and the other

for other tasks. The number of search processes in this structure is the same as Ns1 and Ns2.

The VND-MSeq method starts with all the neighbourhood structures and applies them to the current solution. Then, the best solution is chosen as the incumbent one. The search process stays on the structure that gives the best answer and searches that neighbourhood until an improvement is obtained. Otherwise, all other structures are applied to the current solution and then, the search moves toward the best one. This process repeats until the number of defined iterations, which is 50, is attained. In broad lines, the process is the same as the VND process which was explained in Chapter 4.

5.3 Proposed Solution Method: GA-MSeq

Considering the literature review and given the type of problem, we decided to use Genetic Algorithms in this part as another solution approach. The solution method is called Genetic Algorithm Mixed-Model Sequencing ($GA-MSeq$). According to Pezzella et al. [2008], GA have been successfully used to solve FJSSP, as proven by the increasing number of papers on the topic. Different dispatching rules such as those in the previous section are used for generating initial populations but in different ways. All the procedures are explained in details in the following order: First, creating the initial populations and second, the improvement procedure which is GA.

5.3.1 GA-MSeq: Initial Population

The first step of applying GA is to generate the initial population. In order to enable proper exploration of the search space, the set of initial solutions should be as diverse as possible. As completely randomised, task sequences may result in infeasible solutions. Hence, it is advisable to employ a constructive heuristic. The initial populations are created by using 4 dispatching rules which were explained in the Section 5.2.1 (LNS, CP, LPT, SPT). Based on these 4 methods, tasks are sorted in 7 different ways as in the previous solution method (VND-MSeq). The

assignment of tasks to the operators and machines starts from the first task of *Table of Tasks* and continues to the next one by considering their order. By this method, 7 solutions are generated and we named them as main solutions. However, we decided to generate 70 solutions as the initial population for large instances and 28 for small instances in the GA-MSeq method. To reach another solution, after creating one main solution based on each dispatching rule, changes should be made in the table of tasks by modifying tasks positions to create new solutions. Still, large and small problems in this section are designed as in the previous method (dealing with the number of operators).

The changes for creating new solutions are made by swapping tasks. To this end, 8 random numbers are considered for swapping 8 tasks. To create a sequence of tasks which be feasible, the last two steps in creating initial solutions in Section 5.2.2 are applied in this part. According to those rules, the related tasks of the picked tasks are considered, and if it is needed, they are transferred to the new position for satisfying the constraint of model routing. After that, the remaining tasks are put in the same position in the new table of tasks. Each new solution is created from the last one by randomly changing the position of tasks, as mentioned earlier.

To create 70 solutions in large instances, 9 other results are created from each main solution and in small instances, 3 results are created from each main solution to have 28 solutions. Each solution should be feasible taking into account all the constraints which are given in the optimisation model, but not considering the available time as mentioned in the previous method. At this step, it is not necessary to respect the available time since they are initial populations. Also, as mentioned earlier, the input data in this Chapter are the output data of the previous Chapter and the feasibility of the results is checked in advance. To sum up, the general process for producing initial populations is shown in Figure 5.5.

5.3.2 GA-MSeq: Genetic Algorithm

GA are a population based metaheuristic. Beside generating initial populations, it is fundamental/critical to encode the solutions (chromosomes). The encoding

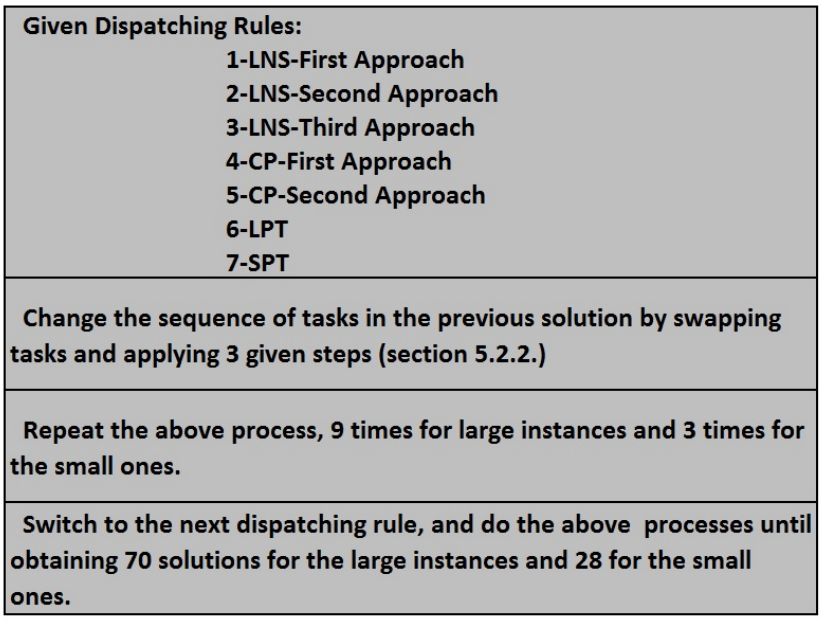

Figure 5.5: Generating Initial Populations.

procedure varies and should depend on the problem under study. There are various types of encoding such as permutation encoding, binary encoding and other types. In continue, the solution encoding, the selection process, the genetic operators, the evaluation function and GA conditions are elucidated.

Solution encoding: In this study, the solution encoding approach is based on permutation. The characterisation of the type of the genes that are composed of the chromosomes is related to the particular problem. Since different tasks of various boxes and models are assigned to proper operators and machines, each gene has a format of five elements (i.e. task, model, box, operator, machine). Moreover, the position of genes shows their order for the assignment by taking into account the earliest starting time. The earliest starting time depends on two issues; first, the position of a task in its model routings and second, the free time of the associated operator and machine. According to the given example in Section 5.2.1, we have shown:

1. Task assignment in Table 5.3;

2. Initial solution in Table 5.13;

The chromosome related to this solution is depicted in Figure 5.6. The chromosome in this solution method works the same as the table of tasks in the earlier solution method.

Workstation N.er	(Operato, Machine)	(Task, Model, Box)	S: Time
1	(Op2, Ma2)	(10, B, 1)	0
2	(Op1, Ma1)	(20, B, 1)	30
2	(Op1, Ma1)	(20, A, 1)	80
2	(Op1, Ma1)	(20, A, 2)	160
4	(Op4, Ma3)	(40, B, 1)	80
1	(Op2, Ma2)	(10, A, 1)	160
1	(Op2, Ma2)	(10, A, 2)	240
3	(Op6, Ma4)	(30, B, 1)	110
4	(Op4, Ma3)	(40, C, 1)	0
4	(Op4, Ma3)	(40, C, 2)	30
4	(Op4, Ma3)	(40, C, 3)	110
3	(Op6, Ma4)	(30, A, 1)	230
3	(Op6, Ma4)	(30, A, 2)	310
5	(Op4, Ma5)	(50, B, 1)	160
2	(Op1, Ma1)	(20, C, 1)	240
3	(Op6, Ma4)	(20, C, 2)	160
3	(Op6, Ma4)	(20, C, 3)	350

Figure 5.6: A Chromosome for the *SeqInstance* (Given Example).

Genetic operators: Genetic operators are important for combining the existing solutions with others and generating diversity. A crossover operator has the ability to generate new solutions by combining other solutions and a mutation operator has the duty of creating diversity. The crossover operation is used to create a pair of children chromosomes from a pair of parent chromosomes. Selection of parents from a population is a random process which is explained in the following. The number of children chromosomes is usually between 0.6 and 1.0 of the dimension of the population. The mutation operation creates another solution from one single solution. Choosing a solution for the mutation is also a random process similar to the crossover and the number of the crossover answers in each iteration is usually

between 0.4 and 0.2 of the population size. As a result, selection of solutions and crossovers and mutations are involved in GA, as explained below.

Selection process: This process includes both the selection of a gene and a chromosome to apply different operators such as crossover and mutation. Moreover, the selection is also used for reproduction in the next step. In the proposed GA-MSeq, two types of selection methods are used:

1. Randomised Selection: As the name implies, this method is completely randomised and the GA-MSeq chooses random values which could identify a chromosome or clarify the position of a gene.

2. Tournament Selection: In reproduction new population, the chromosomes are sorted according to the fitness. Then, the individual representing the highest fitness (smallest makespan) is the winner.

The randomised selection is applied for choosing parents for crossover, picking chromosome for mutation and selecting genes for applying crossovers and mutations. Further, the tournament selection is used for selecting the chromosomes for reproduction in the next iteration.

Crossover and mutation: Crossovers and mutations are applied while taking into account the sequencing problem. Two-point crossover is used in two different ways and mutation is also applied in three varied manners based on *Inversion* and *Insertion*. More details are presented below:

Two-point crossover-1: First, after choosing the parents based on the selection process, as mentioned above, since there are two points in this type of crossover, two random numbers between zero and maximum number of available genes should be generated. These two numbers determine the positions of two genes. The genes of each parent between two selected points are transferred to each child, in the same positions as the parents. Precedence preserving rule is used to avoid infeasibility. The precedence preserving rule changes the position of genes in a way to avoid infeasibility considering the routing diagram of the task in the selected chromosome. Other genes of each child are the same as those of other parents which were not used

before in this child. This type of crossover and the applied rule are explained in detail by the given example ($SeqInstance$) in Figures 5.7 and 5.8. Figure 5.7 shows two chosen chromosomes and the genes between two points which are positions 6 and 12 in color in $SeqInstance$. Figure 5.8 depicts the offspring and the cells with dots are the ones whose position is changed according to the precedence preserving rule.

First Chromosome

Workstation N.er	(Operator, Machine)	(Task, Model, Box)
2	(Op1, Ma1)	(20, A, 1)
2	(Op1, Ma1)	(20, A, 2)
1	(Op2, Ma2)	(10, A, 1)
1	(Op2, Ma2)	(10, A, 2)
4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)
4	(Op4, Ma3)	(40, C, 2)
3	(Op6, Ma4)	(20, C, 2)
4	(Op4, Ma3)	(40, C, 3)
3	(Op6, Ma4)	(20, C, 3)
1	(Op2, Ma2)	(10, B, 1)
2	(Op1, Ma1)	(20, B, 1)
4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)
3	(Op6, Ma4)	(30, A, 1)
3	(Op6, Ma4)	(30, A, 2)

Second Chromosome

Workstation N.er	(Operator, Machine)	(Task, Model, Box)
1	(Op2, Ma2)	(10, B, 1)
2	(Op1, Ma1)	(20, B, 1)
4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)
2	(Op1, Ma1)	(20, A, 1)
1	(Op2, Ma2)	(10, A, 1)
3	(Op6, Ma4)	(30, A, 1)
2	(Op1, Ma1)	(20, A, 2)
1	(Op2, Ma2)	(10, A, 2)
3	(Op6, Ma4)	(30, A, 2)
4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)
4	(Op4, Ma3)	(40, C, 2)
3	(Op6, Ma4)	(20, C, 2)
4	(Op4, Ma3)	(40, C, 3)
3	(Op6, Ma4)	(20, C, 3)

Figure 5.7: Chromosomes of Two-Point Crossover-1.

Two-point crossover-2: In this crossover, two points are selected after choosing parents. The genes of the first and second points are copied to each other's child, considering the precedence preserving rule. According to this rule, if a gene is copied to a child, then its related genes, which are of the same model and box, are also copied by considering the sequencing constraint of that model routing. Figures 5.9 and 5.10 display this type of crossover. The chromosomes are shown in Figure 5.9, which are similar to the chromosomes of the previous type of crossover. Moreover, the selected genes are coloured (genes in positions 6 and 12). Offspring are provided in Figure 5.10 and the cells with dots are those whose positions are changed according to the precedence preserving rule.

First Child

Workstation N.er	(Operator, Machine)	(Task, Model, Box)
4	(Op4, Ma3)	(40, C, 2)
3	(Op6, Ma4)	(20, C, 2)
4	(Op4, Ma3)	(40, C, 3)
3	(Op6, Ma4)	(20, C, 3)
1	(Op2, Ma2)	(10, B, 1)
2	(Op1, Ma1)	(20, A, 1)
1	(Op2, Ma2)	(10, A, 1)
3	(Op6, Ma4)	(30, A, 1)
2	(Op1, Ma1)	(20, A, 2)
1	(Op2, Ma2)	(10, A, 2)
3	(Op6, Ma4)	(30, A, 2)
4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)
2	(Op1, Ma1)	(20, B, 1)
4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)

Second Child

Workstation N.er	(Operator, Machine)	(Task, Model, Box)
2	(Op1, Ma1)	(20, A, 1)
1	(Op2, Ma2)	(10, A, 1)
3	(Op6, Ma4)	(30, A, 1)
2	(Op1, Ma1)	(20, A, 2)
4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)
4	(Op4, Ma3)	(40, C, 2)
3	(Op6, Ma4)	(20, C, 2)
4	(Op4, Ma3)	(40, C, 3)
3	(Op6, Ma4)	(20, C, 3)
1	(Op2, Ma2)	(10, B, 1)
2	(Op1, Ma1)	(20, B, 1)
4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)
1	(Op2, Ma2)	(10, A, 2)
3	(Op6, Ma4)	(30, A, 2)

Cells with dots are the genes which are transferred because of the precedence preserving rule. They are from the parent that gives that child the genes which are between two coloured points. In other words, they are from the first parent of each child.

Figure 5.8: Offspring of Two-Point Crossover-1.

First Chromosome

Workstation N.er	(Operator, Machine)	(Task, Model, Box)
2	(Op1, Ma1)	(20, A, 1)
2	(Op1, Ma1)	(20, A, 2)
1	(Op2, Ma2)	(10, A, 1)
1	(Op2, Ma2)	(10, A, 2)
4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)
4	(Op4, Ma3)	(40, C, 2)
3	(Op6, Ma4)	(20, C, 2)
4	(Op4, Ma3)	(40, C, 3)
3	(Op6, Ma4)	(20, C, 3)
1	(Op2, Ma2)	(10, B, 1)
2	(Op1, Ma1)	(20, B, 1)
4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)
3	(Op6, Ma4)	(30, A, 1)
3	(Op6, Ma4)	(30, A, 2)

Second Chromosome

Workstation N.er	(Operator, Machine)	(Task, Model, Box)
1	(Op2, Ma2)	(10, B, 1)
2	(Op1, Ma1)	(20, B, 1)
4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)
2	(Op1, Ma1)	(20, A, 1)
1	(Op2, Ma2)	(10, A, 1)
3	(Op6, Ma4)	(30, A, 1)
2	(Op1, Ma1)	(20, A, 2)
1	(Op2, Ma2)	(10, A, 2)
3	(Op6, Ma4)	(30, A, 2)
4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)
4	(Op4, Ma3)	(40, C, 2)
3	(Op6, Ma4)	(20, C, 2)
4	(Op4, Ma3)	(40, C, 3)
3	(Op6, Ma4)	(20, C, 3)

Figure 5.9: Chromosomes of Two-Point Crossover-2.

First Child

Workstation N.er	(Operator, Machine)	(Task, Model, Box)
2	(Op1, Ma1)	(20, A, 2)
1	(Op2, Ma2)	(10, A, 2)
4	(Op4, Ma3)	(40, C, 2)
3	(Op6, Ma4)	(20, C, 2)
4	(Op4, Ma3)	(40, C, 3)
2	(Op1, Ma1)	(20, A, 1)
1	(Op2, Ma2)	(10, A, 1)
3	(Op6, Ma4)	(30, A, 1)
3	(Op6, Ma4)	(20, C, 3)
1	(Op2, Ma2)	(10, B, 1)
2	(Op1, Ma1)	(20, B, 1)
4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)
4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)
3	(Op6, Ma4)	(30, A, 2)

Second Child

Workstation N.er	(Operator, Machine)	(Task, Model, Box)
2	(Op1, Ma1)	(20, A, 1)
1	(Op2, Ma2)	(10, A, 1)
3	(Op6, Ma4)	(30, A, 1)
2	(Op1, Ma1)	(20, A, 2)
4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)
1	(Op2, Ma2)	(10, A, 2)
3	(Op6, Ma4)	(30, A, 2)
4	(Op4, Ma3)	(40, C, 2)
3	(Op6, Ma4)	(20, C, 2)
1	(Op2, Ma2)	(10, B, 1)
2	(Op1, Ma1)	(20, B, 1)
4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)
4	(Op4, Ma3)	(40, C, 3)
3	(Op6, Ma4)	(20, C, 3)

Cells with dots are the genes which are transferred because of the precedence preserving rule . They are from the parent that gives that child the genes which are in two coloured points. In other words, they are from the first parent of each child.

Figure 5.10: Offspring of Two-Point Crossover-2.

Mutation inversion: In general a mutation is any type of changes in a chromosome. However, this mutation, after choosing a chromosome selects two mutation points randomly and reverses order of the genes between these two points. However, the rule of precedence preserving is applied in a way to avoid creating an infeasible solution. Figure 5.11 illustrates this type of mutation (mutation inversion).

Mutation insertion: The second mutation is based on the insertion of genes from the first position to the second one, considering the sequence of tasks in the selected model routing (precedence preserving rule). In large instances, 4 random numbers are chosen for applying 2 insertions and in small instances, only 1 insertion is applied. Figure 5.12 depicts this mutation. In this Figure, the gene in yellow is transferred to the position of the gene in green and the related genes are also transferred.

Mutation insertion-critical genes: The basic issues in this mutation are the same as the previous one, but with one difference. The change is associated with the selection of the genes for insertion. In this mutation, the first and second genes

Chromosome			Chromosome - After Mutation		
Workstation N.er	(Operator, Machine)	(Task, Model, Box)	Workstation N.er	(Operator, Machine)	(Task, Model, Box)
2	(Op1, Ma1)	(20, A, 1)	2	(Op1, Ma1)	(20, A, 1)
2	(Op1, Ma1)	(20, A, 2)	2	(Op1, Ma1)	(20, A, 2)
1	(Op2, Ma2)	(10, A, 1)	1	(Op2, Ma2)	(10, A, 1)
1	(Op2, Ma2)	(10, A, 2)	1	(Op2, Ma2)	(10, A, 2)
4	(Op4, Ma3)	(40, C, 1)	4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)	1	(Op2, Ma2)	(10, B, 1)
4	(Op4, Ma3)	(40, C, 2)	2	(Op1, Ma1)	(20, B, 1)
3	(Op6, Ma4)	(20, C, 2)	4	(Op4, Ma3)	(40, C, 3)
4	(Op4, Ma3)	(40, C, 3)	3	(Op6, Ma4)	(20, C, 3)
3	(Op6, Ma4)	(20, C, 3)	4	(Op4, Ma3)	(40, C, 2)
1	(Op2, Ma2)	(10, B, 1)	3	(Op6, Ma4)	(20, C, 2)
2	(Op1, Ma1)	(20, B, 1)	2	(Op1, Ma1)	(20, C, 1)
4	(Op4, Ma3)	(40, B, 1)	4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)	3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)	5	(Op4, Ma5)	(50, B, 1)
3	(Op6, Ma4)	(30, A, 1)	3	(Op6, Ma4)	(30, A, 1)
3	(Op6, Ma4)	(30, A, 2)	3	(Op6, Ma4)	(30, A, 2)

Cells with dots are the genes which are inverted based on the given rule.

Figure 5.11: Mutation Inversion.

Chromosome			Chromosome - After Mutation		
Workstation N.er	(Operator, Machine)	(Task, Model, Box)	Workstation N.er	(Operator, Machine)	(Task, Model, Box)
2	(Op1, Ma1)	(20, A, 1)	2	(Op1, Ma1)	(20, A, 1)
1	(Op1, Ma1)	(20, A, 2)	1	(Op1, Ma1)	(20, A, 2)
1	(Op2,Ma2)	(10, A, 1)	1	(Op2,Ma2)	(10, A, 1)
2	(Op2, Ma2)	(10, A, 2)	2	(Op2, Ma2)	(10, A, 2)
4	(Op4, Ma3)	(40, C, 1)	4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)	3	(Op6, Ma4)	(20, C, 2)
4	(Op4, Ma3)	(40, C, 2)	4	(Op4, Ma3)	(40, C, 3)
3	(Op6, Ma4)	(20, C, 2)	3	(Op6, Ma4)	(20, C, 3)
4	(Op4, Ma3)	(40, C, 3)	1	(Op2,Ma2)	(10, B, 1)
3	(Op6, Ma4)	(20, C, 3)	2	(Op1, Ma1)	(20, B, 1)
1	(Op2,Ma2)	(10, B, 1)	2	(Op1, Ma1)	(20, C, 1)
2	(Op1, Ma1)	(20, B, 1)	4	(Op4, Ma3)	(40, C, 2)
4	(Op4, Ma3)	(40, B, 1)	4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)	3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)	5	(Op4, Ma5)	(50, B, 1)
3	(Op6, Ma4)	(30, A, 1)	3	(Op6, Ma4)	(30, A, 1)
3	(Op6, Ma4)	(30, A, 2)	3	(Op6, Ma4)	(30, A, 2)

Cells with dots are the genes which are inserted based on the given rule.

Figure 5.12: Mutation Insertion.

should be chosen from the critical ones which are related to the critical tasks and they should be transferred to the beginning of the picked chromosome. Therefore, by this method, we give priority to the critical tasks for the assignment. Moreover, genes are transferred considering their routing diagram. The mutation process is generally the same as the previous mutation but obviously with a fundamental difference as mentioned. Therefore, we do not explain it with more example. Below, more details are provided.

1. A chromosome is chosen randomly.
2. The critical genes of that chromosome are determined.
3. Two random numbers from the critical genes are generated.
4. The selected genes are transferred to the beginning of the picked chromosome.
5. The sequencing diagram of the selected genes is considered. If there are any genes which should be assigned before them, they have to be considered. This means that they should be transferred before the selected genes to the new position, which is the beginning of the picked chromosome.

Evaluation function and GA conditions: The evaluation function determines the quality of a candidate solution. Therefore, the fitness evaluation is based on the objective function which is the minimisation of the maximum completion time (makespan or Cmax). The population size for large instances is 70 chromosomes and for small ones, it includes 28. The percentage of the crossover is 90% or 0.9 and the mutation percentage is 40% or 0.4. As mentioned above, there are two different crossovers and three varied mutations. Thus, the given percentages are divided among them but not equally. GA-MSeq runs for 50 iterations unless the populations converge to a solution earlier, which will stop GA-MSeq.

5.4 Conclusions

In this Chapter, the focus was on solving Mixed-model Assembly Line Sequencing Problems (MALSP). For this reason, an optimisation model was developed. It

allowed us to better understand the problem and solve small instances. Besides, the model could be easily adapted to another industrial case. However, the main contributions are the two proposed approximate solution methods. One of them is based on VND and the other is based on GA. In both methods, different dispatching rules are used for generating the initial solution and populations. General rules of both solution methods are depicted in Figures 5.13 and 5.14.

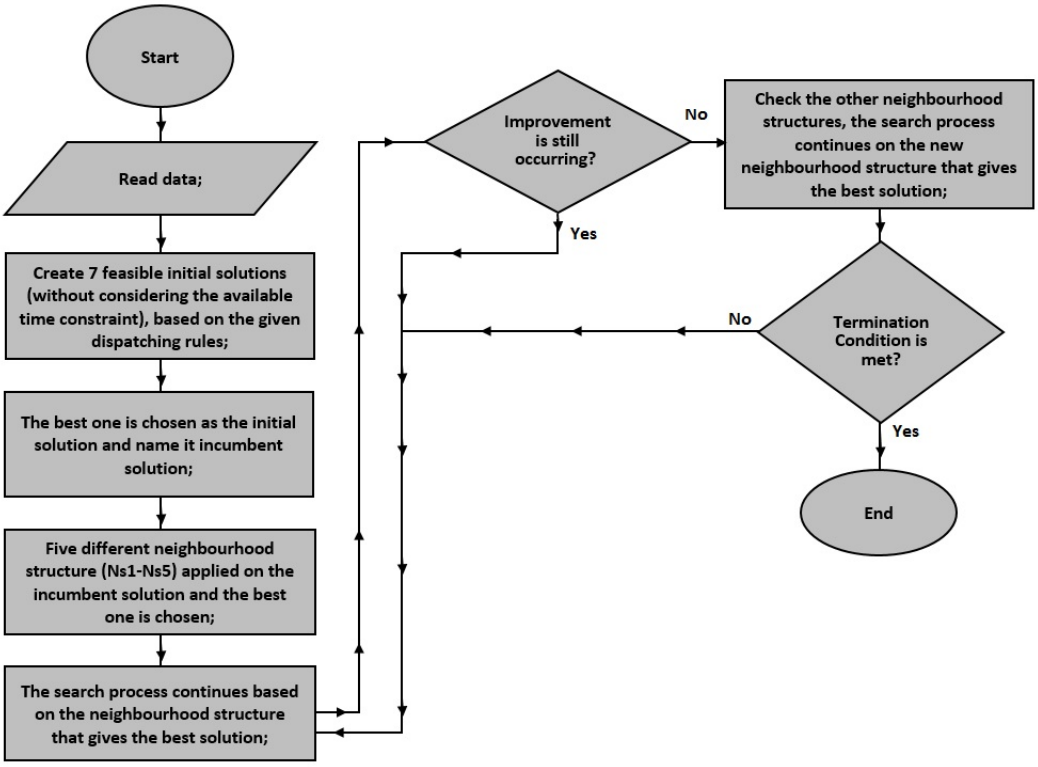

Figure 5.13: General Rules of VND-MSeq Solution Method.

Both approaches were compared in various ways and in different types of instances, from generated ones to those based on real data. Additionally, the optimisation model was tested in small instances. Such results may be found in Chapter 7. Nevertheless, we can already say that in general, the VND-MSeq solution method is faster than GA-MSeq, but the GA-MSeq works better than VND-MSeq in terms of the quality of the solutions.

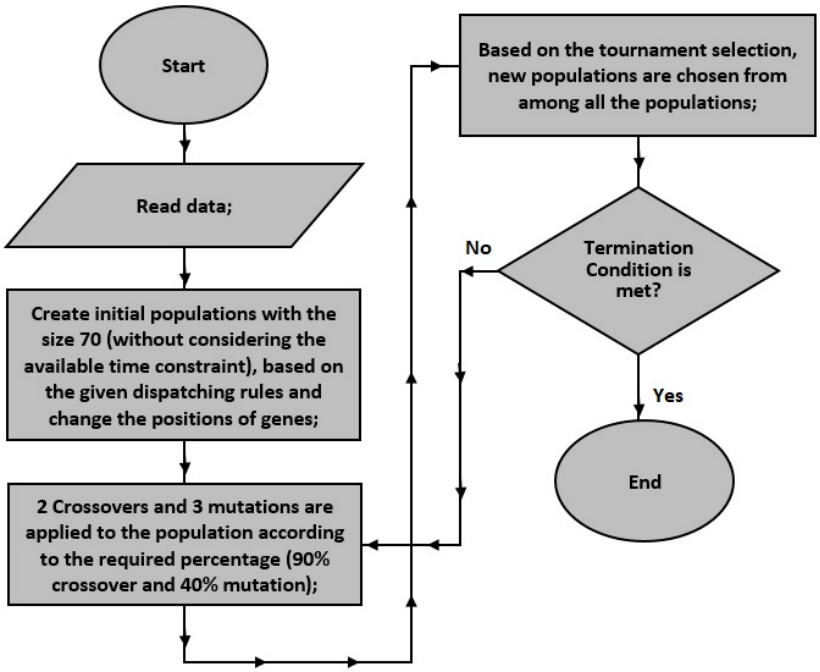

Figure 5.14: General Rules of GA-MSeq Solution Method.

Chapter 6

Simultaneous Balancing and Sequencing Mixed-model Assembly Systems

This Chapter focuses on the presentation of Simultaneous Balancing and Sequencing Mixed-Model Assembly Problems ($MALB\&SP$). An optimisation model is presented and programmed in CPLEX Optimisation Studio. However, for reasons similar to those described in the previous two chapters, the large size of the problems prevents their exact resolution. The methods chosen for solving the real instances are based on GA. Two types of problems are studied: multi-objective and single-objective. The multi-objective problem has two objectives, one is the objective of the balancing problem which was in Chapter 4, and the other is the objective of the sequencing problem which was in Chapter 5. The single-objective problem considers the mentioned objectives together, with weights to combine both objectives in order to obtain a single-objective formulation. These issues are explained in more details in the next section.

In what concerns MOP, a non-dominated sorting based Multi-Objective Evolutionary Algorithm ($MOEA$), called Non-Dominated Sorting Genetic Algorithm II ($NSGA-II$), is used. The method is able to search for a representative set

of pareto-optimal solutions. MOEA ranks solutions according to how much they contribute to the objectives which are considered in a research. The number of publications related to the MOEA area has increased over the recent years. Besides, MOEA has been a real success story, and Coello et al. [2007] is one of the major references in this area. However, one of the most outstanding methods among all MOEAs is NSGA-II. This method, as the first goal, prefers all non-dominated solutions to the dominated ones, and as the second goal, among non-dominated solutions, it prefers the extreme solution(s) with a larger distance to the other ones.

The single-objective method is a simple GA, with appropriate adaptations to the given problems.

In addition, for both developed methods, different dispatching rules are used to create the initial populations.

6.1 Optimisation Model

As mentioned, since this study concentrates on a daily line balancing and sequencing issue, WIP is also considered. There are different models of shoes with diverse quantities in boxes, and the boxes move on any direction on the line. Each task inside each model routing requires a certain processing time. Moreover, the resources, operators and machines are included and the operators are classified according to their skills and machines are of various types. In order to simplify the proposed model, the types of the machines are omitted, but a constraint indicating whether a machine may perform a task or not, is included. Therefore, in this formulation, machines are directly linked to the tasks without considering their types.

In Chapter 4, the main issue was balancing with the objective minimisation of the number of workstations (the first part) and smoothing the operators' workload (the second part). Chapter 5 contemplates sequencing with the objective minimisation of makespan. Since this chapter considers simultaneous balancing and sequencing, both objectives of the two other chapters are established as the multiple- and single-objectives, which is depicted in Figure 6.1.

Objective: In this optimisation model, the problem was addressed as single-

<table>
<tr><th colspan="4">MALB&SP</th></tr>
<tr><th colspan="2">Multi-objective</th><th colspan="2">Single-objective</th></tr>
<tr><td>First Objective</td><td>Minimising the Number of Workstations + Smoothing Operators' Workload (Balancing Objective)</td><td rowspan="2">One Objective</td><td rowspan="2">Minimising the Number of Workstations + Smoothing Operators' Workload (Balancing Objective) + Minimising the Maximum Completion time of the Line (Sequencing Objective)</td></tr>
<tr><td>Second Objective</td><td>Minimising the Maximum Completion time of the Line (Sequencing Objective)</td></tr>
</table>

Figure 6.1: The Considered MALB&SP Problems and Objectives.

objective with only one objective consisting of three mentioned parts in Figure 6.1. This will be explained in the following in more detail. Moreover, the following indices, parameters and decision variables are considered in the mixed integer optimisation model.

Indices:

j: index for model, $j \in [J]$, $[J] = \{1, 2, \ldots, J\}$

i: index for task, $i \in [I]$, $[I] = \{1, 2, \ldots, I\}$

k: index for number of boxes, $k \in [K]$, $[K] = \{1, 2, \ldots, K\}$

o: index for operator, $o \in [O]$, $[O] = \{1, 2, \ldots, O\}$

m: index for machine, $m \in [M]$, $[M] = \{1, 2, \ldots, M\}$

Input parameters:

P_{ijk} = processing time of task i for model j of box k

S_{ijk} = sequence of task i for model j of box k

EC_{ijk} = earliest completion time of task i for model j of box k

$OT_{oi} = 1$: if operator o can perform task i; 0: otherwise

$MT_{mi} = 1$: if machine m can perform task i; 0: otherwise

MW = minimum number of needed workstations

Av = available time

GM = great number

Υ = weight in the objective function

α = weight in the objective function

Decision variables:

$X_{ijko} = 1$: if task i of model j of box k is assigned to operator o; 0: otherwise

$Y_{ijkm} = 1$: if task i of model j of box k is assigned to machine m; 0: otherwise

C_{ijk}: completion time of task i of model j of box k

$U_{ijki'j'k'} = 1$: if completion time of task i' of model j' of box k' is greater than completion time of task i of model j of box k; 0: otherwise

$Z_{om} = 1$: if operator o is assigned to machine m; 0: otherwise

W_o: workload of each operator o

$\bar{W}$: average workload of operators

$Cmax$: the longest completion time

Optimisation model:

The objective function is the minimisation of the number of workstations (the first part), the smoothing of the operators' workload (the second part) and the minimisation of the makespan (the third part).

$$Minimise \left(\sum_{o=1}^{O} \sum_{m=1}^{M} Z_{om} + \frac{\Upsilon}{O} \sum_{o=1}^{O} |W_o - \bar{W}| + \alpha Cmax \right)$$

Parameter Υ is the same as the Chapter 4 and places a lower weight on the second part of the objective function. The α is another parameter used to make the weight of the third part equal to the first part. The values of Υ and α for different instances regarding their sizes, are changed. This issue will be explained in Chapter 7, together with the results of the proposed model.

The **constraints** are established in the following:

1. Task i of model j of box k should be assigned to the required operator o:

$$OT_{oi} - X_{ijko} \geq 0, \quad i \in [I], \quad j \in [J], \quad k \in [K], \quad o \in [O]$$

2. Task i of model j of box k should be assigned to exactly one operator o, if its processing time is greater than zero:

$$\sum_{o=1}^{O} X_{ijko} = 1, \quad i \in [I], \quad j \in [J], \quad k \in [K], \quad P_{ij} \neq 0$$

3. Task i of model j of box k should be assigned to the required machine m:

$$MT_{mi} - Y_{ijkm} \geq 0, \quad i \in [I], \quad j \in [J], \quad k \in [K], \quad m \in [M]$$

4. Task i of model j of box k should be assigned to exactly one machine m, if its processing time is greater than zero:

$$\sum_{m=1}^{M} Y_{ijkm} = 1, \quad i \in [I], \quad j \in [J], \quad k \in [K], \quad P_{ijk} \neq 0$$

5. The sequence of any task i for each model j in box k should be respected:

$$C_{i'jk} \geq C_{ijk} + P_{i'jk}, \quad i, i' \in [I], \quad j \in [J], \quad k \in [K], \quad S_{i'j} > S_{ij}$$

6. Each operator o must work on only one task i of model j of box k at any time (tasks of the same operator do not overlap in time):

$$C_{ijk} + GM(2 - X_{ijko} - X_{i'j'k'o} + U_{ijki'j'k'}) \geq C_{i'j'k'} + P_{ijk},$$

$$i, i' \in [I], \quad j, j' \in [J], \quad k, k' \in [K], \quad o \in [O]:$$

$$(j \neq j') \quad \vee \quad (j = j' \quad \wedge \quad k \neq k')$$

$$C_{i'j'k'} + GM(3 - X_{ijko} - X_{i'j'k'o} - U_{ijki'j'k'}) \geq C_{ijk} + P_{i'j'k'},$$

$$i, i' \in [I], \quad j, j' \in [J], \quad k, k' \in [K], \quad o \in [O]:$$

$$(j \neq j') \quad \vee \quad (j = j' \quad \wedge \quad k \neq k')$$

7. Each machine m must work on only on one task i of model j of box k at any time (tasks of the same machine do not overlap in time):

$$C_{ijk} + GM(2 - Y_{ijkm} - Y_{i'j'k'm} + U_{ijki'j'k'}) \geq C_{i'j'k'} + P_{ijk},$$

$$i, i' \in [I], \quad j, j' \in [J], \quad k, k' \in [K], \quad m \in [M]:$$

$$(j \neq j') \quad \vee \quad (j = j' \quad \wedge \quad k \neq k')$$

$$C_{i'j'k'} + GM(3 - Y_{ijkm} - Y_{i'j'k'm} - U_{ijki'j'k'}) \geq C_{ijk} + P_{i'j'k'},$$

$$i, i' \in [I], \quad j, j' \in [J], \quad k, k' \in [K], \quad m \in [M]:$$

$$(j \neq j') \quad \vee \quad (j = j' \quad \wedge \quad k \neq k')$$

8. The completion time of each task i of model j of box k should be less than the available time:

$$C_{ijk} \leq Av, \quad i \in [I], \quad j \in [J], \quad k \in [K]$$

9. The completion time of each task i of model j of box k cannot be less than the earliest completion time:

$$C_{ijk} \geq EC_{ijk}, \quad i \in [I], \quad j \in [J], \quad k \in [K]$$

10. Cmax (MakeSpan) is the longest completion time:

$$C_{ijk} \leq Cmax, \quad i \in [I], \quad j \in [J], \quad k \in [K]$$

11. If task i of model j of box k is assigned to operator o and to machine m then operator o is allocated to machine m:

$$X_{ijko} + Y_{ijkm} - 1 \leq Z_{om}, \quad i \in [I], \quad j \in [J], \quad k \in [K], \quad o \in [O], \quad m \in [M]$$

12. Operators' workload:

$$W_o = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} X_{ijko} P_{ijk}, \quad o \in [O]$$

13. Average operators' workload:

$$\bar{W} = \frac{1}{MW} \sum_{o=1}^{O} W_o$$

The 'minimum' number of workstations (MW) required is calculated by the same expression as in Chapter 4, where D_j represents the demand of model j:

$$MW = \frac{\sum_{i=1}^{I} \sum_{j=1}^{K} \sum_{k=1}^{K} P_{ijk}}{Av}$$

The optimisation model was programmed into the CPLEX optimisation Studio. However, the model is not applied directly to the actual data of the company due to the corresponding dimension of the problems. As in Chapters 4 and 5, the model is used to better understand the problem and also evaluate the approximate method.

The method to solve this single-objective problem is GA and for the multi-objective, NSGA-II is applied. They will be explained in the next sections. Efficient genetic representations and operator schemes are also provided. When designing the schemes, we take into account specific features of the problems.

6.2 Multi-Objective Solution Method: NSGA-II

According to the type of problem, which is simultaneously balancing and sequencing, a multi-objective approach seems to be a good idea. As stated, the large dimension of the given problems made their optimal resolution unrealistic. So, this is the reason for the development of such an approximate but relevant method. The NSGA-II algorithm can be found in Reed et al. [2003], Coello et al. [2007], Sheng et al. [2015] and Chatterjee et al. [2017]. By considering the literature review, in particular the authors like Gen and Lin [2014], who claim that NSGA-II gives good solutions in some testing problems, we decided to use NSGA-II. More details about the method are presented in this section.

The solution method proposed (NSGA-II) is based on GA and resorts to a constructive heuristic to generate initial populations. The constructive heuristic employs different dispatching rules. GA improve the initial solutions by considering different crossovers and mutations of each concentrate on balancing and sequencing. The approach which is non-dominated sorting-based is used for choosing new populations among all available populations in different iterations. Besides, this

technique helps to create different pareto fronts. Figure 6.2 shows an example having different pareto fronts.

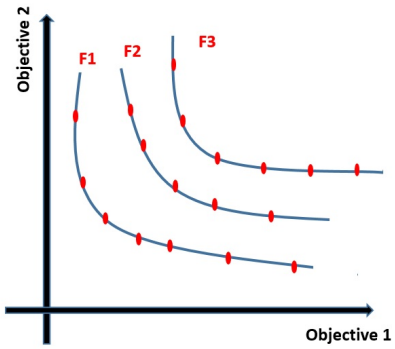

Figure 6.2: Different Pareto Fronts (F).

NSGA-II is very similar to the simple GA in terms of form. The crossover and mutation operators remain as usual and only the selection process is manipulated. Before applying the selection procedure, the population is ranked based on the non-dominated sorting approach which will be described in the next section. All solutions not dominated by any other solutions are determined as the first front. All solutions only dominated by solutions in the first front are assigned as the second front, and so on. Selection is made using the contest between two solutions. The solution with the lowest front number is selected if the two solutions are from different fronts. If they are from the same front, then another concept called Crowding Distance (CD) is involved in this method. The solutions with the highest CD are selected if they are from the same front. The CD gives priority to the solutions with a larger distance to other ones. The details are presented ahead. Figure 6.3 depicts the way the NSGA-II works. In this Figure, the rejected solutions from $F4$ have lower CD than the selected ones.

The CD value of a solution gives approximate density of solutions. Moreover, the CD value of a specific solution is its average distance compared to two neighbour solutions. The method used to calculate the CD for a solution is illustrated in Figure 6.4. In this Figure, the points on the curve are the populations and it is intended to calculate the CD value of point p. The two neighbours of this point are $p-1$ and

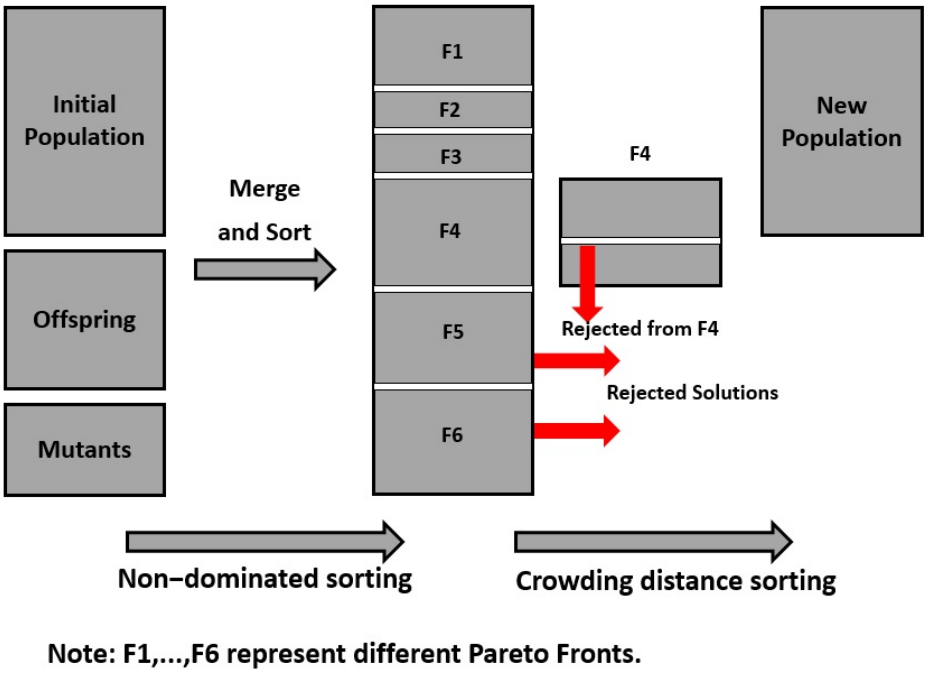

Figure 6.3: How NSGA-II works.

$p+1$, and points 1 and 2 are the first and last points in this population or curve. The formula to calculate CD value for point p is provided below:

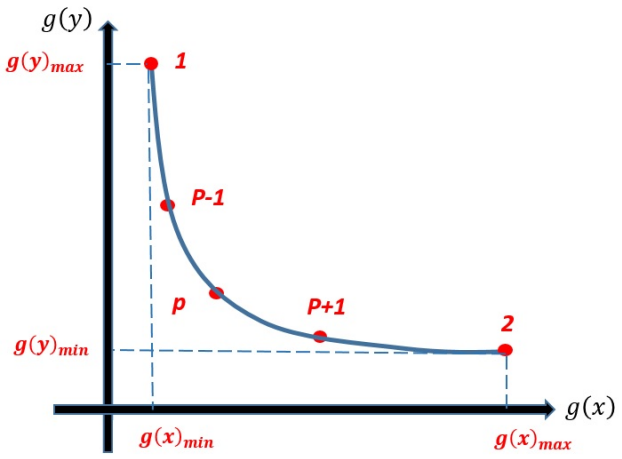

Figure 6.4: Crowding Distance (CD) Sample.

$$CD_x^p = \frac{|g_x^{(p+1)} - g_x^{(p-1)}|}{g_x^{(max)} - g_x^{(min)}}, \qquad CD_y^p = \frac{|g_y^{(p+1)} - g_y^{(p-1)}|}{g_y^{(max)} - g_y^{(min)}}$$

$$CD^p = CD_x^p + CD_y^p$$

The way CD is used in the given method will be explained in detail via GA in section 6.2.2.2.

6.2.1 NSGA-II: Non-dominated Sorting-based Approach (NS)

There are some solutions which are superior to the rest of solutions in the search space in a multi-objective optimisation problem when all objectives are taken into account but are inferior to other solutions in the space in one or more objectives. Pareto-optimal solutions or n $non-dominated$ solutions are considered among these solutions. The rest of the solutions are known as *dominated* solutions.

The idea behind the non-dominated sorting procedure is that a ranking selection method is used to emphasise good points. The algorithm is developed based on this concept. The concept of *dominance* is clearly shown in Figure 6.5 with a graph. It is a minimisation problem with two objectives. Point Z is better than all points in B area and this point (Z) is worse than all points in A area. Therefore, one could say that Z dominates all the points in B but all the points in A dominate Z. Considering the other areas which are C and D, we couldn't say easily what the relation between them is. Since none of the solutions in the non-dominated set is absolutely better than others, any of them could be an acceptable solution. Selection of one solution over the others may require more information.

For ranking the solution based on NS method, two items are needed: one is a set and the other is a value. They have been provided below:

1. S_p : is a set of members in the population in which p dominates;
2. n_p : is a number of solutions which dominate p;

In this method, after having these two sets, different pareto fronts could be calculated, and as mentioned before, we named them $F_1, F_2, ...F_r, ...F_R$. The r is

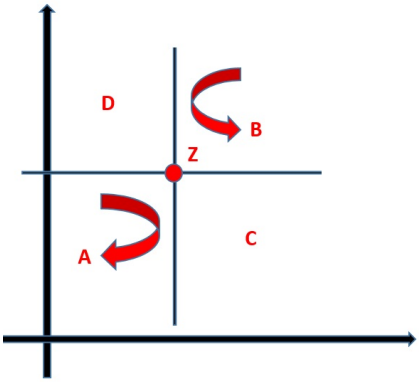

Figure 6.5: Dominance Concept.

an index for fronts, $r \in \{1, 2, \ldots, R\}$. The way NS works is explained in Figure 6.6. In a broad view, this algorithm in lines between 1 and 11 has been calculated as the S and n for all the members in the population. After that, in lines between 12 and 31, different fronts are determined.

In detail, after calculating the solutions which are allocated in the first front, the NS Algorithm eliminates their effect on the other solutions. This means that after having the first front and without considering the solutions in that front, we again calculate the first front by having the rest of the solutions. This first new front is not really a first front but the second front. This process is defined in lines 16 to 22.

6.2.2 NSGA-II: Genetic Algorithm

A GA was used in the previous chapter for the sequencing problem and in this chapter, it is used in two contexts: multi-objective and single-objective. As mentioned, the multi-objective technique is NSGA-II and the single-objective is a simple GA, adapted to the given problem, which will be explained in the following. However, a GA consists of several components which must together reach an efficient search method. We now introduce those parts, their function and how they interact with each other. At first, the initialisation method which creates the initial population is explained and then, the main issues of the GA concerning the given problem are

Figure 6.6: NS Algorithm.

For all solutions in a population: $S_p = \{\}$ and $n_p = 0$;
for *any member of the population such as* v **do**
  for *all other members of the population such as* w **do**
    if v *dominates* w **then**
      w should be added to $S_v = \{\}$ and we have: $S_v = \{w\}$;
    end
    if w *dominates* v **then**
      we should add a number to n_v and we have: $n_v = 1$;
    end
  end
end
Put a counter of the fronts equal to one: $r = 1$;
Define the current front as equal to an empty set, such as $F_r = \{\}$;
while F_r *is empty* **do**
  for *all the populations such as* v **do**
    if $n_v = 0$ **then**
      v should be added to the current front which is F_r;
      we consider S_v;
      for *any member of* S_v *such as* w *(*$S_v = \{w\}$*)* **do**
        we should reduce one unit from n_w and we have $(n_w)new = (n_w)old - 1$;
      end
    end
  end
  if F_r *is not empty* **then**
    For calculating the next front add one unit to $r = r + 1$;
    Go back to the step number 13;
  end
  else
    The NS Algorithm is finished.
  end
end

described.

6.2.2.1 Initial Population

As mentioned in the previous chapter, the initial populations are critical in GA because of their effect on the method performance. The initial populations are a set of solutions which indicate the assignment of varied tasks of different boxes and models to the operators and machines while their sequence is determined. The balancing and sequencing are considered at the same time. Therefore, we integrate the dispatching rule with the GA to increase the performance of the solution procedure. By applying different dispatching methods, we change the order of tasks for assignment and sequencing. As referred, by changing the order of tasks in the input data, we create different *table of tasks*. The methods used for creating the initial populations are the same as in Section 5.2.1 in Chapter 5, which are:

1. Largest Number of Successors (LNS);
2. Critical Path (CP);
3. Longest Processing Time (LPT);
4. Shortest Processing Time (SPT);

Their application for creating tables of tasks is exactly the same as mentioned earlier. The LNS is used in 3 different ways, CP is applied in 2 varied ways and LPT and SPT are used in 1 way each. Therefore, we have 7 main solutions. Moreover, in this section, another constraint is imposed on the problem which is related to the assignment of the tasks to operators and machines while satisfying the balancing rules. One more difference is also considered. The method facilitates a population with more diversity. To reach this goal, the number of used machines and operators and also their order, after creating some initial solutions, are changed to create some other solutions. In this way, we produce diverse solutions. Moreover, the number of initial solutions which are created from each main chromosome or individual are 9 solutions. This allows the creation of 70 initial individuals. Moreover, feasibility

of tables of tasks for creating solutions is satisfied by considering the precedence relation among different tasks. So, the *precedence preserving* rule is considered. Moreover, all the constraints in Section 6.1 are satisfied in generating the first population and in all the NSGA-II runs.

6.2.2.2 Genetic Algorithm, Relevant Issues

The relevant issues and main aspects of GA are defined in the literature review which is provided in Chapter 3 and also in Chapter 5. The solution encoding and the type of chromosomes are exactly the same as in Chapter 5, but the other fundamental components will be explained now.

Selection process: The selection process involves selecting chromosomes for applying different operators, selecting genes of a chromosome for varied operators and also choosing chromosomes, as the new population in different iterations of NSGA-II runs. The way each of them is selected is as follows:

1. Selecting chromosomes for applying different operators: Firstly, it is based on the NS algorithm and secondly, it is according to the CD values of chromosomes. For selecting a chromosome anterior, two solutions among the population are chosen randomly. Then, according to the NS algorithm, the one which is allocated in the lower front is picked. However, if both are in one front, then the one with a higher CD is chosen. The higher CD value demonstrates increasing diversity of solutions.

2. Selecting genes for applying different operators: Based on different crossovers and mutations, one or more points should be chosen. This selection indicates that genes are chosen randomly by having a random number among the available genes in a chromosome.

3. Selecting chromosomes in each NSGA-II run as the new population: This process was briefly explained in Section 6.2. Firstly, according to the NS algorithm, the solutions are allocated in different fronts and then sorted considering their fronts (from the smaller to the larger ones). Next, the solutions

which are in the lower fronts will be chosen according to the number of population. Moreover, if there are solutions in the same fronts and all of them couldn't be chosen according to the limitation on the number of selected solutions, then their CDs are considered. They are sorted according to the CD value, from the higher to the lower ones. Then, solutions with a higher CD will be chosen.

Genetic operators: We take into account a main feature of the problem: simultaneously balancing and sequencing. Genetic operators are defined in such a way that they are suitable to handle both matters. Six crossovers and five mutations types are defined, some concentrate on the balancing and some focus on the sequencing. Therefore, we named them according to their goal. Moreover, all of them preserve feasibility. Let us remember again the structure of the genes, which is the combination of a *task*, a *model*, a *box*, an *operator* and a *machine*, as in Figure 6.7. In this Figure, the first two parts, including an operator and a machine, are related to the balancing problem and the rest three are related to the sequencing issue. Therefore, when the crossovers and mutations are concentrated on the sequencing, then part of the genes which is related to the sequencing (task, model and box) is manipulated according to the given rule. On the other hand, if the crossovers and mutations are related to the balancing, then part of the genes in relation to the balancing (operator and machine) is manipulated.

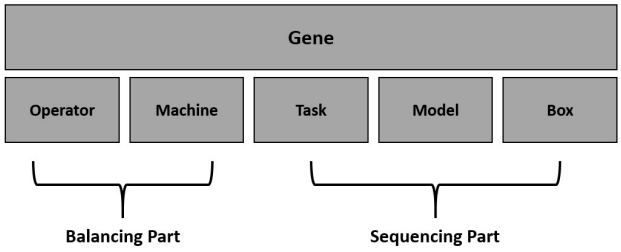

Figure 6.7: Gene Structure.

Crossovers: There are three different crossovers types related to the sequencing and also three crossovers types related to the assignment.

Crossover sequencing-simple: According to the selection process, two par-

ents are chosen and then, two random numbers are picked. Each number determines a gene in a chromosome. Since this crossover concentrates on the sequencing, then the sequencing part of the selected genes of the first parent is transferred to the same position of the first offspring (*step* 1). Moreover, by creating a feasible solution, as mentioned in Section 5.2.2 of Chapter 5, the precedence preserving rule should be satisfied. According to this method, the tasks related to the tasks in the picked genes, which are the ones in the same model routings, should be allocated to the same positions where they were before. Therefore, the genes containing them (the related tasks) should be transferred, but only their sequencing parts (*step* 2). The balancing parts of the transferred genes (assignments) should be the same as the other parent (second parent), which has not been used before in this offspring (*step* 3). Also, the rest of the genes and their positions are the same as the the second parent and similar to the previous phase (*step* 4). The creation of the other offspring follows the same way as the one already mentioned. In general, in this crossover, we change the sequence of some genes and create a new solution. To better understand the problem, an instance named $BalandSeqInstance$ is provided. Table 6.1 illustrates the example with 3 models and different tasks in each model routing. In this example, only one box of each model is available with the amounts of 5, 10 and 6. Figure 6.8 depicts two chromosomes and all the steps taken in this crossover to create offspring. In this crossover, for the small instances (instances with less than 11 operators according to Chapter 5), only 1 random number or gene is selected.

Table 6.1: Model Routings.

Model	Task	Sequence	Processing Time	Order Q.ty
A	10	2	7	5
A	20	1	8	5
A	30	3	4	5
B	10	1	3	10
B	20	2	5	10
B	30	4	5	10
B	40	3	3	10
C	10	1	6	6
C	30	2	4	6

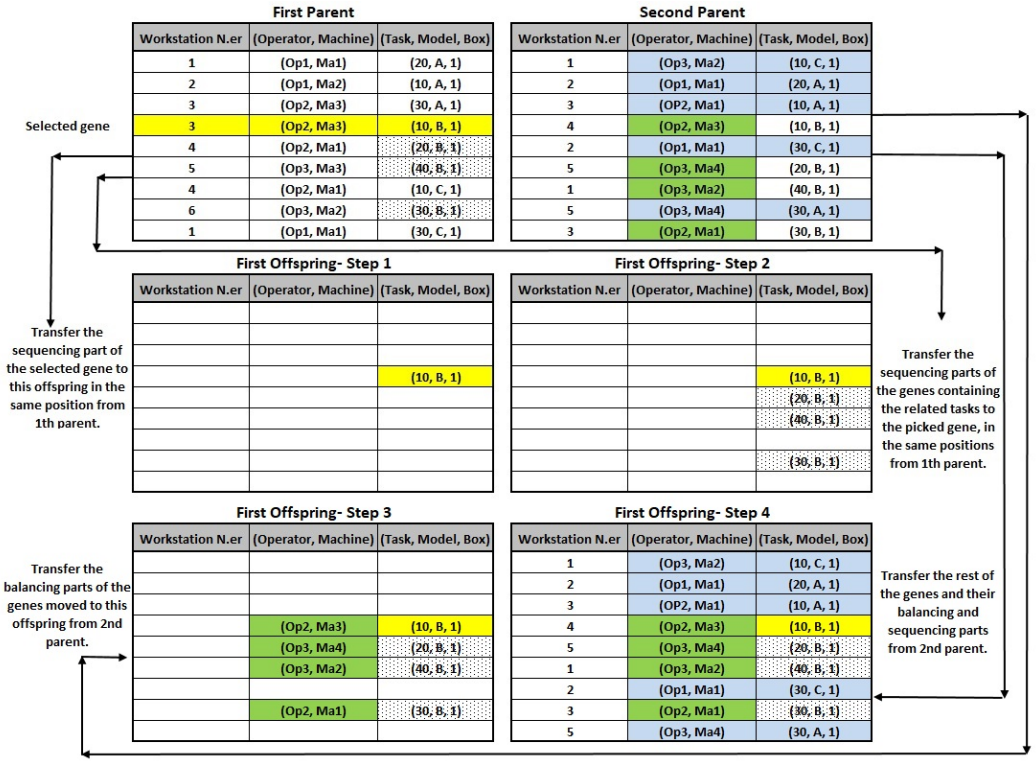

Figure 6.8: Applying Crossover Sequencing-Simple.

Crossover sequencing-critical tasks regarding operators: This crossover concentrates on changing the sequence of the critical genes considering operators. This means that operator(s) which work more than others are the critical ones and the genes related to them are considered critical. Therefore, a critical gene from the beginning of the first parent is picked and then, the tasks related to the task in the picked gene are considered, and all genes with those tasks are transferred to the beginning of the first offspring (*step* 1). As a result, we give them priority in allocation process. This movement continues until the transferred genes become less than a random number which is greater than 0 and less than 30 in the large instances. In the small instances, it is greater than 0 and smaller than 3 (*step* 2). From the other parent, the sequencing part of the other genes with the same order is transferred to an empty place (*step* 3). The balancing part of all genes is also from the first parent (*step* 4). The other offspring is created similarly. Moreover, the whole process is depicted in Figure 6.9, considering the $BalandSeqInstance$

instance.

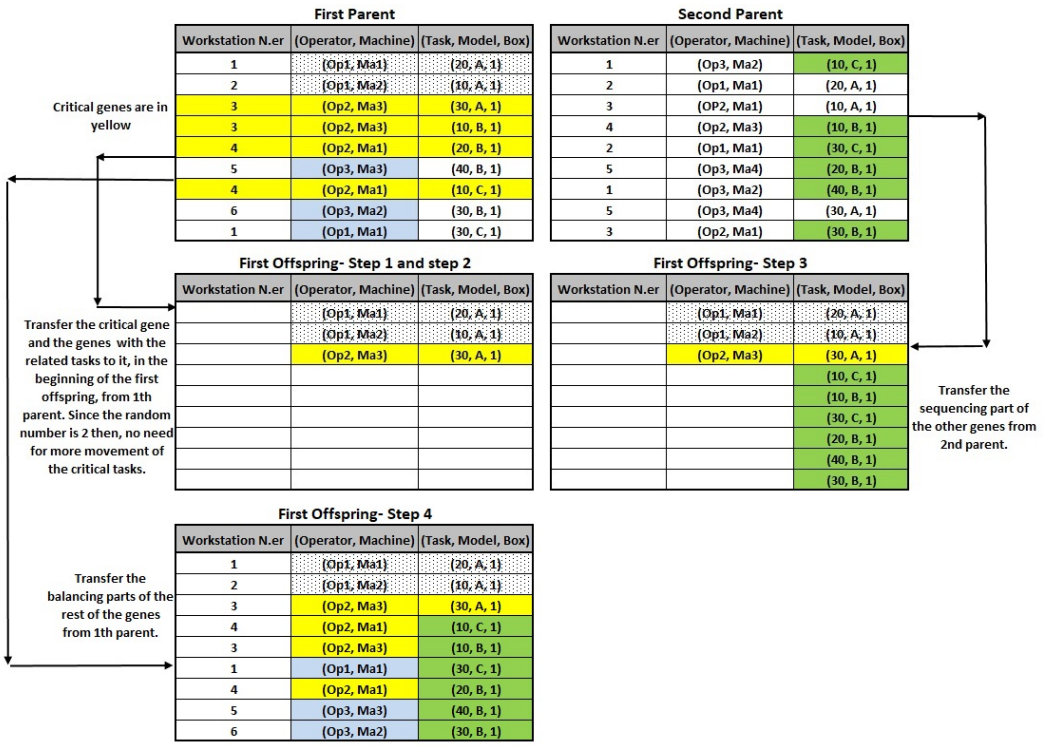

Figure 6.9: Applying Crossover Sequencing-Critical Tasks Regarding Operators.

Crossover sequencing-critical tasks regarding machines: The difference from the previous crossover is associated with the picked genes. The genes containing tasks in the critical machines are picked and the other steps are the same.

Crossover assignment-simple: In this crossover, after choosing two parents, two random numbers are generated which determine two genes. The crucial idea in this crossover is fixing the sequence of genes and only changing their balancing parts. Therefore, the first step is to create an offspring from a parent by transferring only the sequencing parts of all the genes in the same position (*step* 1). In the next phase, the balancing parts of the genes which are before and after the two random numbers are equal to the first parent (*step* 2). The balancing parts of the genes which are between these two generated values are the same as the other parent (*step* 3). All these steps are shown by having the *BalandSeqInstance* instance in Figure 6.10. Moreover, the other offspring is created in the same way.

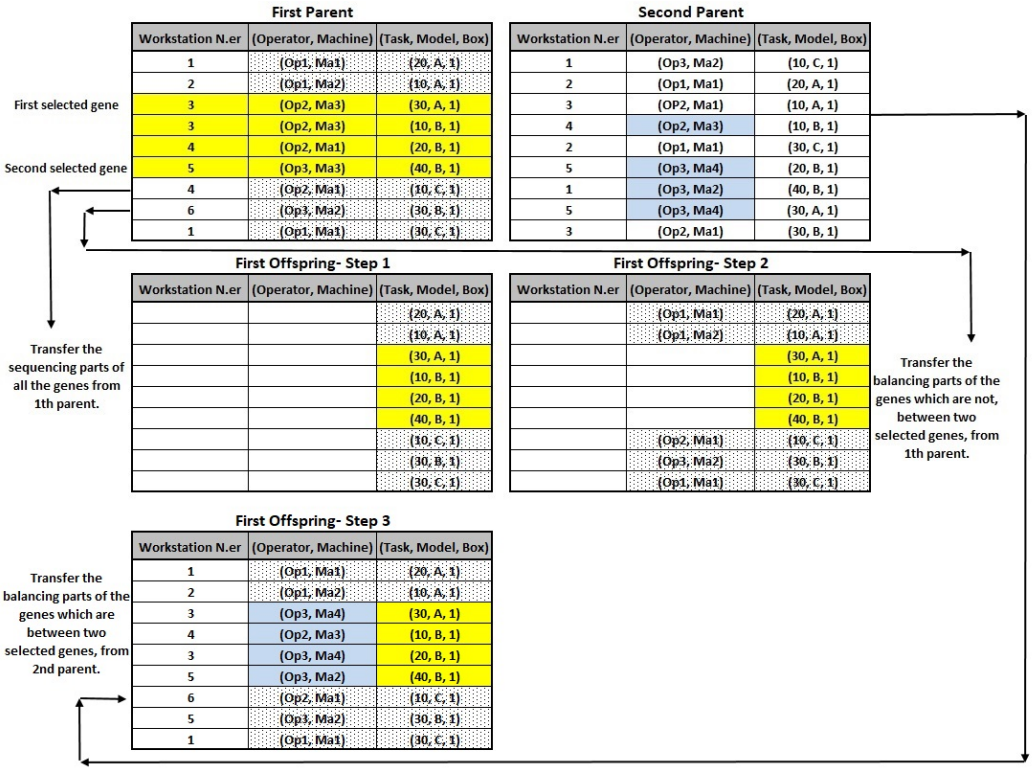

Figure 6.10: Applying Crossover Assignment-Simple.

Crossover assignment-operators with fewer tasks: The operators with fewer tasks are the ones which are working less than the others, and the genes containing them are the main core of this crossover. To find these operators, we need to define a threshold. Therefore, this threshold is equal to a small percentage of available tasks considering models and boxes which are equal to the available genes in a chromosome. This percentage is a random number greater than 0.5% and lower than 1%. Back to the given crossover, after picking two parents, the applied method determines the mentioned operators for both parents separately. Next, an operator among those working less than others (as previously mentioned) is picked randomly and then, the genes with that operator are considered (*step* 1). Moreover, if in an instance, there is not any of this kind of operator, then any operator is picked randomly. Afterwards, an offspring is equal to a parent (*step* 2). Since the focus of this crossover is on the balancing, then the balancing part of

the picked genes in that offspring is changed according to the balancing part of the other parent which has not been used before (*step* 3). This means that the tasks of the critical operators are assigned to other operators and any machines (could be the machine exactly used previously), but the operators and machines should be selected from the other parents. The new operators and machines which are the balancing part should have the ability to perform the tasks. In Figure 6.11, this crossover is explained considering the given instance ($BalandSeqInstance$).

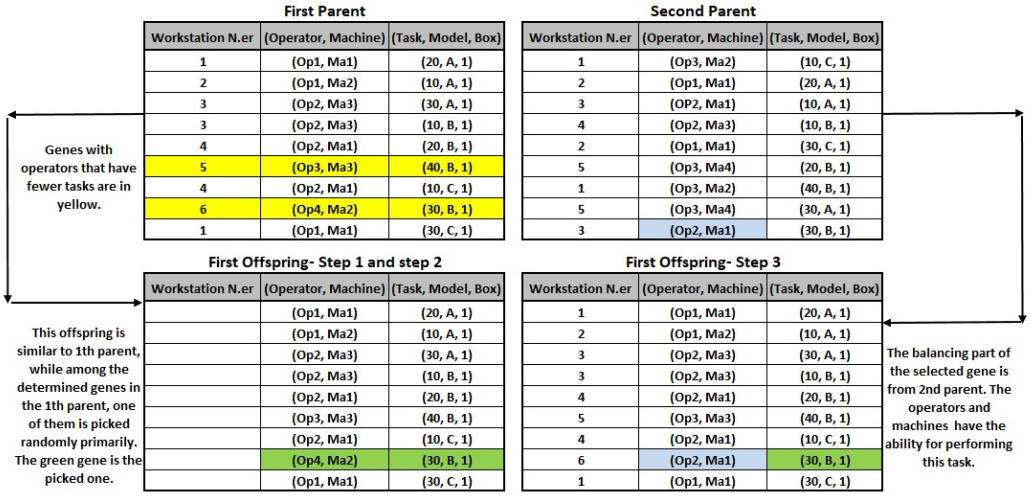

Figure 6.11: Applying Crossover Assignment-Operators with fewer Tasks.

Crossover assignment-machines with fewer tasks: This crossover is the same as the previous one, but considers the genes that contain machines with fewer tasks (instead of operators with fewer tasks). The rest of the process is the same as the earlier crossover. However, the machines in the balancing parts of the genes which are going to be manipulated could not be the same as the machines used before. Therefore, the new ones with the desired ability are considered. Moreover, at the same time the operators could be any one with the desired ability.

Mutations: There are two different types of mutations related to the sequencing and also three mutations types related to the assignment.

Mutation sequencing-critical tasks regarding operators: The concept of critical operators in this mutation is the same as the critical operators in the

crossover sequencing-critical tasks regarding operators, which are the operators working more than others. The tasks which are assigned to these operators are considered in this mutation. Therefore, after choosing a chromosome, according to the mentioned selection rules, the tasks on the critical operators are calculated and thus, the genes containing them are considered. After that, two of them in the large instances and one of them in the small ones are selected randomly. Then, the genes with the tasks related to the picked one which are the tasks in the same model routings are calculated. The next step is to change the order of the genes with these tasks and transferring them to the beginning of the new chromosome. The rest of the genes of the new solution are according to the picked chromosome. By this mutation, we randomly insert some of the critical genes in the beginning of that chromosome and create a new solution. Figure 6.12 depicts this mutation in the given example.

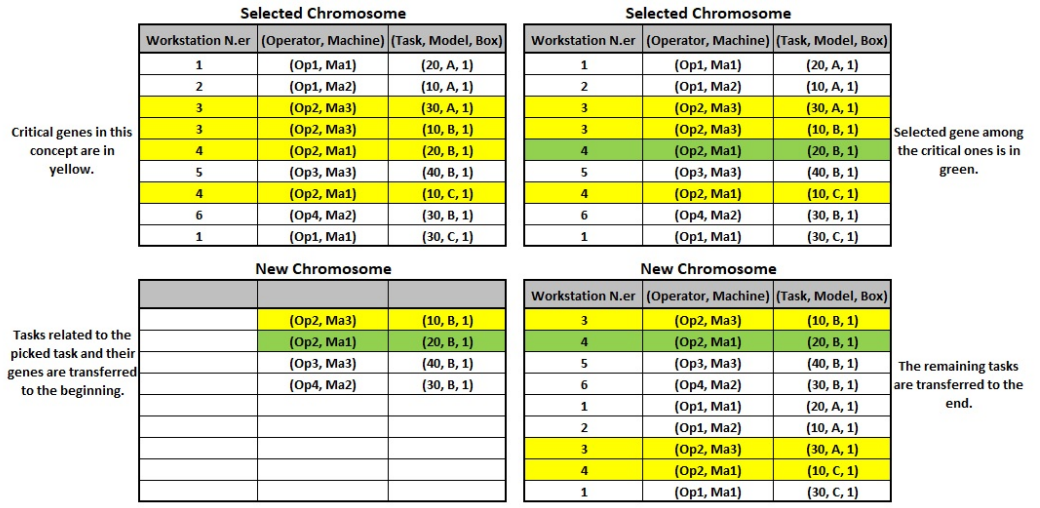

Figure 6.12: Applying Mutation Sequencing-Critical Tasks Regarding Operators.

Mutation sequencing-critical tasks regarding machines: This mutation is the same as the previous one, but the only difference is related to the picked genes. The critical tasks in this mutation are the ones in the critical machines. Therefore, the genes containing them are considered. The critical machines are those working more than others like the critical operators.

Mutation assignment-operators with maximum movement: In this mutation, another solution is created by making changes in the balancing part of some genes of a selected chromosome. The genes which have the following equities are picked in this mutation. Firstly, their operator shifted more than others between the machines. Secondly, their machines worked less than others. After choosing the genes with the mentioned properties, their balancing part will be changed by considering the balancing part of other genes. If the task in any selected gene could be done by new operators and any machines (could be exactly a machine used before) from that chromosome, then changes could be made. This mutation is explained in Figure 6.13.

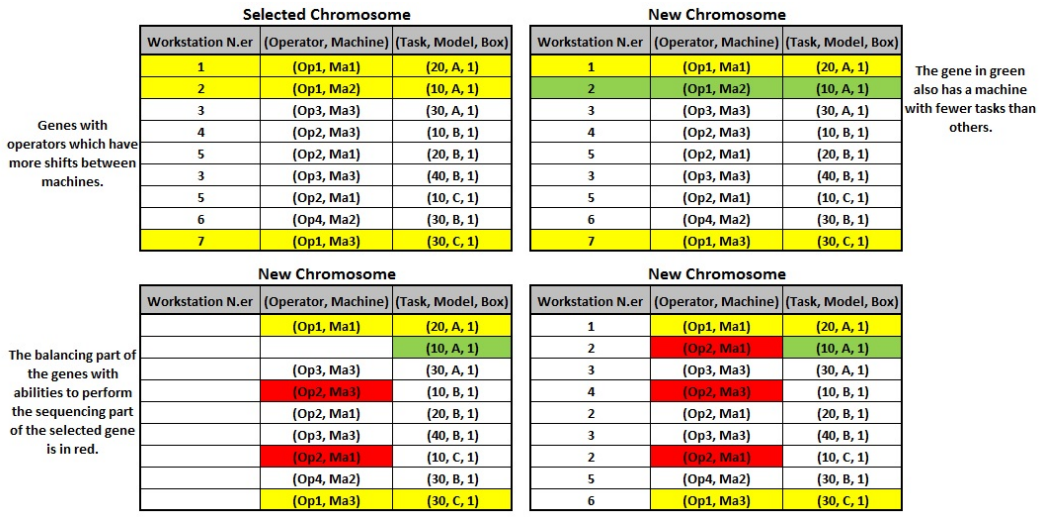

Figure 6.13: Applying Mutation Assignment-Operators with Maximum Movement.

Mutation assignment-operators with fewer tasks: In this mutation, the genes with operators working less than others are picked. If the tasks of the picked genes could be done by new operators and any machines from that chromosome, then their balancing part is changed. Note that the machine could be the one used before. However, the general procedures are the same as the previous mutation, but the only difference is related to the selected genes. Therefore, we did not define more examples here.

Mutation assignment-machines with fewer tasks: This mutation is fo-

cused on the genes with the machine working less than others. After considering those genes, the rest of the mutation process is the same as the earlier mutation. But in the balancing part of the selected gene, the machine could not be the same as the one used before while the operators could be any one with the desired ability (opposite to the previous mutation).

Evaluation function and GA conditions: The mentioned problem is a multi-objective one. The objective of the problem has two dimensions. The first dimension expresses a weighted sum of the number of workstations and the smoothing of the operators' workload. This is the objective of the balancing problem in Chapter 4. The other dimension concerns the sequencing objective, which is the makespan. Fitness evaluation function for the chromosomes includes their fronts combined with their crowding distances. Therefore, solutions with the higher quality or with lower fronts are searched. For each generation, all the chromosomes are evaluated and the best individual is saved to be used in the next iteration. The number of the population includes 70 solutions and the percentages of the crossovers and mutations are 80% or 0.8 and 40% or 0.4. NSGA-II terminates after 50 iterations. This method is tested on different instances in Chapter 7.

6.3 Single-Objective Solution Method: GA

This section addresses the single-objective problem. Therefore, both objectives proposed in the previous sections are joined together by a weight and the evaluation function is the same as the optimisation model:

$$Minimize \left(\sum_{o=1}^{O} \sum_{m=1}^{M} Z_{om} + \frac{\Upsilon}{O} \sum_{o=1}^{O} |W_o - \bar{W}| + \alpha Cmax \right)$$

The procedure essentially follows NSGA-II method, to create initial populations, define solution encoding chromosomes and genetic operators and implement them. The main differences are related to the selection process and fitness function. The selection process is explained below:

1. Selecting chromosomes and genes for applying different operators: The chromosomes and genes selection is the same as the previous chapter, which is done randomly;

2. Selecting chromosomes in each iteration of GA run as the new population: This selection is based on the tournament selection and competency of the solutions, which means that individuals with higher competence are chosen.

Another difference concerns the evaluation function, which was explained earlier in the beginning of this section. This method is used for different instances and the numerical results are evaluated in Chapter 7.

6.4 Conclusions

In this Chapter, we developed two solution approaches, based on GA, to solve multiple- and single-objective Mixed-Model Assembly Line Balancing and Sequencing Problems (MALB&SP). The multi-objective method is NSGA-II and the single-objective is a simple GA, with convenient adaptations. The optimisation model for the simultaneous problems of balancing and sequencing was presented. Both approaches were compared in various ways and in different types of instances, from generated ones to those based on real data. Additionally, the optimisation model was tested in small instances. Such results can be found in Chapter 7.

The GA with one objective enables us to compare both separated and integrated approaches. The general rules of both solution methods are depicted in Figures 6.14 and 6.15. Moreover the extensive computational results in Chapter 7 show that the integrated approach of balancing and sequencing outperforms the separated one. In general, the results demonstrated that the NSGA-II framework is an effective approach to develop efficient algorithms for MALB&SP.

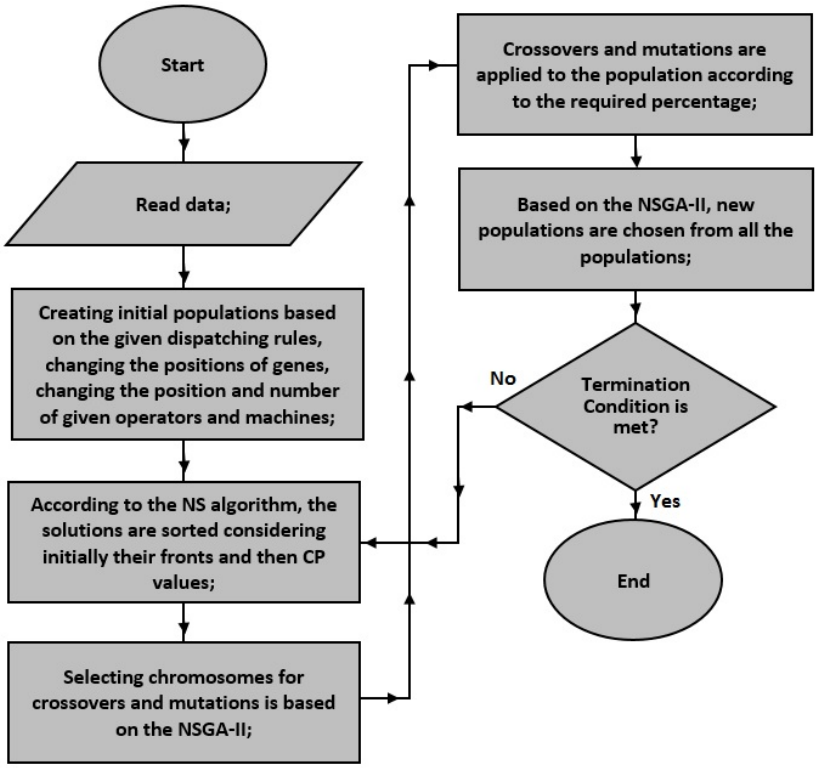

Figure 6.14: General Rules of NSGA-II Solution Method.

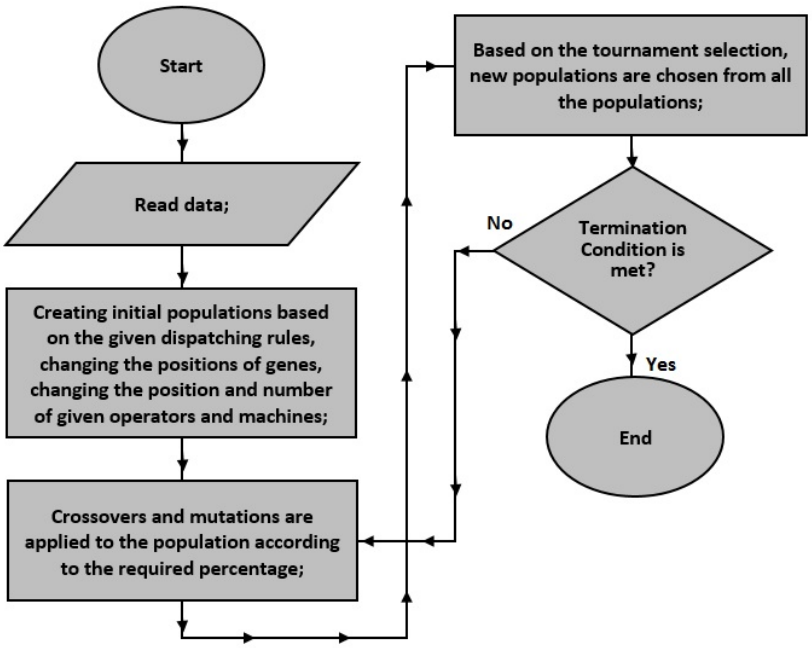

Figure 6.15: General Rules of GA Solution Method.

Chapter 7

Computational Results

Computational experiments were performed in some randomly generated instances and in real data. They were divided into small-size, medium-size and large-size instances. The small sizes are generated, but the medium and large sizes are based on real data from two different lines of the footwear company (see Chapter 2). In one of the real instances from the large-size data set for different problems given in this thesis (included in `www.inescporto.pt/~jsoeiro/Footwear_Balancing`), WIP is involved being treated as new models (this will be explained in Section 7.1).

The sizes of the small instances increase according to the quantity of available resources, such as models, tasks, operators or machines. The optimisation model presented in Chapters 4, 5 and 6 reaches optimal solutions by using the Solver IBM ILOG CPLEX optimisation Studio, version 12.6. In those optimisation models there are two weights, Υ and α. There are chosen in a way to give equal weights to the number of workstations and Cmax and less weight to the smoothing indicator.

The computational tests were run on an Intel (R) Core (TM) i5-5200U CPU with 8 Gbs of random access memory. All the defined methods are implemented in C++ and compiled with Microsoft Visual Studio 2008, running until the termination conditions were achieved, according to each mentioned rules. This chapter is divided into three main sections, which are related to balancing, sequencing and then simultaneous balancing and sequencing problems. The solution methods developed to solve these problems are tested in different types of bodies, as mentioned

above. Finally, in the last section of the chapter, comparisons between the two problems, i.e. balancing and sequencing ($MALBP$ and then $MALSP$), ($MALB\&SP$) are provided separately or simultaneously.

7.1 Balancing Results

This section contains the results based on the balancing optimisation model when applied to the small instances and then, the results of the $ASBsm$ and $RPW-VNDbal$ based on the small, medium and large instances are presented. Given that the optimal solutions of the real instances are not known, a Lower Bound (LB) for the objective function in Chapter 4 is calculated. Furthermore, as mentioned previously, simulation aided our analysis of the validity of the solutions obtained by ASBsm and RPW-VNDbal to effectively meet the established production plans. The simulation was conducted using the AnyLogic 7.3.6 Personal learning edition.

This section features the results of the computational tests performed on the 8 generated instances and the 14 real instances. Tables 7.1 and 7.2 summarise the small-size instances by using two given methods. The computational times are times in seconds. The *Gap_BalOpt* is calculated using the following:

$$Gap_BalOpt = \left(\frac{ImprovedObjective - OptimalObjective}{OptimalObjective}\right) \times 100$$

Here, *ImprovedObjective* in each table is the objective value of the final solution achieved by using each developed method (ASBsm and RPW-VNDbal). *OptimalObjective* is the optimal solution determined by the optimisation model.

According to Table 7.1, ASBsm ends the optimal solution in 5 instances and has a *Gap_BalOpt* of less than 1% for one instance and for the other two instances, it is less than 18%, which is a high value. In these instances (number 2 and 7), the operator's abilities are diverse compared to the number of available operators. The number of models, available boxes and tasks has a significant impact on the running times. The number of workstations required by ASBsm is equal to the optimal solution in 6 instances.

Table 7.1: Computational Results of the Balancing Problems for the Small-size Instances, Using the Optimisation Model and ASBsm.

Instance	N.er Models	N.er Boxes	N.er Tasks	N.er Operators	N.er Machines	N.er Machine Types	Initial Solution				Improved Solution				Optimal Solution				
							N.er Operators	N.er Machines	Objective	Time(s)	N.er Operators	N.er Machines	Objective	Time(s)	N.er Operators	N.er Machines	Objective	Time(s)	Gap_BalOpt
S. Size-1	2	3	3	4	3	3	3	3	3.750	0.04	2	2	3.000	0.05	2	2	3.000	0.131	0.0%
S. Size-2	3	5	4	6	7	4	5	5	6.258	0.06	4	4	5.073	0.08	3	3	4.322	0.146	17.3%
S. Size-3	3	8	4	4	5	4	4	5	5.965	0.06	4	4	4.965	0.09	4	4	4.965	0.155	0.0%
S. Size-4	4	6	5	4	6	5	4	5	6.999	0.04	3	3	4.995	0.06	3	3	4.995	0.139	0.0%
S. Size-5	4	13	5	6	6	4	5	5	7.333	0.07	4	4	6.213	0.11	4	4	6.166	0.251	0.8%
S. Size-6	6	34	6	9	10	5	5	5	6.606	0.12	4	4	5.606	0.24	4	4	5.606	0.543	0.0%
S. Size-7	7	47	9	10	12	7	7	7	10.710	0.26	5	5	8.101	0.79	4	4	7.405	2.523	9.4%
S. Size-8	9	54	11	10	12	7	9	9	11.971	0.25	6	6	8.898	0.86	6	6	8.898	4.136	0.0%

In the following, the results of RPW-VNDbal based on small instances are given. According to Table 7.2, RPW-VNDbal finds the optimal solution in 6 instances and has a *Gap_BalOpt* of less than 1% for the other two. The number of workstations required by RPW-VNDbal in all instances is equal to the optimal solution even when the *Gap_BalOpt* is not zero. The only difference occurs in the smoothing function due to other assignments of the types of machines and/or operators. In general, according to the given tables, the RPW-VNDbal works better on the small instances than ASBsm.

LB is calculated based on the procedure used by Scholl and Klein [1999] and Vilarinho and Simaria [2002], but the necessary adaptations are as follows

1. The processing time of each task regarding order quantity is measured and

Table 7.2: Computational Results of the Balancing Problems for the Small-size Instances, Using the Optimisation Model and RPW-VNDbal.

Instance	N.er Models	N.er Boxes	N.er Tasks	N.er Operators	N.er Machines	N.er Machine Types	Initial Solution				Improved Solution				Optimal Solution				
							N.er Operators	N.er Machines	Objective	Time(s)	N.er Operators	N.er Machines	Objective	Time(s)	N.er Operators	N.er Machines	Objective	Time(s)	Gap_BalOpt
S. Size-1	2	3	3	4	3	3	2	2	3.000	0.04	2	2	3.000	0.07	2	2	3.000	0.131	0.0%
S. Size-2	3	5	4	6	7	4	4	4	5.244	0.03	3	3	4.322	0.05	3	3	4.322	0.146	0.0%
S. Size-3	3	8	4	4	5	4	4	5	6.035	0.06	4	4	4.965	0.07	4	4	4.965	1	0.0%
S. Size-4	4	6	5	4	6	5	4	5	6.995	0.05	3	3	4.995	0.08	3	3	4.995	0.139	0.0%
S. Size-5	4	13	5	6	6	4	5	5	7.333	0.08	4	4	6.166	0.08	4	4	6.166	2	0.0%
S. Size-6	6	34	6	9	10	5	4	4	5.627	0.15	4	4	5.627	0.17	4	4	5.606	139	0.4%
S. Size-7	7	47	9	10	12	7	7	7	10.614	0.13	4	4	7.465	0.26	4	4	7.405	153	0.8%
S. Size-8	9	54	11	10	12	7	8	9	11.975	0.23	6	6	8.898	0.44	6	6	8.898	242	0.0%

called $NeededTaskTime$;

2. Each task needs a unique machine type. Therefore, the time required for each type of machine is calculated by adding the $NeededTaskTime$s for the tasks which require the same type of machine. This time is called $NeededTaskTime-foraMachType$;

3. $NeededTaskTimeforaMachType$ is divided by the available time and the value is rounded up to get the number of workstations for this machine type. However, as there are various types of machines, the following rule should be observed:

   (a) If there are machines with more than one type for the selected type of ma-

chine, all combinations of the decimal parts of the $NeededTaskTimefor-aMachType$, related to the multi machine types, should be considered and no workstation should work more than the available time;

4. LB is derived by adding the number of workstations required for all machine types to the smoothing part of the objective function, which depends on the value of parameter Υ (see the optimisation model). Here, the average total time required is defined considering MW (the minimum number of workstations, without considering different machine types, which was calculated in section 4.1 of Chapter 4).

An essential goal is to compare the results obtained by the reality of the company. The input data provided by the company comprise several days. The systems usually have WIP, meaning that various boxes with diverse models and quantities with tasks already executed must be considered. Consequently, each box can have dependent variables and altered model routings. New boxes with maximum number of 10 will be created and then, ASBsm and RPW-VNDbal will run for a one-day production plan, which can last from 8 to 9 hours. The daily data is quite extensive (an average of 230 excel lines for medium-size problems and 1800 excel lines for large-size problems). An algorithm was developed to adapt and prepare data for the format required by two mentioned methods. Table 7.3 summarises general information of different instances which are used in different parts of this chapter.

Table 7.4 summarises the results obtained by ASBsm for medium- and large-size problems considering LB. As the optimal solutions are not known, we seek to evaluate the quality of the solutions by considering two types of indicators. $Improvement_Bal$ is calculated by measuring the difference between the initial objective and the improved objective and dividing it by the initial objective. The Gap_BalLB in this table is measured by calculating the difference between the improved objective and LB and dividing it by the LB. Theses two formulas are given in below.

Table 7.3: Real Instance Information.

Instances	N.er Models	N.er Boxes	N.er Tasks	N.er Operators	N.er Machines	N.er Machine Types	Max Operator Ab.ties
Medium-Size-1	15	127	15	28	41	15	11
Medium-Size-2	11	171	10	26	40	10	8
Medium-Size-3	16	159	15	28	41	14	12
Medium-Size-4	41	209	30	27	42	20	19
Medium-Size-5	35	94	47	17	31	5	39
Large-Size-1	303	334	70	86	146	15	36
Large-Size-2	320	356	62	86	152	16	31
Large-Size-3	298	335	68	86	148	14	36
Large-Size-4	326	377	73	86	143	15	36
Large-Size-5	344	399	69	86	148	14	36
Large-Size-6	185	354	85	86	141	14	41
Large-Size-7	196	401	70	86	141	14	33
Large-Size-8	157	283	77	86	144	14	42
Large-Size-9	148	291	93	86	143	14	45

$$Improvement_Bal = \left(\frac{InitialObjective - ImprovedObjective}{InitialObjective} \right) \times 100$$

$$Gap_BalLB = \left(\frac{ImprovedObjective - LB}{LB} \right) \times 100$$

There is a minimum $Improvement\ _Bal$ of 9% while the maximum is 31%, and the average is 18%. ASBsm exhibits acceptable Gap_BalLB considering LB: the minimum Gap_BalLB is 3%, the maximum is 20% and the average is 12%. These values seem reasonable for a problem of this size.

The same comparison is done by using the RPW-VNDbal method and having LB. Table 7.5 outlines the results.

There is a minimum $Improvement_Bal$ of 14% while the maximum is 29%, and the average is 19%. RPW-VNDbal displays acceptable Gap_BalLB considering LB, which is better than ASBsm: the minimum Gap_BalLB is 0%, the maximum is 21% and the average is 9%. Moreover, a comparison between these two methods are shown in Table 7.6. These values indicated that the RPW-VNDbal method gives us better results considering all instances.

Table 7.4: Computational Results of the Balancing Problems Using ASBsm for Real Instances considering LB.

Instances	Initial Solution				Improved Solution					LB		
	N.er Operators	N.er Machines	Objective	Time(s)	N.er Operators	N.er Machines	Objective	Time(s)	Improvement_Bal	Workstations	Objective	Gap_BalLB
Medium-Size-1	14	14	16.271	0.15	9	9	11.139	0.58	31%	8	10.139	10%
Medium-Size-2	9	9	11.063	0.15	8	8	10.040	1.04	9%	7	9.040	11%
Medium-Size-3	13	13	14.946	0.17	10	10	11.877	2.50	21%	8	9.903	20%
Medium-Size-4	15	17	19.326	0.31	10	11	13.293	0.76	31%	9	11.229	18%
Medium-Size-5	11	12	13.163	0.54	10	11	11.953	1.57	9%	9	10.480	14%
Large-Size-1	59	61	79.094	5.03	46	48	65.995	108.74	17%	42	59.945	10%
Large-Size-2	58	58	76.113	5.23	48	48	65.935	130.95	13%	42	59.847	10%
Large-Size-3	64	64	81.726	5.06	45	45	62.650	130.16	23%	43	60.556	3%
Large-Size-4	64	64	80.726	6.48	52	52	68.280	171.83	15%	45	61.185	12%
Large-Size-5	63	63	81.088	6.58	48	48	66.017	143.56	19%	43	60.970	8%
Large-Size-6	68	68	82.187	3.65	55	55	68.958	195.83	16%	49	62.550	10%
Large-Size-7	64	64	82.566	3.21	52	52	68.461	226.94	22%	43	59.438	10%
Large-Size-8	66	67	82.127	2.75	54	55	70.0343	130.38	15%	49	63.931	10%
Large-Size-9	679	70	85.489	2.24	54	55	70.422	117.57	18%	46	61.265	15%

However, it is very important for the company to access comparative results and to identify the advantages of having automatic optimisation means in line balancing. It is worth noting that the solutions implemented by the company often accommodate some sequencing when, for instance, a line manager assigns boxes to free workstations. As specified in the previous section, ASBsm and RPW-VNDbal ran successively with the available time decreasing in intervals of 5%. After that, the simulation verifies if all boxes can be produced. The results of ASBsm method depicted in Table 7.7 and the results of the RPW-VNDbal methods are shown in Table 7.8. Both tables, with the instances as before, show that the production plans could be satisfied in 80% of the available time for medium-size instances and 60% of the available time for large-size problems. The *Gain_Bal*, in those tables, is calculated by measuring the disparity between the company reality and the improved objective, and dividing it by the company reality, as below.

Table 7.5: Computational Results of the Balancing Problems Using RPW-VNDbal for Real Instances considering LB.

Instances	Initial Solution				Improved Solution					LB		
	N.er Operators	N.er Machines	Objective	Time(s)	N.er Operators	N.er Machines	Objective	Time(s)	Improvement_Bal	Workstations	Objective	Gap_BalLB
Medium-Size-1	11	11	13.141	0.20	9	9	11.139	0.54	15%	8	10.139	10%
Medium-Size-2	9	9	11.063	0.18	7	7	9.050	0.99	18%	7	9.040	0%
Medium-Size-3	13	13	14.946	0.23	9	9	10.888	2.88	27%	8	9.903	10%
Medium-Size-4	14	15	17.268	0.40	9	10	12.275	0.62	29%	9	11.229	9%
Medium-Size-5	12	14	14.678	0.36	11	12	12.629	0.55	14%	9	10.480	21%
Large-Size-1	64	64	82.090	7.58	45	45	63.027	61.96	23%	42	59.945	5%
Large-Size-2	60	60	78.092	8.01	47	47	64.931	91.34	17%	42	59.847	8%
Large-Size-3	62	62	79.715	7.69	45	45	62.659	64.78	21%	43	60.556	3%
Large-Size-4	64	64	80.757	9.88	50	50	66.533	101.26	18%	45	61.185	9%
Large-Size-5	55	57	75.084	10.44	47	48	65.994	132.04	12%	43	60.970	8%
Large-Size-6	68	69	83.310	8.32	54	54	68.084	97.06	18%	49	62.550	9%
Large-Size-7	64	67	83.595	6.97	47	49	65.533	75.25	22%	43	59.438	10%
Large-Size-8	67	68	83.110	6.33	54	55	69.983	115.79	16%	49	63.931	9%
Large-Size-9	67	69	84.516	5.89	52	53	67.463	71.60	20%	46	61.265	10%

Table 7.6: A Comparison Between ASBsm and RPW-VNDbal Methods for the Balancing Problems considering the LB.

Method	Improvement_Bal			Gap_BalLB		
	Min	Max	Average	Min	Max	Average
ASBsm	9%	31%	18%	3%	20%	12%
RPW-VNDbal	14%	29%	19%	0%	21%	9%

$$Gain_Bal = \left(\frac{CompanyReality - ImprovedObjective}{CompanyReality} \right) \times 100$$

As shown in Table 7.7, the average *Improvement_Bal* of the initial solution, which is obtained by ASBsm via reducing the available time, is 13% and the maximum *Improvement_Bal* is 29%, while the minimum is 7%. However, it is clear that when the available time is reduced to 80% and 60%, ASBsm results are still better than company reality, with an average *Gain_Bal* of 15%. Moreover, as de-

Table 7.7: Computational Results of the Balancing Problems Using ASBsm Method for Real Instances considering the Company Reality.

Instances	Initial Solution				Improved Solution					Company Reality			
	N.er Operators	N.er Machines	Objective	Time(s)	N.er Operators	N.er Machines	Objective	Time(s)	Improvement_Bal	N.er Operators	N.er Machines	Objective	Gain_Bal
Medium-Size-1	16	16	17.521	0.18	11	11	12.506	0.84	29%	12	12	13.682	9%
Medium-Size-2	11	11	12.470	0.30	10	10	11.421	8.05	8%	11	11	12.815	11%
Medium-Size-3	13	13	14.421	0.33	12	12	13.362	5.71	7%	12	12	12.455	1%
Medium-Size-4	16	18	19.838	0.63	11	13	14.783	3.16	25%	14	16	17.751	17%
Medium-Size-5	12	13	13.672	0.53	11	12	12.442	2.14	9%	14	15	16.205	23%
Large-Size-1	79	81	86.577	12.34	70	72	77.535	265.54	10%	70	81	91.771	16%
Large-Size-2	79	83	88.675	20.98	70	73	78.587	382.499	11%	70	83	96.432	19%
Large-Size-3	81	81	86.235	16.80	70	70	75.146	299.99	13%	69	79	89.184	16%
Large-Size-4	85	91	95.392	21.75	72	75	79.285	382.499	17%	71	85	95.741	17%
Large-Size-5	80	81	86.559	23.80	70	70	75.146	299.99	13%	70	81	91.418	18%
Large-Size-6	86	101	103.078	29.14	83	90	91.936	535.17	11%	83	95	104.196	12%
Large-Size-7	82	85	89.254	9.42	74	76	80.194	311.21	10%	80	98	104.309	23%
Large-Size-8	85	94	96.865	16.34	80	85	87.73	338.24	9%	84	94	101.524	14%
Large-Size-9	86	89	92.232	17.2	79	81	84.127	101.83	9%	83	97	285.66	18%

picted in Table 7.8, the average *Improvement_Bal* of the initial solution by the RPW-VNDbal method, that is obtained by reducing the available time, is 16% and the maximum *Improvement_Bal* is 26%, while the minimum is 12%. However, the average *Gain_Bal* by RPW-VNDbal is 16% which means that it is 16% better than reality by using only some percentage of the available time.

The summary of the results of both methods, considering reality of the company, is shown in Table 7.9. According to this Table and Table 7.6, it is clear that RPW-VNDbal method attains better results compared to company reality and at the same time, the method is better than ASBsm method, considering the quality of the solutions and the execution times of the algorithm. Moreover, simulation is used to validate the results of both methods. Two graphs (Figure 7.1) summarise the comparisons illustrated for the medium- and large-size instances of this section.

Table 7.8: Computational Results of the Balancing Problems Using RPW-VNDbal Method for Real Instances considering the Company Reality.

Instances	Initial Solution				Improved Solution					Company Reality			
	N.er Operators	N.er Machines	Objective	Time(s)	N.er Operators	N.er Machines	Objective	Time(s)	Improvement_Bal	N.er Operators	N.er Machines	Objective	Gain_Bal
Medium-Size-1	14	14	15.523	0.18	10	10	11.511	0.81	26%	12	12	13.682	16%
Medium-Size-2	12	12	13.472	0.18	9	9	10.477	0.46	22%	11	11	12.815	18%
Medium-Size-3	14	14	15.443	0.23	11	11	12.447	0.55	19%	12	12	12.455	7%
Medium-Size-4	15	17	18.825	0.39	13	14	15.782	1.04	16%	14	16	17.751	11%
Medium-Size-5	11	12	13.170	0.33	10	11	11.953	0.69	9%	14	15	16.205	26%
Large-Size-1	77	82	87.570	7.90	69	71	76.553	78.39	13%	70	81	91.771	17%
Large-Size-2	81	85	90.669	8.10	70	73	78.585	93.70	13%	70	83	96.432	19%
Large-Size-3	80	85	90.246	7.87	67	70	75.149	84.91	17%	69	79	89.184	16%
Large-Size-4	85	94	98.410	22.26	73	77	81.304	90.51	17%	71	85	95.741	15%
Large-Size-5	80	82	87.560	11.07	70	71	76.532	100.74	13%	70	81	91.418	16%
Large-Size-6	86	100	102.221	24.75	82	88	90.062	119.96	12%	83	95	104.196	14%
Large-Size-7	84	92	96.285	9.16	74	80	84.217	114.18	13%	80	98	104.309	19%
Large-Size-8	85	96	98.865	7.40	78	84	86.763	106.74	12%	84	94	101.524	15%
Large-Size-9	85	93	96.253	6.76	78	81	84.148	101.83	13%	83	97	103.028	18%

Table 7.9: A Comparison Between ASBsm and RPW-VNDbal Methods for the Balancing Problems considering the Company Reality.

Method	Improvement_Bal			Gain_Bal		
	Min	Max	Average	Min	Max	Average
ASBsm	7%	29%	13%	1%	23%	15%
RPW-VNDbal	12%	26%	16%	7%	22%	16%

7.2 Sequencing Results

This section addresses the sequencing issue. Firstly, the results of the optimisation model for the small instances are reported. Then, a comparison is made between its results and those of the two developed methods ($VND - MSeq$ and $GA - MSeq$) for small instances. Secondly, the two approximate sequencing methods are applied to the real instances and each instance is run five times for the best solution in

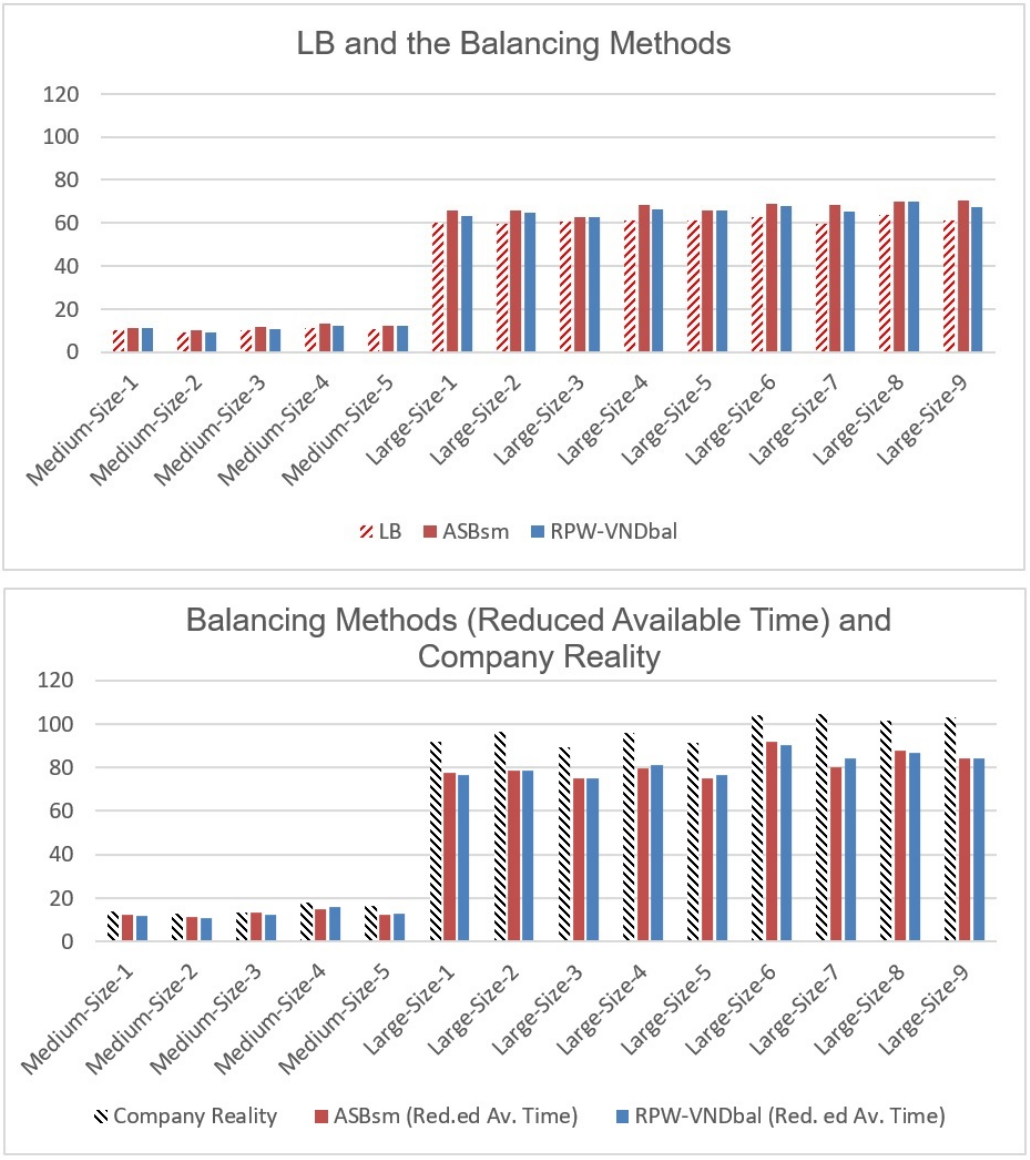

Figure 7.1: Comparison of the Results of the Balancing Problems for the Real Data.

both methods. Then, concerning those outcomes, another comparison is made with company's realities and LB. Moreover, as mentioned before, it is supposed that the balancing problem is already solved and the sequencing in this section is the main issue. Therefore, the input data for the sequencing are the output data of the balancing. Meanwhile, all the instances used in this category are similar to those in the previous section.

Table 7.10 features the results of the computational tests on the generated instances performed by the optimisation model and VND-MSeq method. As previously mentioned, these instances are the same as those in the previous section, but only the first 6 instances are in an acceptable time since the last two instances could not be solved by the given optimisation model. The *Gap_SeqOpt* is also calculated according to Section 7.1 equal to the *Gap_BalOpt* formula.

Table 7.10: Computational Results of the Sequencing Problems for the Small-size Instances, Using the Optimisation Model and VND-MSeq.

Instances	N.er Models	N.er Boxes	N.er Tasks	N.er Used Operators	N.er Used Machines	Initial Solution		Improved Solution		Optimal Solution		
						Objective (Cmax)	Time(s)	Objective (Cmax)	Time(s)	Objective (Cmax)	Time(s)	Gap_SeqOpt
S. Size-1	2	3	3	2	2	200	0.03	200	1.96	200	0.16	0.0%
S. Size-2	3	5	4	3	3	270	0.06	270	6.76	270	0.19	0.0%
S. Size-3	3	8	4	4	4	458	0.06	412	4.54	412	0.25	0.0%
S. Size-4	4	6	5	3	3	414	0.13	342	9.60	332	0.23	3.0%
S. Size-5	4	13	5	4	4	688	0.45	630	62.44	614	19512	2.6%
S. Size-6	6	34	6	4	4	2400	1.99	2400	104.04	2400	81638	0.0%

Considering Table 7.10, the optimal solution is found by VND-MSeq in 4 instances and the other two instances have a *Gap_SeqOpt* equal to 3.0% and 2.6%. As it is clear, instances number 5 and 6 are solved optimally in hours but not seconds, and the given method (VND-MSeq) solved them quickly; one of them has a *Gap_SeqOpt* equal to 0. Considering the next method, which is GA-MSeq, Table 7.11 depicts the results related to its application in the given small instances. Like the previous method, GA-MSeq gives the optimal solution in 4 instances, and the other two instances have a *Gap_SeqOpt* equal to 3.0% and 1.6%, which are a

bit less than the *Gap_SeqOpt* in the previous approach. To sum up, the average *Gap_SeqOpt* for the small instances solved by the VND-MSeq and GA-MSeq is 1%. Moreover, the initial solutions or populations in both methods are the ones with good qualities as expected since they are not generated randomly.

Table 7.11: Computational Results of the Sequencing Problems for the Small-size Instances, Using the Optimisation Model and GA-MSeq.

Instances	N.er Models	N.er Boxes	N.er Tasks	N.er Used Operators	N.er Used Machines	Best Initial Population		Improved Solution		Optimal Solution		
						Objective (Cmax)	Time(s)	Objective (Cmax)	Time(s)	Objective (Cmax)	Time(s)	Gap_SeqOpt
S. Size-1	2	3	3	2	2	200	0.19	200	3.60	200	0.16	0.0%
S. Size-2	3	5	4	3	3	270	0.51	270	18.14	270	0.19	0.0%
S. Size-3	3	8	4	4	4	428	0.69	412	22.20	412	0.25	0.0%
S. Size-4	4	6	5	3	3	352	0.65	342	23.33	332	0.23	3.0%
S. Size-5	4	13	5	4	4	658	3.22	624	133.85	614	19512	1.6%
S. Size-6	6	34	6	4	4	2400	16.76	2400	629.76	2400	81638	0.0%

The medium and large instances (the real data) which are used in the balancing section are solved in this section by two sequencing methods. The results of both methods were then compared with the reality and a defined LB. Table 7.3 which was provided before summarises general information about those instances. We use the results of the RPW-VNDbal method, while considering a percentage of the available time as mentioned before, to have the input data for the sequencing. Therefore, the final information of the used instances is outlined in Table 7.12.

According to those data, the VND-MSeq method is initially considered and the results are shown in Table 7.13. In this table, the company's realities are considered, which means that the makespan or Cmax of the company in this table is

Table 7.12: Sequencing Problem: Real Instance Data and Number of Operators and Machines Used.

Instances	N.er Models	N.er Boxes	N.er Tasks	N.er Ava.le Operators	N.er Ava.le Machines	N.er Machine Types	Max Operator Ab.ties	N.er Used Operators	N.er Used Machines	Used Per.ge of Av.le Time
Medium-Size-1	15	127	15	28	41	15	11	10	10	80%
Medium-Size-2	11	171	10	26	40	10	8	9	9	80%
Medium-Size-3	16	159	15	28	41	14	12	11	11	80%
Medium-Size-4	41	209	30	27	42	20	19	13	14	80%
Medium-Size-5	35	94	47	17	31	5	39	11	12	80%
Large-Size-1	303	334	70	86	146	15	36	69	71	60%
Large-Size-2	320	356	62	86	152	16	31	70	73	60%
Large-Size-3	298	335	68	86	148	14	36	67	70	60%
Large-Size-4	326	377	73	86	143	15	36	73	77	60%
Large-Size-5	344	399	69	86	148	14	36	70	71	60%
Large-Size-6	185	354	85	86	141	14	41	82	88	60%
Large-Size-7	196	401	70	86	141	14	33	74	80	60%
Large-Size-8	157	283	77	86	144	14	42	78	84	60%
Large-Size-9	148	291	93	86	143	14	45	78	81	60%

the one which occurs on each day in reality. Certainly, more operators and machines are used in reality. The *Improvement_Seq* in this table is related to the initial and final solutions and, the method of its calculation is the same as calculating *Improvement_Bal* in the balancing section. *Gain_Seq_*1 calculation is also the same as *Gain_Bal* calculation shown previously, which is the difference between the company reality (available time) and proposed method divided by the company reality (available time). However, there is another indicator for comparison which considers the available time, algorithms result and a LB and is called *Gain_Seq_*2. Below, the method of its calculation is explained.

There is a LB for any sequencing problem indicating that a production plan could not be finished sooner than this LB. According to Section 5.1 in Chapter 5, considering constraint number 1 in the given optimisation model, the earliest starting time of all the available tasks considering model and box could be calculated. Then, the finishing time of each task is calculated by having the earliest starting time of it and considering the processing time. Afterwards, the LB is defined as the maximum value of the finishing times among all. Moreover, the balancing re-

sults are controlled by simulation and during the available time defined for each instance, the production plan could be satisfied. Note that in the company's reality with more used operators and machines compared to the balancing results, the production plan is satisfied during the available time, which is a one-day production plan. Among all those issues, the sequencing methods improve the results of balancing and simulation and give better results than the available time. Therefore, as another indicator for evaluating the defined sequencing methods, the $Gain_Seq_2$ is defined. In $Gain_Seq_2$ calculation, three elements are considered, including LB, reality completion time (Av) and sequencing methods completion times. Besides, the defined LB in this paper is the smallest LB that could be defined for any sequencing problem. However, the $Gain_Seq_2$ formulation is as below:

$$Gain_Seq_2 = (Gap_Seq_AvtoLB - Gap_Seq_AlgtoLB) \times 100$$

Therefore, Gap_Seq_AvtoLB and $Gap_Seq_AlgtoLB$ should be defined in advance. As a result, the difference between the LB and reality completion time (Av) available time divided by LB is equal to Gap_Seq_AvtoLB. Moreover, the difference between the LB and each solution method completion time ($Cmax_{alg}$) divided by LB is equal to $Gap_Seq_AlgtoLB$. Meanwhile, both of these two values are calculated in percentage as below:

$$Gap_Seq_AvtoLB = \left(\frac{Av - LB}{LB}\right) \times 100$$

$$Gap_Seq_AlgtoLB = \left(\frac{Cmax_{alg} - LB}{LB}\right) \times 100$$

Generally, the $Gain_Seq_2$ is an indicator which is used only for comparison and clarify how much the developed methods are better than reality, but considering the defied LB.

In the following, GA-MSeq (another sequencing method) is used for the real instances. Its results are shown in Table 7.14. Every indicator in this table is the same as Table 7.13. However, the summary of both of used methods is provided in Table 7.15.

Table 7.13: Computational Results of the Sequencing Problems Using VND-MSeq Method for Real Instances.

Instance	N.er Used Operators	N.er Used Machines	Initial Sol.n		Improved Sol.n			Company Real.y		LB			
			Objective (Cmax)	Time(s)	Objective (Cmax)	Time(s)	Improvement_Seq	Objective (Cmax)	Gain_Seq_1	LB	Gap_Seq_AvtoLB	Gap_Seq_AlgtoLB	Gain_Seq_2
Medium-Size-1	10	10	26490	0.9	23871	66.3	10%	28800	17%	7670	275%	211%	64%
Medium-Size-2	9	9	27600	0.9	23400	268.8	15%	28800	19%	900	3100%	2500%	600%
Medium-Size-3	11	11	26100	1.0	23580	133.5	10%	28800	18%	4800	500%	391%	109%
Medium-Size-4	13	14	25800	0.8	25800	75.0	0%	32400	20%	1740	1762%	1383%	379%
Medium-Size-5	11	12	33462	1.2	27657	587.0	17%	32400	15%	23460	38%	18%	20%
Large-Size-1	69	71	34705	38.3	31829	1398.7	8%	33000	4%	13830	139%	130%	9%
Large-Size-2	70	73	36445	35.8	31804	3561.4	13%	33000	4%	14470	128%	120%	8%
Large-Size-3	67	70	35680	37.8	30216	3041.2	15%	33000	8%	11570	185%	161%	24%
Large-Size-4	73	77	34026	122.0	31370	3946.9	8%	33000	5%	13670	141%	129%	12%
Large-Size-5	70	71	32212	58.4	29556	3336.1	8%	33000	10%	10530	213%	181%	32%
Large-Size-6	82	88	39419	60.7	31659	3453.8	20%	33000	4%	11760	181%	169%	12%
Large-Size-7	74	80	31234	54.8	28302	3175.7	9%	33000	14%	13140	151%	115%	36%
Large-Size-8	78	84	34829	54.7	32226	3038.5	7%	33000	2%	13740	140%	134%	6%
Large-Size-9	78	81	35308	47.3	31287	3470.0	11%	33000	5%	13390	146%	134%	12%

Finally, according to the sequencing results, the following points can be obtained:

1. The initial solution of GA-MSeq is better than the VND-MSeq. In some instances, if it is not better, then it is equal but not worse.

2. The time of gaining the initial solution of VND-MSeq is better than GA-MSeq. It is related to fining population instead of a solution in GA-MSeq.

3. *Improvement_Seq* of the VND-MSeq is better than GA-MSeq.

4. Generally, the *Gain_Seq_*1 and *Gain_Seq_*2 in GA-MSeq are better than VND-MSeq meaning that GA-MSeq produce better results. However, the difference between the final solutions is not significant.

5. The running times of both methods for the small and medium instances are short. It is less than seconds for small instances and less than 7 minutes for

Table 7.14: Computational Results of the Sequencing Problems Using GA-MSeq Method for Real Instances.

Instance	N.er Used Operators	N.er Used Machines	Initial Sol.n		Improved Sol.n			Company Real.y		LB			
			Objective (Cmax)	Time(s)	Objective (Cmax)	Time(s)	Improvement_Seq	Objective (Cmax)	Gain_Seq_1	LB	Gap_Seq_AvtoLB	Gap_Seq_AlgtoLB	Gain_Seq_2
Medium-Size-1	10	10	25877	9.9	23871	262.5	8%	28800	17%	7670	275%	211%	64%
Medium-Size-2	9	9	24900	8.6	23100	307.8	7%	28800	20%	900	3100%	2467%	633%
Medium-Size-3	11	11	25080	10.4	23420	348.0	7%	28800	19%	4800	500%	388%	112%
Medium-Size-4	13	14	25800	10.5	25800	325.2	0%	32400	20%	1740	1762%	1383%	379%
Medium-Size-5	11	12	30654	9.5	27180	370.1	11%	32400	16%	23460	38%	16%	22%
Large-Size-1	69	71	34520	107.1	30966	2136.8	10%	33000	6%	13830	139%	124%	15%
Large-Size-2	70	73	34697	112.0	31828	1715.6	8%	33000	4%	14470	128%	120%	8%
Large-Size-3	67	70	33286	126.2	30129	3549.2	9%	33000	9%	11570	185%	160%	25%
Large-Size-4	73	77	32964	141.8	31099	3552.6	6%	33000	6%	13670	141%	127%	14%
Large-Size-5	70	71	31432	194.3	28457	3241.6	9%	33000	14%	10530	213%	170%	43%
Large-Size-6	82	88	35489	147.8	31844	3050.1	10%	33000	3%	11760	181%	172%	9%
Large-Size-7	74	80	31234	154.8	28290	3173.9	9%	33000	14%	13140	151%	115%	36%
Large-Size-8	78	84	34123	149.8	31199	3565.1	9%	33000	5%	13740	140%	127%	13%
Large-Size-9	78	81	34026	104.2	31370	3258.9	8%	33000	5%	13390	146%	134%	12%

Table 7.15: A Comparison Between VND-MSeq and GA-MSeq Methods for the Sequencing Problem.

Method	Improvement_Seq			Gain_Seq_1			Gain_Seq_2		
	Min	Max	Average	Min	Max	Average	Min	Max	Average
VND-MSeq	0%	20%	11%	2%	20%	10%	6%	600%	95%
GA-MSeq	0%	11%	8%	4%	20%	11%	8%	633%	99%

medium ones.

6. The running times for small and medium size instances in VND-MSeq are less than GA-MSeq.

7. The running times of large instances are around 1 hour which is acceptable regarding their large dimension.

8. The running times of GA-MSeq for large instances are less than VND-MSeq.

9. There is an instance that cannot be improved by both methods (instance of $Medium - Size - 4$). The reason is that the cmax on that instance is equal to the working time of one of the operators in the line.

Figure 7.2 summarises the comparisons illustrated for the medium- and large-size instances of this section.

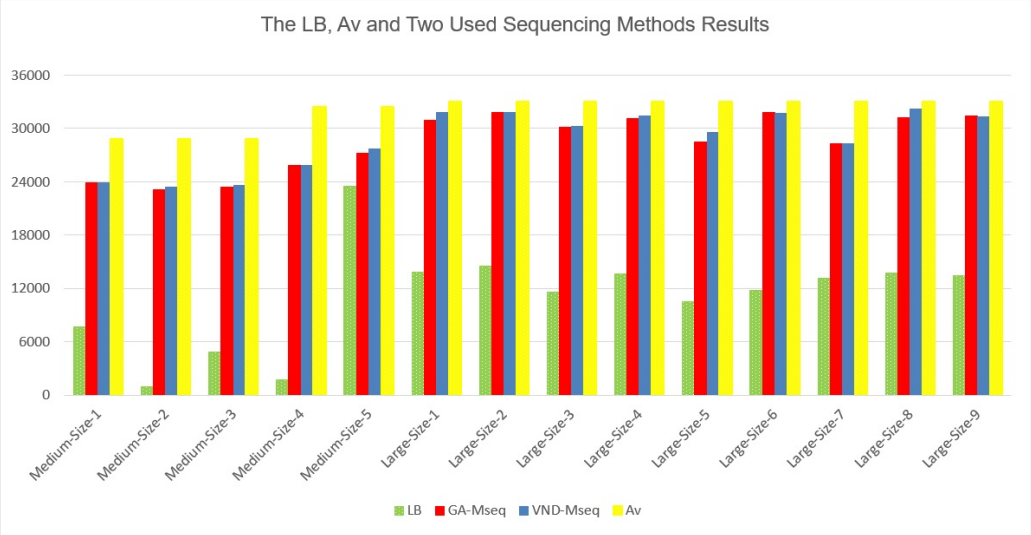

Figure 7.2: Comparing Different Results of the Sequencing Problem for the Real Data.

7.3 Balancing and Sequencing Results

This section contains the results based on simultaneously considering the balancing and sequencing problems. According to the related chapter about the given problems (Chapter 6), the optimisation model is first considered for the small instances and at the same time, the results of $NSGA - II$ and GA for the small instances are presented. Secondly, the results of NSGA-II and GA for the real large instances are provided. First, GA (as a single-objective solution method) results and then NSGA-II results are given. Note that due to considering the multi-objective solution method, i.e. NSGA-II, a set of multi-objective solutions, and not just one

solution, is obtained. Moreover, the best solution of the single-objective method is usually presented in that set.

Table 7.16 features the results of the optimisation models in chapters 4, 5 and 6 for the generated small instances. As previously indicated, these instances are the first 5 instances of the two previous sections. These 5 instances are such that the optimal solutions are found within an acceptable time. To allow comparison, in the given table, the best results of these problems are considered separately (MALBP and then MALSP) and simultaneously (MALBSP). In the following, the first *N.er Used Operators Bal*, the *N.er Used Machines Bal* and *Smoothing Bal* in Table 7.16 are based on the results of the optimisation model related to the balancing in Chapter 4 and *Cmax Seq* is based on the results of the optimisation model related to the sequencing in Chapter 5. The *Gain_BalandSeq_*1 calculation is mentioned in below, in which both objectives are considered (separately and simultaneously). According to this table and concerning the *Gain_BalandSeq_*1 (with the average of 0.7%), it is clear that the balancing and sequencing problems together give better results when compared with the solutions obtained separately.

$$Gain_BalandSeq_1 = \left(\frac{SeparateObjective - SimultaneousObjective}{SeparateObjective} \right) \times 100$$

Table 7.17 includes the results of the computational tests for the 5 generated instances performed by the GA method. The results are then compared with the optimal solution objectives. The *Gap_BalandSeqOpt* calculation in this table is the same as mentioned before, in Section 7.1 equal to the *Gap_BalOpt* formula.

According to this table, in 3 instances, the GA attains the optimal solution and the other two instances have a *Gap_BalandSeqOpt* equal to 2.5% and 2.9%. The average *Gap_BalandSeqOpt* is 1.1%. The number of used operators and machines in this table is optimal and the values of Camx are generally better than those of the sequencing methods. Next, the results of the multi-objective problems are drawn in a graph presented in Figure 7.3. In this figure, only the first front of each problem is depicted since, as mentioned before, they are better solutions than the other solutions in other fronts. Each curve contains solutions of the first front. Moreover,

Table 7.16: Computational Results of the Optimisation Models for the Balancing and Sequencing Problems Separately and Simultaneously.

Instances	Separately					Simultaneously						
	N.er Used Operators Bal	N.er Used Machines Bal	Smoothing Bal	Cmax Seq	Objective	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	Time(s)	Gain_BalandSeq_1
S. Size-1	2	2	100	200	5.0	2	2	100	200	5.0	0.1	0.0%
S. Size-2	3	3	132	270	7.02	3	3	132	270	7.02	0.2	0.0%
S. Size-3	4	4	96	412	9.08	4	4	96	384	8.80	0.2	3.1%
S. Size-4	3	3	199	332	8.31	3	3	199	332	8.31	0.2	0.0%
S. Size-5	4	4	216	614	12.30	4	4	216	610	12.26	37500	0.3%

any multi-objective problem gives us two extreme solutions. In this research, they are solutions that, for example, use more operators and machines and reduce Cmax and, on the contrary, employ fewer operators and machines and increase Cmax; the other solutions are in between. Therefore, the solutions which could be compared with GA (single-objective) solutions are bolded in the given figure (7.3). Those bolded solutions in some instances are the optimal ones. The solution values and the running times of the NSGA-II are shown in Table 7.18.

Given NSGA-II results for the small instances (Table 7.18) 3 instances reach the optimal solutions and the other two instances have a *Gap_BalandSeqOpt* equal to 1.4% and 3.3%. The average *Gap_BalandSeqOpt* is 0.9%. According to this table, the results are better than those of GA, but the running times are longer, as it is expected. Moreover, for better comparison, Figure 7.4 depicts all the given results concerning the small instances, the optimisation models and the given solution methods of this section.

Table 7.17: Computational Results of Simultaneously Balancing and Sequencing Problems for the Small-size Instances, Using the Optimisation Model and GA.

Instances	Initial Solution						Improved Solution						Optimal Solution						
	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	Time(s)	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	Time(s)	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	Time(s)	Gap_BalandSeqOpt
S. Size-1	3	3	75	150	5.25	0.3	2	2	100	200	5.0	4.2	2	2	100	200	5.0	0.1	0.0%
S. Size-2	4	4	119	310	8.29	0.6	3	3	132	270	7.02	12.4	3	3	132	270	7.02	0.2	0.0%
S. Size-3	4	5	96	392	9.88	0.7	4	4	96	406	9.02	14.7	4	4	96	384	8.80	0.2	2.5%
S. Size-4	4	5	199	282	9.83	0.9	3	3	199	332	8.31	21.7	3	3	199	332	8.31	0.2	0.0%
S. Size-5	5	6	247	724	18.42	9.3	4	4	216	646	12.62	81.5	4	4	216	610	12.26	37500	2.9%

In the following, the medium and large real instances are considered. Generally, by giving weights to the different parts of the objective function, one could reach a solution that is usually available in the set of solutions of the multi-objective method (NSGA-II). Therefore, the results of those problems are given by the multi-objective method. Table 7.19 indicates the results of NSGA-II, as compared with the best previously developed methods, after each instance is run five times, for the best solution. Additionally, the best results in the given table for the MALBP are based on the RPW-VNDbal method and the best solutions of the MALSP are base on the GA-MSeq. The objective function considered is according to the objective function of the optimisation model in Chapter 6.

The *Gain_BalandSeq_*1 calculation is mentioned before. However, the *Gain_Ba -landSeq_*1 considered the separated methods objectives (MALBP and then MALSP) and the NSGA-II results. As shown in that table (7.19), the average *Gain_BalandSeq_*1 is 4% and the minimum and maximum values are 1% and 11%. The values of

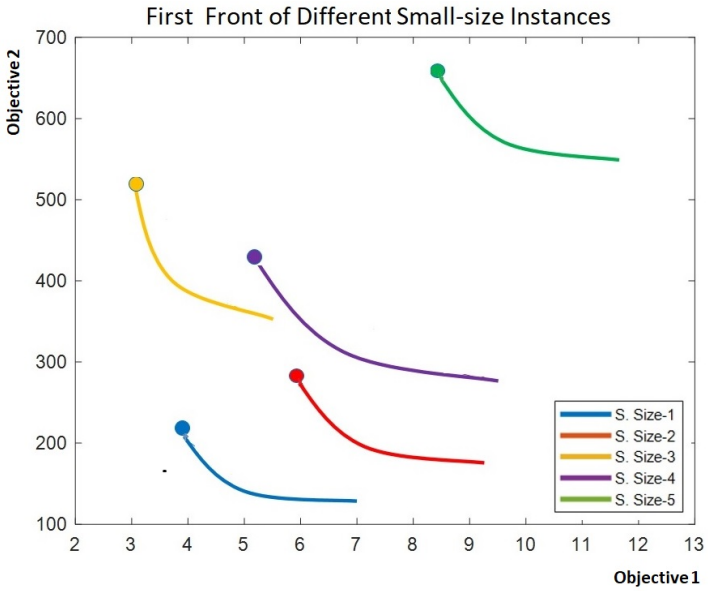

Figure 7.3: First Front Results for the Small-size Instances, Using NSGA-II, in Simultaneously Balancing and Sequencing.

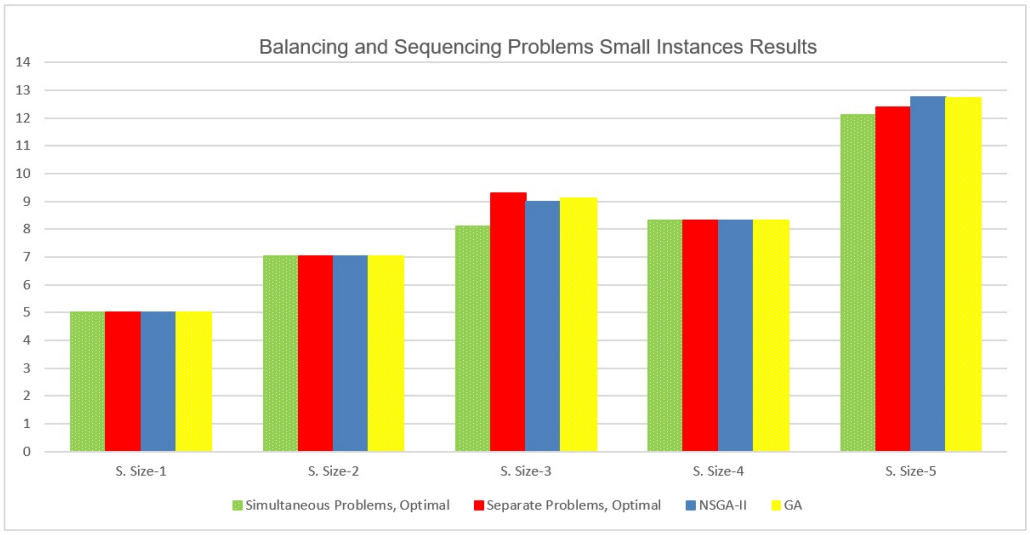

Figure 7.4: Comparing Different Results of the Balancing and Sequencing Problems for the Small Instances.

Gain_BalandSeq_1 indicate that the NSGA-II method is better than other separate methods. Since, the results are better than those of other approaches, it is

Table 7.18: Computational Results of Simultaneously Balancing and Sequencing Problems for the Small-size Instances, Using the Optimisation Model and NSGA-II.

Instances	Initial Solution						Improved Solution						Optimal Solution						
	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	Time(s)	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	Time(s)	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	Time(s)	Gap_BalandSeqOpt
S. Size-1	3	3	75	150	5.25	0.7	2	2	100	200	5.0	19.6	2	2	100	200	5.0	0.1	0.0%
S. Size-2	4	4	119	310	8.29	1.4	3	3	132	270	7.02	37.7	3	3	132	270	7.02	0.2	0.0%
S. Size-3	4	5	96	392	9.88	1.7	4	4	96	396	8.92	45.4	4	4	96	384	8.80	0.2	1.4%
S. Size-4	4	5	199	282	9.83	0.9	3	3	199	332	8.31	26.5	3	3	199	332	8.31	0.2	0.0%
S. Size-5	5	6	247	724	18.42	20.4	4	4	216	650	12.66	267.2	4	4	216	610	12.26	37500	3.3%

expected that NSGA-II results should be better than the ones obtained by the company and be closer to the defined LB. Table 7.20 displays the results of NSGA-II and the company reality.

The *Gain_BalandSeq_2* indicates the improvement relative to the company reality by using NSGA-II, the formulation is mentioned in continue. Considering at Table 7.20, the average *Gain_BalandSeq_2* value is 19%, while the minimum and maximum values are 14% and 26%. The summary of the results of NSGA-II methods, considering other separate methods for balancing and sequencing (MALBP and then MALSP) and reality of the company, is shown in Table 7.21. Moreover, Figure 7.5 summarises the comparisons made for the real instances of this section.

$$Gain_BalandSeq_2 = \left(\frac{CompanyReality - ImprovedObjective}{CompanyReality}\right) \times 100$$

Table 7.19: Computational Results of Simultaneously Balancing and Sequencing Problems Using NSGA-II and the Best Previously Developed Methods.

Instances	Previous Solution					Improved Solution						
	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	Time(s)	Gain_BalandSeq_1
Medium-Size-1	10	10	1511	23871	35.382	12	12	1132	21653	34.785	230	2%
Medium-Size-2	9	9	1477	23100	33.577	12	12	1033	20200	33.233	348	1%
Medium-Size-3	11	11	1447	23420	35.867	4	14	938	20450	35.388	277	1%
Medium-Size-4	13	14	1782	25800	41.582	11	12	209 0	23020	30.11	450	11%
Medium-Size-5	11	12	629	27180	39.809	11	13	983	24660	38.643	393	3%
Large-Size-1	69	71	5553	30966	107.519	76	76	1586	19820	97.406	3366	9%
Large-Size-2	70	73	5585	31804	110.389	76	76	1520	24649	102.169	3645	7%
Large-Size-3	67	70	5149	30129	105.278	69	71	3089	26687	100.776	3180	4%
Large-Size-4	73	77	4304	31099	112.403	77	81	1200	26645	108.845	5325	3%
Large-Size-5	70	71	5532	28457	104.989	68	68	5532	29685	103.217	4920	2%
Large-Size-6	82	88	2063	31659	121.722	82	91	766	25975	117.741	3250	3%
Large-Size-7	74	80	4217	28290	112.507	82	83	1030	24838	108.868	4020	3%
Large-Size-8	78	84	2763	31199	117.962	80	83	1364	29840	114.204	84	5%
Large-Size-9	78	81	3148	31287	115.435	80	80	2015	27216	109.231	83	5%

7.4 Conclusions

This chapter benefits from the consideration of various solution approaches developed in this thesis on varied instances. To allow a good evaluation and comparison, all the instances of different sizes are the same throughout the chapter. The small-size instances are generated but the medium- and large-size instances correspond to real data of the company. Small sizes are specially used to compare the approximate solution methods with the optimal ones.

Considering the separate and simultaneous balancing and sequencing mixed model problems, and the optimisation models which are given, the simultaneous (MALB&SP) gives better results than the separate one (MALBP and then MALSP). The following points summarise the conclusions of the chapter:

Table 7.20: Computational Results of Simultaneously Balancing and Sequencing Problems Using NSGA-II Method for Real Instances considering the Company Reality.

Instances	Improved Solution					Company Reality					
	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	N.er Used Operators	N.er Used Machines	Smoothing	Cmax	Objective	Gain_BalandSeq_2
Medium-Size-1	12	12	1132	21653	34.785	12	12	1682	28800	42.482	18%
Medium-Size-2	12	12	1033	20200	33.233	11	11	1815	28800	41.615	20%
Medium-Size-3	14	14	938	20450	35.388	12	12	1455	28800	42.255	16%
Medium-Size-4	11	12	2090	23020	30.11	14	16	1751	32400	50.151	26%
Medium-Size-5	11	13	983	24660	38.643	14	15	1205	32400	48.605	20%
Large-Size-1	76	76	1586	19820	97.406	70	81	10771	33000	124.771	22%
Large-Size-2	76	76	1520	24649	102.169	70	83	13432	33000	129.432	21%
Large-Size-3	69	71	3089	26687	100.776	69	79	10184	33000	122.184	18%
Large-Size-4	77	81	1200	26645	108.845	71	85	10741	33000	128.741	15%
Large-Size-5	68	68	5532	29685	103.217	70	81	10418	33300	124.418	17%
Large-Size-6	82	91	766	25975	117.741	83	95	9196	33000	137.196	14%
Large-Size-7	82	83	1030	24838	108.868	80	98	6309	33000	137.309	21%
Large-Size-8	80	83	1364	29840	114.204	84	94	7524	33000	134.524	15%
Large-Size-9	80	80	2015	27216	109.231	83	97	6028	30000	136.028	20%

Table 7.21: Summary Results of NSGA-II.

Method	Gain_BalandSeq_1			Gain_BalandSeq_2		
	Min	Max	Average	Min	Max	Average
NSGA-II	1%	11%	4%	14%	26%	19%

- Regarding the balancing methods, the RPW-VNDbal outperforms the ASBsm and the running times of both methods are short for all instances.

- Concerning the sequencing approaches, GA-MSeq results are better than VND-MSeq, but the difference between them is not significant. Also, the running times of VND-MSeq for small and medium instances are less than those of GA-MSeq. In contrast, in large instances, GA-MSeq execution times are shorter than those of VND-MSeq.

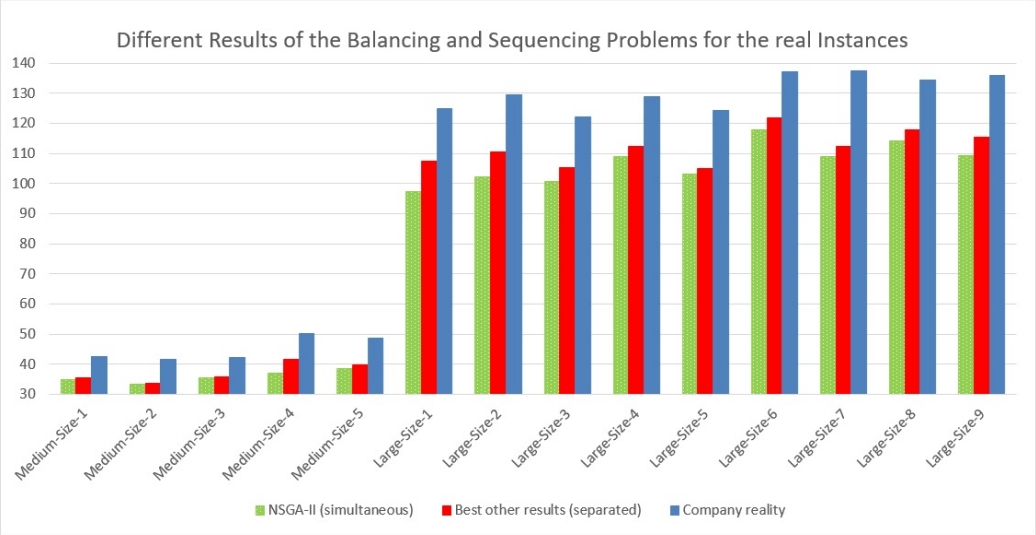

Figure 7.5: Comparing Different Results of the Balancing and Sequencing Problems for the real Instances.

- As to balancing and sequencing simultaneously, the NSGA-II results are superior, if compared with GA, but GA is a bit faster than NSGA-II. Besides, NSGA-II is a multi-objective method and provides different types of solution for the decision maker.

- Finally, simultaneous approaches outperform the separate ones, but execution times are superior to those of the separate approaches.

Chapter 8

Conclusions

This thesis has considered Balancing and Sequencing Mixed-Model Assembly Systems. The industrial context is a large footwear company with new flexible stitching systems and the practical problems solved in this work are, therefore, unique.

In this way, the work contributes both to the scientific area of mixed-model assembly line balancing and sequencing and the practical scope of helping to overcome the daily difficulties of the company with new technologies to be managed. In fact, the company has handled the problem of balancing and sequencing fairly manually, so the methods here devised can clearly improve its procedures. The case study involved two different stitching systems integrated in the new dedicated automatic transportation equipment, which will only be effective if well managed. Given that production plans change rapidly, it is crucial to know and adapt the number of the operators and machines required and understand the way of allocation the tasks to them. Such problems simultaneously involve operators and machines, and the units moving in the lines are not pairs of shoes but boxes containing numerous quantities of product components.

Therefore, the research conducted in this thesis was divided into two main categories: separately ($MALBP$ and then $MALSP$) and simultaneously balancing and sequencing ($MALB\&SP$). Other essential issues are related to the objective functions, which try to interpret the aims and goals of company managers, namely: minimising the number of necessary operators and machines, smoothing the oper-

ators' workload and minimising the maximum completion times. Such objectives were contemplated in single and multiple-objective problems. Consequently, the solutions approaches were developed accordingly.

The footwear industry, the given problems and the literature related to them were presented in the first three chapters. Chapters 4 and 5 are dedicated to the first category (MALBP and then MALSP) and Chapter 6 has considered the second category (MALB&SP). Each of these chapters, firstly focused on presenting an optimisation model associated with the studied problem and an argument concerning the optimal solution. Secondly, the approximate methods were described. However, by considering a variety of models, boxes, tasks and WIP in reality the given problems are very complex, it was observed that exact methods failed to provide good solutions within an acceptable time. Consequently, approximate methods were designed to realistically cope with the case studies. Simulation was also used in this study to evaluate the results. Chapter 7 contains the computational results and some comments about the developed methods. The used rules and developed methods in this thesis were:

- Heuristic inspired by Tabu Search (TS);
- Ranked Positional Weighted (RPW) method;
- Variable Neighbourhood Descent (VND) metaheuristic;
- Dispatching rules;
- Genetic Algorithm (GA) metaheuristic;
- Non-Dominated Sorting Genetic Algorithm II ($NSGA-II$) metaheuristic.

This PhD thesis, started with the balancing problems and developed two different approaches. The first one was based on a heuristic inspired by TS and the second one was a VND metaheuristic that used the adapted RPW method to create the initial solution. The objective function was minimising the number of workstation and smoothing operator's workloads. The results of both methods, in small

instances, had optimality gaps less than 1% and an average gap of 0.03%, which is a reasonable optimality gap. Considering the real instances the results of both methods are much better than the results from the company and also have an acceptable gap to the calculated LB. However, the method which is based on the VND (RPW-VNDbal) clearly outperforms the other one (ASBsm). They are able to quickly improve the initial solution by searching different neighbourhood structures. Since the sequencing is not considered in the balancing, simulation was used to evaluate the results.

Another fundamental theme of the thesis, i.e. sequencing the Assembly Systems, employed as input the output data of the balancing problem solutions and selected as objective the minimisation of the maximum completion times. Two methods, based on VND (VND-MSeq) and GA (GA-MSeq) were designed and implemented as solution procedures. The optimality gaps of both methods for the small instances are less than 3% with the average of 1%. Considering the real instances, they improved the known results from the company, with significant gains. In what concerns the objective function, GA-MSeq slightly outperforms VND-MSeq. The running times of VND-MSeq, for the small and medium instances, were shorter than those of GA-MSeq, but the opposite was true for the large instances (GA-MSeq was better than VND-MSeq). These methods were not as fast as the balancing solution methods, for the large instances.

Finally, balancing and sequencing were considered simultaneously. Single- and multi-objective approaches were followed, and for each one, a solution method was created. Both methods were based on GA. The multi-objective method, named NSGA-II, is based on the non-dominated sorting rule. Moreover, different dispatching rules were employed to obtain the initial populations. The optimal gaps of both methods are less than 3.4% and the average is around 1.1%, quite acceptable for such a complex problem.

To sum up, the integrated approach provided better results than the separate one but, if computational times are in question, then the separate one is recommended. On the other hand, the multi-objective approach can have the advantage

of providing more solutions to the decision-maker.

Operational efficiency and consumer satisfaction are two crucial elements for a business success in this competitive world. Certainly, this thesis contributes to these aspects and, in particular, narrows the gap between research and practice and brings new insights to the industry. Additionally, it includes novelties which are significant for mixed-model balancing and sequencing. The approaches and solution methods developed can help to optimise line balancing and to expedite sequencing and increase resource utilisation. This may be another support to the extraordinary international panorama and high competitiveness of the Portuguese footwear industry.

The approaches and outcomes of the work can be easily adapted and used in other footwear companies or similar industries.

Bibliography

A.Dolgui and I.Ihnatsenka. Branch and bound algorithm for optimal design of transfer lines with multi-spindle stations. *Ecole National Suoerieure des Mines, Sain*, 2004.

S Agrawal and MK Tiwari. A collaborative ant colony algorithm to stochastic mixed-model u-shaped disassembly line balancing and sequencing problem. *International Journal of Production Research*, 46(6):1405 – 1429, 2008.

Sener Akpınar, G. Mirac Bayhan, and Adil Baykasoglu. Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. *Applied Soft Computing*, 13 (1):574 – 589, jan 2013.

Enrique Alba and Rafael Martí. *Metaheuristic procedures for training neural networks*, volume 35. Springer Science & Business Media, 2006.

Corominas Albert, Pastor Rafael, and Plans Joan. Balancing assembly line with skilled and unskilled workers. *Omega*, 36(6):1126 – 1132, dec 2008.

F Tevhide Altekin, Levent Kandiller, and Nur Evin Ozdemirel. Profit-oriented disassembly-line balancing. *International Journal of Production Research*, 46 (10):2675 – 2693, 2008.

M Amiri, M Zandieh, M Yazdani, and A Bagheri. A variable neighbourhood search algorithm for the flexible job-shop scheduling problem. *International Journal of Production Research*, 48(19):5671 – 5689, 2010.

Edward J Anderson and Michael C Ferris. Genetic algorithms for combinatorial optimization: the assemble line balancing problem. *ORSA Journal on Computing*, 6(2):161 – 173, 1994.

Carlos Andres, Cristobal Miralles, and Rafael Pastor. Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, 187(3):1212 – 1223, 2008.

Leila Asadzadeh. A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy. *Computers & Industrial Engineering*, 102:359 – 367, 2016.

Emel Kızılkaya Aydoğan, Yılmaz Delice, Uğur Özcan, Cevriye Gencer, and Özkan Bali. Balancing stochastic u-lines using particle swarm optimization. *Journal of Intelligent Manufacturing*, pages 1 – 15, 2016.

Ying Bai, Hongshun Zhao, and Liao Zhu. Mixed-model assembly line balancing using the hybrid genetic algorithm. In *2009 International Conference on Measuring Technology and Mechatronics Automation*. IEEE, 2009.

Kenneth R Baker. *Introduction to sequencing and scheduling.* John Wiley & Sons, 1974.

JF Bard, A Shtub, and SB Joshi. Sequencing mixed-model assembly lines to level parts usage and minimize line length. *The International Journal of Production Research*, 32(10):2431 – 2454, 1994.

Jonathan F. Bard. Assembly line balancing with parallel workstations and dead time. *International Journal of Production Research*, 27(6):1005 – 1018, jun 1989.

Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316 – 329, 1998.

Olga Battaï and Alexandre Dolgu. A taxonomy of line balancing problems and their

solutionapproaches. *International Journal of Production Economics*, 142(2):259 – 277, apr 2013.

Olga Battaïa and Alexandre Dolgui. Reduction approaches for a generalized line balancing problem. *Computers and Operations Research*, 39(10):2337 –2345, 2012.

Daria Battin, Maurizio Faccio, Alessandro Persona, and Fabio Sgarbossa. Balancing- sequencing procedure for a mixed model assembly system in case of finite buffer capacity. *The International Journal of Advanced Manufacturing Technology*, 44(3 - 4):345 – 359, 2009.

Joaquín Bautista and Jordi Pereira. Ant algorithms for assembly line balancing. *Ant algorithms*, pages 49 – 61, 2002.

Joaquín Bautista, Rocío Alfaro, and Cristina Batalla. Modeling and solving the mixed-model sequencing problem to improve productivity. *International Journal of Production Economics*, 161:83 – 95, 2015.

Joaquín Bautista, Rocío Alfaro-Pozo, and Cristina Batalla-García. Grasp for sequencing mixed models in an assembly line with work overload, useless time and production regularity. *Progress in Artificial Intelligence*, 5(1):27 – 33, 2016.

Ilker Baybars. A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, 32(8):909 – 932, 1986.

Christian Becker and Armin Scholl. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168(3): 694 – 715, feb 2006.

Richard Bellman. *Dynamic programming*. Courier Corporation, 2013.

Mohand Lounes Bentaha, Olga Battaïa, and Alexandre Dolgui. Disassembly line balancing and sequencing under uncertainty. *Procedia CIRP*, 15:239 – 244, 2014.

Christian Bierwirth and Dirk C Mattfeld. Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation*, 7(1):1 – 17, 1999.

Dirk Biskup. Single-machine scheduling with learning considerations. *European Journal of Operational Research*, 115(1):173 – 178, 1999.

Christian Blum. *Theoretical and practical aspects of ant colony optimization*, volume 282. IOS press, 2004.

Christian Blum and Cristobal Miralles. On solving the assembly line worker assignment and balancing problem via beam search. *Computers & Operations Research*, 38(1):328 – 339, 2011.

Christian Blum and Andrea Rol. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3): 268 – 308, 2003.

Christian Blum and Michael Sampels. An ant colony optimization algorithm for shop scheduling problems. *Journal of Mathematical Modelling and Algorithms*, 3 (3):285 – 308, 2004.

Ahmet Bolat, Mehmet Savsar, and Mohammed A Al-Fawzan. Algorithms for real-time scheduling of jobs on mixed model assembly line. *Computers & Operations Research*, 21(5):487 – 498, 1994.

Leonardo Borba and Marcus Ritt. A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem. *Computers & Operations Research*, 45:87 – 96, 2014.

Nils Boysen, Malte Fliedner, and Armin Scholl. A classification of assembly line balancing problems. *European Journal of Operational Research*, 183(2):674 – 693, dec 2007a.

Nils Boysen, Malte Fliedner, and Armin Scholl. A classification of assembly line balancing problems. *European Journal of Operational Research*, 183(2):674 – 693, 2007b.

Nils Boysen, Malte Fliedner, and Armin Scholl. Assembly line balancing: Which

model to use when? *International Journal of Production Economics*, 111(2):509 – 528, 2008.

Nils Boysen, Malte Fliedner, and Armin Scholl. Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research*, 192(2):349 – 373, 2009.

Stephen Bradley, Arnoldo Hax, and Thomas Magnanti. Applied mathematical programming. 1977.

Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157 – 183, 1993.

Evelyn C Brown and Robert T Sumichrast. Evaluating performance advantages of grouping genetic algorithms. *Engineering Applications of Artificial Intelligence*, 18(1):1 – 12, 2005.

Michael J Brusco and Stephanie Stahl. *Branch-and-bound applications in combinatorial data analysis*. Springer Science & Business Media, 2006.

Joseph Bukchin and Jacob Rubinovitz. A weighted approach for assembly line design with station paralleling and equipment selection. *IIE Transactions*, 35(1): 73 – 85, 2003.

Kadir Buyukozkan, Ibrahim Kucukkoc, Sule Itir Satoglu, and David Z Zhang. Lexicographic bottleneck mixed-model assembly line balancing problem: artificial bee colony and tabu search approaches with optimised parameters. *Expert Systems with Applications*, 50:151 – 166, 2016.

Burcin Cakir, Fulya Altiparmak, and Berna Dengiz. Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm. *Computers & Industrial Engineering*, 60(3):376 – 384, 2011.

Zhenxin Cao and Shiping Ma. Balancing and sequencing optimization of the mixed model assembly lines. In *Computer Science and Computational Technology, 2008. ISCSCT'08. International Symposium on*, volume 1, pages 732 – 736. IEEE, 2008.

Robert L Carraway. A dynamic programming approach to stochastic assembly line balancing. *Management Science*, 35(4):459 – 471, 1989.

Sankhadeep Chatterjee, Soumen Banerjee, Kaustav Guha Mazumdar, Soumyadip Bose, and Soumya Sen. Non-dominated sorting genetic algorithmâĂŤii supported neural network in classifying forest types. In *Electronics, Materials Engineering and Nano-Technology (IEMENTech), 2017 1st International Conference on*, pages 1 – 6. IEEE, 2017.

Imran Ali Chaudhry and Abid Ali Khan. A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3):551 – 591, 2016.

James C. Chen, Chun-Chieh Chen, Ling-Huey Su, Han-Bin Wu, and Cheng-Ju Sun. Assembly line balancing in garment industry. *Expert Systems with Applications*, 39(11):10073 – 10081, sep 2012a.

James C Chen, Sudiana Wirasambada, I Nyoman Pujawan, KJ Wang, CW Wu, CJ Lin, and S Chiu. Minimizing total tardiness and makespan for hybrid flow shop problem with resource constraint in footwear manufacturing. In *Asia Pacific Industrial Engineering & Management Systems Conference*, pages 1093 – 1099, 2012b.

James C Chen, Aryka Pradhana Putra, Nikita Anggono, Jeff Chen, and Yung-Sheng Su. Simulation modeling and analysis for stitching line of footwear industry. In *International Conference on Industrial Engineering and Operations Management*, pages 1099 – 1106, 2014a.

James C Chen, Chien Wei Wu, Tran Dinh Duy Thao, Ling Huey Su, Wen Haiung Hsieh, and Tiffany Chen. Hybrid genetic algorithm for solving assembly line balancing problem in footwear industry. In *Advanced Materials Research*, volume 939, pages 623 – 629. Trans Tech Publ, 2014b.

JC Chen, KH Chen, JJ Wu, and CW Chen. A study of the flexible job shop

scheduling problem with parallel machines and reentrant process. *The International Journal of Advanced Manufacturing Technology*, 39(3-4):344 – 354, 2008.

M Chen and Y Dong. Applications of neural networks to solving smt scheduling problems a case study. *International Journal of Production Research*, 37(17):4007 – 4020, 1999.

Yin-Yann Chen, Chen-Yang Cheng, Li-Chih Wang, and Tzu-Li Chen. A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problemsâĂŤa case study for solar cell industry. *International Journal of Production Economics*, 141(1):66 – 78, 2013.

Tsung-Che Chiang and Li-Chen Fu. Multiobjective job shop scheduling using genetic algorithm with cyclic fitness assignment. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 3266 – 3273. IEEE, 2006.

Tsung-Che Chiang and Li-Chen Fu. Using dispatching rules for job shop scheduling with due date-based objectives. *International Journal of Production Research*, 45 (14):3245 – 3262, 2007.

Wen-Chyuan Chiang. The application of a tabu search metaheuristic to the assembly line balancing problem. *Annals of Operations Research*, 77:209 – 227, 1998.

Chin Soon Chong, Appa Iyer Sivakumar, Malcolm Yoke Hean Low, and Kheng Leng Gay. A bee colony optimization algorithm to job shop scheduling. In *Proceedings of the 38th conference on Winter simulation*, pages 1954 – 1961. Winter Simulation Conference, 2006.

Wai Mun Choo, Li-Pei Wong, and Ahamad Tajudin Khader. A modified bee colony optimization with local search approach for job shop scheduling problems relevant to bottleneck machines. *International Journal of Advanced Soft Computing Applications*, 8(2):52 – 78, 2016.

George Chryssolouris and Velusamy Subramaniam. Dynamic scheduling of manufacturing job shops using genetic algorithms. *Journal of Intelligent Manufacturing*, 12(3):281 – 293, 2001.

Parames Chutima and Palida Chimklai. Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. *Computers & Industrial Engineering*, 62(1):39 – 55, 2012.

Jeffery K Cochran, Shwu-Min Horng, and John W Fowler. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers & Operations Research*, 30(7):1087 – 1102, 2003.

Carlos A Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

Yann Collette and Patrick Siarry. *Multiobjective optimization: principles and case studies*. Springer Science & Business Media, 2013.

Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, and Marco Trubian. Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39 – 53, 1994.

Maria Teresa Costa and José Soeiro Ferreira. A simulation analysis of sequencing rules in a flexible flowline. *European Journal of Operational Research*, 119(2):440 – 450, 1999.

Quang-Vinh Dang and Khoa Pham. Design of a footwear assembly line using simulation-based alns. *Procedia CIRP*, 40:596 – 601, 2016.

Ezey M Dar-El. Mixed-model assembly line sequencing problems. *Omega*, 6(4):313 – 323, 1978.

Yılmaz Delice, Emel Kızılkaya Aydoğan, Uğur Özcan, and Mehmet Sıtkı İlkay. A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. *Journal of Intelligent Manufacturing*, 28(1):23 – 36, 2014.

Alexandre Dolgui, Nikolai Guschinsky, and Genrikh Levin. A special case of transfer lines balancing by graph approach. *European Journal of Operational Research*, 168(3):732 –746, 2006.

Jietao Dong, Linxuan Zhang, Tianyuan Xiao, and Huachao Mao. Balancing and sequencing of stochastic mixed-model assembly u-lines to minimise the expectation of work overload time. *International Journal of Production Research*, 52(24): 7529 – 7548, 2014.

Johann Dréo, Alain Pétrowski, Patrick Siarry, and Eric Taillard. *Metaheuristics for hard optimization: methods and case studies*. Springer Science & Business Media, 2006.

Rene Driessel and Lars Mönch. Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times. *Computers & Industrial Engineering*, 61(2): 336 – 345, 2011.

EA Duplaga and DJ Bragg. Mixed-model assembly line sequencing heuristics for smoothing component parts usage: a comparative analysis. *International Journal of Production Research*, 36(8):2209 – 2224, 1998.

Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39 – 43. IEEE, 1995.

Tarek A El-Mihoub, Adrian A Hopgood, Lars Nolle, and Alan Battersby. Hybrid genetic algorithms: A review. *Engineering Letters*, 13(2):124 – 137, 2006.

Erdal Erel and Hadi Gokcen. Shortest-route formulation of mixed-model assembly line balancing problem. *European Journal of Operational Research*, 116(1):194 – 204, 1999.

Erdal Erel and Subhash C. Sarin. A survey of the assembly line balancing procedures. *Production Planning & Control*, 9(5):414 – 434, jan 1998.

Erdal Erel, Yasin Gocgun, and İhsan Sabuncuoğlu. Mixed-model assembly line sequencing using beam search. *International Journal of Production Research*, 45 (22):5265 – 5284, 2007.

S Esquivel, S Ferrero, R Gallard, Carolina Salto, Hugo Alfonso, and Martin Schütz. Enhanced evolutionary algorithms for single and multiobjective optimization in the job shop scheduling problem. *Knowledge-Based Systems*, 15(1-2):13 – 25, 2002.

Emanuel Falkenauer and Alain Delchambre. A genetic algorithm for bin packing and line balancing. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 1186 – 1192. IEEE, 1992.

Kun Fan and Ren Qian Zhang. Notice of retraction an analysis of research in job shop scheduling problem (2000–2009). In *Advanced Management Science (ICAMS), 2010 IEEE International Conference on*, volume 1, pages 282 – 288. IEEE, 2010.

Parviz Fattahi and Arezoo Askari. A multi-objective mixed-model assembly line sequencing problem with stochastic operation time. *Journal of Optimization in Industrial Engineering*, 11(1):157 – 167, 2018.

J Soeiro Ferreira. Multimethodology in metaheuristics. *Journal of the Operational Research Society*, 64(6):873 – 883, 2013.

Jie Gao, Linyan Sun, and Mitsuo Gen. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, 35(9):2892 – 2907, 2008.

Mitsuo Gen and Lin Lin. Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey. *Journal of Intelligent Manufacturing*, 25(5):849 – 866, 2014.

Soumen Ghosh and Roger J. Gagnon. A comprehensive literature review and anal-

ysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, 27(4):637 – 670, april 1989.

Hadi Gökçen and Erdal Erel. Binary integer formulation for mixed-model assembly line balancing problem. *Computers & Industrial Engineering*, 34(2):451, 1998.

LBdeM Guimarães, JLD Ribeiro, JS Renner, and PAB De Oliveira. Worker evaluation of a macroergonomic intervention in a brazilian footwear company. *Applied Ergonomics*, 45(4):923 – 935, 2014.

Allan L Gutjahr and George LNemhauser. An algorithm for the line balancing problem. *Management Science*, 11(2):308 – 315, 1964.

Prabhat Hajela and C Y Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4(2):99 – 107, 1992.

Nima Hamta, SMT Fatemi Ghomi, F Jolai, and M Akbarpour Shirazi. A hybrid pso algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *International Journal of Production Economics*, 141(1):99 – 111, 2013.

Alper Hamzadayi and Gokalp Yildiz. A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model u-lines with parallel workstations and zoning constraints. *Computers & Industrial Engineering*, 62(1):206 – 215, 2012.

Alper Hamzadayi and Gokalp Yildiz. A simulated annealing algorithm based approach for balancing and sequencing of mixed-model u-lines. *Computers & Industrial Engineering*, 66(4):1070 – 1084, 2013.

A Noorul Haq, J Rengarajan, and K Jayaprakash. A hybrid genetic algorithm approach to mixed-model assembly line balancing. *The International Journal of Advanced Manufacturing Technology*, 28(3-4):337 – 341, 2006.

Jens Heger, Jürgen Branke, Torsten Hildebrandt, and Bernd Scholz-Reiter. Dynamic adjustment of dispatching rule parameters in flow shops with sequence-

dependent set-up times. *International Journal of Production Research*, 54(22): 6812 – 6824, 2016.

Albrecht Heinric. A comparison between simulated annealing and tabu search with an example from the production planning. In *Operations Research Proceedings 1993*, pages 498 – 503. Springer Berlin Heidelberg, 1994.

W. B. Helgeson and Dunbar P. Birnie. Assembly line balancing using the ranked positional weight technique. *The Journal of Industrial Engineering*, 12(6):394 – 398, 1961.

Thomas R Hoffmann. Eureka: A hybrid system for assembly line balancing. *Management Science*, 38(1):39 – 47, 1992.

John H Holland. Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press*, pages 439 – 444, 1975.

Oliver Holthaus and Chandrasekharan Rajendran. Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics*, 48(1): 87 – 105, 1997.

Jing Huang and Gürsel A Süer. A dispatching rule-based genetic algorithm for multi-objective job shop scheduling using fuzzy satisfaction levels. *Computers & Industrial Engineering*, 86:29 – 42, 2015.

Jing Huang, Gürsel A Süer, and Shravan BR Urs. Genetic algorithm for rotary machine scheduling with dependent processing times. *Journal of Intelligent Manufacturing*, 23(5):1931 – 1948, 2012.

Kuo-Ling Huang and Ching-Jong Liao. Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers & Operations Research*, 35(4):1030 – 1046, 2008.

ReaKook Hwang and Hiroshi Katayama. Integrated procedure of balancing and

sequencing for mixed-model assembly lines: a multi-objective evolutionary approach. *International Journal of Production Research*, 48(21):6417 – 6441, 2010.

J Jayaprakash, K Manoj, and P Ambedkar. Simulation of mixed model assembly line sequencing using pro-model software. *International Journal of Applied Engineering Research*, 10(68), 2015.

J Jerald, Prabaharan Asokan, G Prabaharan, and R Saravanan. Scheduling optimisation of flexible manufacturing systems using particle swarm optimisation algorithm. *The International Journal of Advanced Manufacturing Technology*, 25 (9-10):964 – 971, 2005.

Lu Jian-sha, Jiang Ling-ling, and Li Xiu-lin. Hybrid particle swarm optimization algorithm for assembly line balancing problem-2. In *Industrial Engineering and Engineering Management, 2009. IE&EM'09. 16th International Conference on*, pages 979 – 983. IEEE, 2009.

Dou Jianping, Su Chun, and L Jun. A discrete particle swarm optimization algorithm for assembly line balancing problem of type 1. In *Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on*, volume 1, pages 44 – 47. IEEE, 2011.

Roger V Johnson. A branch and bound algorithm for assembly line balancing problems with formulation irregularities. *Management Science*, 29(11):1309 – 1324, 1983.

Imed Kacem, Slim Hammadi, and Pierre Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(1):1 – 13, 2002.

Yakup Kara. Line balancing and model sequencing to reduce work overload in mixed-model u-line production environments. *Engineering Optimization*, 40(7): 669 – 684, 2008.

Yakup Kara, Ugur Ozcan, and Ahmet Peker. An approach for balancing and sequencing mixed-model jit u-lines. *The International Journal of Advanced Manufacturing Technology*, 32(1-12):218 – 1231, 2007a.

Yakup Kara, Ugur Ozcan, and Ahmet Peker. Balancing and sequencing mixed-model just-in-time u-lines with multiple objectives. *Applied Mathematics and Computation*, 184(2):566 – 588, 2007b.

Selçuk Karabatı and Serpil Saıin. Assembly line balancing in a mixed-model sequencing environment with synchronous transfers. *European Journal of Operational Research*, 149(2):417 – 429, 2003.

Talip Kellegöz. Balancing lexicographic multi-objective assembly lines with multi-manned stations. *Mathematical Problems in Engineering*, 2016, 2016.

Yeo Keun Kim, Yong Ju Kim, and Yeongho Kim. Genetic algorithms for assembly line balancing with various objectives. *Computers & Industrial Engineering*, 30 (3):397, 1996.

Yeo Keun Kim, Jae Yun Kim, and Yeongho Kim. A coevolutionary algorithm for balancing and sequencing in mixed model assembly lines. *Applied Intelligence*, 13(3):247 – 258, 2000a.

Yeo Keun Kim, Sun Jin Kim, and Jae Yun Kim. Balancing and sequencing mixed-model u-lines with a co-evolutionary algorithm. *Production Planning & Control*, 11(8):754 – 764, 2000b.

Yeo Keun Kim, Jae Yun Kim, and Yeongho Kim. An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model u-lines. *European Journal of Operational Research*, 168(3):838 – 852, 2006.

Yeo Keun Kim, Won Seop Song, and Jun Hyuk Kim. A mathematical model and a genetic algorithm for two-sided assembly line balancing. *Computers & Operations Research*, 36(3):853 – 865, 2009.

Nuchsara Kriengkorakot and Nalin Pianthong. The assembly line balancing problem : Review articles. *KKU Engineering Journal*, 34(2):133 –140, 2007.

Wen-Yang Ku and J Christopher Beck. Mixed integer programming models for job shop scheduling: a computational analysis. *Computers & Operations Research*, 73:165 – 173, 2016.

Ibrahim Kucukkoc and David Z Zhang. Mathematical model and agent based solution approach for the simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. *International Journal of Production Economics*, 158:314 – 333, 2014a.

Ibrahim Kucukkoc and David Z Zhang. Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. *International Journal of Production Research*, 52(12):3665 – 3687, 2014b.

Ibrahim Kucukkoc and David Z Zhang. Mixed-model parallel two-sided assembly line balancing problem: A flexible agent-based ant colony optimization approach. *Computers & Industrial Engineering*, 97:58 – 72, 2016.

Xian kun Lin, Ai ping Li, and Bin sen Chen. Scheduling optimization of mixed model assembly lines with hybrid particle swarm optimization algorithm. *Industrial Engineering and Management*, 1(011), 2006.

Sophie D. Lapierre, Angel Ruiz, and Patrick Soriano. Balancing assembly lines with tabu search. *European Journal of Operational Research*, 168(3):826 – 837, 2006.

Deming Lei. Multi-objective production scheduling: a survey. *The International Journal of Advanced Manufacturing Technology*, 43(9 - 10):926, 2009.

Jun-Qing Li, Quan-Ke Pan, and Yun-Chia Liang. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, 59(4):647 – 662, 2010.

Jun-Qing Li, Quan-Ke Pan, and Kai-Zhou Gao. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems.

The International Journal of Advanced Manufacturing Technology, 55(9-12):1159 – 1169, 2011.

Jun-Qing Li, Quan-Ke Pan, and M Fatih Tasgetiren. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Applied Mathematical Modelling*, 38(3):1111 – 1132, 2014.

Zixiang Li, Mukund Nilakantan Janardhanan, Qiuhua Tang, and Peter Nielsen. Mathematical model and metaheuristics for simultaneous balancing and sequencing of a robotic mixed-model assembly line. *Engineering Optimization*, 50(5):877 – 893, 2018.

Kunlei Lian, Chaoyong Zhang, Liang Gao, and Xinyu Shao. A modified colonial competitive algorithm for the mixed-model u-line balancing and sequencing problem. *International Journal of Production Research*, 50(18):5117 – 5131, 2012.

Xiu lin LI, Jian sha LU, Guo zhong CHAI, and Hong tao TANG. Modified artificial bee colony optimization for mixed assembly line sequencing problem [j]. *Computer Integrated Manufacturing Systems*, 12, 2011.

Taicir Loukil, Jacques Teghem, and Philippe Fortemps. A multi-objective production scheduling case study solved by simulated annealing. *European Journal of Operational Research*, 179(3):709 – 722, 2007.

AA Mamun, AA Khaled, SM Ali, and MM Chowdhury. A heuristic approach for balancing mixed-model assembly line of type i using genetic algorithm. *International Journal of Production Research*, 50(18):5106 – 5116, 2012.

Neda Manavizadeh, Masoud Rabbani, and Farzad Radmehr. A new multi-objective approach in order to balancing and sequencing u-shaped mixed model assembly line problem: a proposed heuristic algorithm. *The International Journal of Advanced Manufacturing Technology*, 79(1-4):415 – 425, 2015.

Vittorio Maniezzo, Thomas StÃijtzle, and Stefani VoÃ§. *Matheuristics hybridizing*

metaheuristics and mathematical programming. Springer New York Dordrecht Heidelberg London, 2009. ISBN 978-1-4419-1305-0.

S Afshin Mansouri. A multi-objective genetic algorithm for mixed-model sequencing on jit assembly lines. *European Journal of Operational Research*, 167(3):696 – 716, 2005.

Scott J Mason, John W Fowler, and W Matthew Carlyle. A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. *Journal of Scheduling*, 5(3):247 – 262, 2002.

Gökan May, Bojan Stahl, Marco Taisch, and Vittal Prabhu. Multi-objective genetic algorithm for energy-efficient job shop scheduling. *International Journal of Production Research*, 53(23):7071 – 7089, 2015.

Nuntana Mayteekrieangkrai and Wuthichai Wongthatsanekorn. Optimized ready mixed concrete truck scheduling for uncertain factors using bee algorithm. *Songklanakarin Journal of Science & Technology*, 37(2), 2015.

Patrick R McMullen. An ant colony optimization approach to addressing a jit sequencing problem with multiple objectives. *Artificial Intelligence in Engineering*, 15(3):309 – 317, 2001.

Patrick R McMullen and Gregory V Frazierss. A simulated annealing approach to mixed-model sequencing with multiple objectives on a just-in-time line. *Iie Transactions*, 32(8):679 – 686, 2000.

Patrick R McMullen and Peter Tarasewich. Using ant techniques to solve the assembly line balancing problem. *Iie Transactions*, 35(7):605–617, 2003.

Mohammad Saidi Mehrabad, Saeed Dehnavi Arani, Farshid Evazabadian, and Vahid Mahmoodian. An ant colony algorithm (aca) for solving the new integrated model of job shop scheduling and conflict-free routing of agvs. *Computers & Industrial Engineering*, 86:2 – 13, 2015.

C. Merengo, F. Nava, and A. Pozzetti. Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research*, 37(12):2835 – 2860, aug 1999.

Süleyman Mete, Zeynel Abidin Çil, Kürşad Ağpak, Eren Özceylan, and Alexandre Dolgui. A solution approach based on beam search algorithm for disassembly line balancing problem. *Journal of Manufacturing Systems*, 41:188 – 200, 2016.

Zbigniew Michalewicz and David B Fogel. *How to solve it: modern heuristics*. Springer Science & Business Media, 2013.

J Miltenburg, G Steiner, and S Yeomans. A dynamic programming algorithm for scheduling mixed-model, just-in-time production systems. *Mathematical and Computer Modelling*, 13(3):57 – 66, 1990.

John Miltenburg. Balancing and scheduling mixed-model u-shaped production lines. *International Journal of Flexible Manufacturing Systems*, 14(2):119 – 151, 2002.

Cristobal Miralles, Jose Pedro Garcia-Sabater, Carlos Andres, and Manuel Cardos. Advantages of assembly lines in sheltered work centres for disabled. a case study. *International Journal of Production Economics*, 110(1-2):187 – 197, 2007.

Cristóbal Miralles, José P García-Sabater, Carlos Andrés, and Manuel Cardós. Branch and bound procedures for solving the assembly line worker assignment and balancing problem: Application to sheltered work centres for disabled. *Discrete Applied Mathematics*, 156(3):352 – 367, 2008.

Seyed Mohammed Mirghorbani, Masoud Rabbani, Reza Tavakkoli-Moghaddam, and Alireza R Rahimi-Vahed. A multi-objective particle swarm for a mixed-model assembly line sequencing. In *Operations Research Proceedings 2006*, pages 181 – 186. Springer, 2007.

John E Mitchell. Branch and cut. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100, 1997.

Ethel Mokotoff. Parallel machine scheduling problems: A survey. *Asia-Pacific Journal of Operational Research*, 18(2):193, 2001.

Mayron César O. Moreira, Marcus Ritt, Alysson M. Costa, and Antonio A. Chaves. Simple heuristics for the assembly line worker assignment and balancing problem. *Journal of Heuristics*, 18(3):505 – 524, 2012.

Mayron César O. Moreira, Rafael Pastor, Alysson M. Costa, and Cristóbal Miralles. The multi-objective assembly line worker integration and balancing problem of type-2. *Computers & Operations Research*, 82:114 –125, 2017.

H Mosadegh, SMT Fatemi Ghomi, and M Zandieh. Simultaneous solving of balancing and sequencing problems in mixed-model assembly line systems. *International Journal of Production Research*, 50(18):4994 – 5016, 2012a.

H Mosadegh, Mostafa Zandieh, and SMT Fatemi Ghomi. Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines. *Applied Soft Computing*, 12(4):1359 – 1370, 2012b.

H Mosadegh, SMT Fatemi Ghomi, and GA Süer. Heuristic approaches for mixed-model sequencing problem with stochastic processing times. *International Journal of Production Research*, 55(10):2857 – 2880, 2017.

Özcan Mutlu, Olcay Polat, and Aliye Ayca Supciller. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-ii. *Computers & Operations Research*, 40(1):418 – 426, 2013.

Andreas C Nearchou. Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. *International Journal of Production Economics*, 129(2):242 – 250, 2011.

WC Ng and KL Mak. A branch and bound algorithm for scheduling just-in-time

mixed-model assembly lines. *International Journal of Production Economics*, 33 (1-3):169 – 183, 1994.

Gaia Nicosia, Dario Pacciarelli, and Andrea Pacifici. Optimally balancing assembly lines with different workstations. *Discrete Applied Mathematics*, 118(1):99 – 113, 2002.

J Mukund Nilakantan, Zixiang Li, Qiuhua Tang, and Peter Nielsen. Milp models and metaheuristic for balancing and sequencing of mixed-model two-sided assembly lines. *European Journal of Industrial Engineering*, 11(3):353 – 379, 2017.

Maroua Nouiri, Abdelghani Bekrar, Abderezak Jemai, Smail Niar, and Ahmed Chiheb Ammari. An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, 29(3):603 – 615, 2018.

Eugeniusz Nowicki and Czesław Smutnicki. An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*, 8(2):145 – 159, 2005.

Mehmet Kursat Oksuz, Kadir Buyukozkan, and Sule Itir Satoglu. U-shaped assembly line worker assignment and balancing problem: A mathematical model and two meta-heuristics. *Computers & Industrial Engineering*, 112:246 – 263, 2017.

F. S. Oliveira, K. Vittori, R. M. O. Russel, and X. L. Travassos. Mixed assembly line rebalancing: A binary integer approach applied to real world problems in the automotive industry. *International Journal of Automotive Technology*, 13(6):933, 2012.

Andrzej Osyczka. Multicriteria optimization for engineering design. *Design Optimization*, 1:193 – 227, 1985.

Peng Si Ow and Thomas E Morton. Filtered beam search in scheduling. *The International Journal Of Production Research*, 26(1):35 – 62, 1988.

Uğur Özcan, Talip Kellegöz, and Bilal Toklu. A genetic algorithm for the stochastic

mixed-model u-line balancing and sequencing problem. *International Journal of Production Research*, 49(6):1605 – 1626, 2011.

Uğur Özcan and Bilal Toklui. Balancing of mixed-model two-sided assembly lines. *Computers & Industrial Engineering*, 57(1):217 – 227, 2009.

Uğur Özcan, Hakan Çerçoğlu, Hadi Gökçen, and Bilal Toklui. A tabu search algorithm for the parallel assembly line balancing problem. *Gazi University Journal of Science*, 22(4):313 – 323, 2009.

Uğur Özcan, Hakan Çerçoğlu, Hadi Gökçen, and Bilal Toklu. Balancing and sequencing of parallel mixed-model assembly lines. *International Journal of Production Research*, 48(17):5089 – 5113, 2010a.

Uğur Özcan, Hadi Gökçen, and Bilal Toklu. Balancing parallel two-sided assembly lines. *International Journal of Production Research*, 48(16):4767 – 4784, aug 2010b.

Cemalettin Ozturk, Semra Tunali, Brahim Hnich, and Arslan M Ornek. Simultaneous balancing and scheduling of flexible mixed model assembly lines with sequence-dependent setup times. *Electronic Notes in Discrete Mathematics*, 36: 65 – 72, 2010.

Cemalettin Öztürk, Semra Tunalı, Brahim Hnich, and Arslan Örnek. Balancing and scheduling of flexible mixed model assembly lines. *Constraints*, 18(3):434 – 469, 2013a.

Cemalettin Öztürk, Semra Tunalı, Brahim Hnich, and Arslan Örnek. Balancing and scheduling of flexible mixed model assembly lines with parallel stations. *The International Journal of Advanced Manufacturing Technology*, 67(9-12):2577 – 2591, 2013b.

Bijaya Ketan Panigrahi, Yuhui Shi, and Meng-Hiot Lim. *Handbook of swarm intelligence: concepts, principles and applications*, volume 8. Springer Science & Business Media, 2011.

Rafael Pastor. Lb-albp: the lexicographic bottleneck assembly line balancing problem. *International Journal of Production Research*, 49(8):2425 – 2442, 2011.

Rafael Pastor and Laia Ferrer. An improved mathematical program to solve the simple assembly line balancing problem. *International Journal of Production Research*, 47(11):2943 – 42959, 2009.

Jordi Pereira. The robust (minmax regret) assembly line worker assignment and balancing problem. *Computers & Operations Research*, 2018.

F Pezzella, G Morganti, and G Ciaschetti. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, 35(10):3202 – 3212, 2008.

Michael Pinedo and Xiuli Chao. *Operations scheduling with applications in manufacturing and services*. McGraw Hill, 1999.

Michael L Pinedo. *Scheduling: theory, algorithms, and systems*. Springer, 2016.

Anulark Pinnoi and Wilbert E Wilhelm. Assembly system design: A branch and cut approach. *Management Science*, 44(1):103 – 118, 1998.

Anulark Pinnoi and Wilbert F Wilhelm. A branch and cut approach for workload smoothing on assembly lines. *INFORMS Journal on Computin*, 9(4):335 – 350, 1997.

Olcay Polat, Can B. Kalayci, Özcan Mutlu, and Surendra M. Gupta. A two-phase variable neighbourhood search algorithm for assembly line worker assignment and balancing problem type-ii: an industrial case study. *International Journal of Production Research*, 54(3):722 – 741, 2016.

Portuguese-Shoes. Porto2018. `https://www.porto2018.uitic.org/info/-international-markets-exporting-more-than-95/24.htm`, Jun 2018.

Nguyen Thi Phuong Quyen, James C Chen, and Chao-Lung Yang. Hybrid genetic algorithm to solve resource constrained assembly line balancing problem in footwear manufacturing. *Soft Computing*, 21(21):6279 – 6295, 2017.

Masoud Rabbani, Mehdi Yazdanbakhsh, and Hamed Farrokhi-Asl. Solving a multi-objective mixed-model assembly line balancing and sequencing problem. 2017.

Alireza Rahimi-Vahed and Ali Hossein Mirzaei. A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem. *Computers & Industrial Engineering*, 53(4):642 – 666, 2007.

AR Rahimi-Vahed, SM Mirghorbani, and Masoud Rabbani. A new particle swarm algorithm for a multi-objective mixed-model assembly line sequencing problem. *Soft computing*, 11(10):997 – 1012, 2007.

Senthilkumar Ramalingam and Ramkumar Anna Subramanian. Solving level scheduling in mixed model assembly line by simulated annealing method. *Circuits and Systems*, 7(6), 2016.

Mudassa Rauf, Shoaib Sarfraz, Essam Shehab, Saima Javeed, and Mirza Jahanzaib. Multi-objective optimization or models sequencing in mixed-model assembly lines. In *Advances in Manufacturing Technology XXXI: Proceedings of the 15th International Conference on Manufacturing Research, Incorporating the 32nd National Conference on Manufacturing Research, September 5–7, 2017, University of Greenwich, UK*, volume 6, page 96. IOS Press, 2017.

Maher Rebai, Imed Kacem, and Kondo H Adjallah. Earliness–tardiness minimization on a single machine to schedule preventive maintenance tasks: metaheuristic and exact methods. *Journal of Intelligent Manufacturing*, 23(4):1207 – 1224, 2012.

Patrick Reed, Barbara S Minsker, and David E Goldberg. Simplifying multiobjective optimization: An automated design methodology for the nondominated sorted genetic algorithm-ii. *Water Resources Research*, 39(7), 2003.

Colin R Reeves. *Modern heuristic techniques for combinatorial problems*. John Wiley and Sons, 1993.

V Roshanaei, B Naderi, Fariborz Jolai, and M Khalili. A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. *Future Generation Computer Systems*, 25(6):654 – 661, 2009.

Abdolreza Roshani, Parviz Fattahi, Abdolhassan Roshani, Mohsen Salehi, and Arezoo Roshani. Cost-oriented two-sided assembly line balancing problem: A simulated annealing approach. *International Journal of Computer Integrated Manufacturing*, 25(8):689 – 715, 2012.

Ihsan Sabuncuoglu, Erdal Erel, and Arda Alp. Ant colony optimization for the single model u-type assembly line balancing problem. *International Journal of Production Economics*, 120(2):287 – 300, 2009.

Mohammad Saidi-Mehrabad and Parviz Fattah. Flexible job shop scheduling with tabu search algorithms. *The International Journal of Advanced Manufacturing Technology*, 32(5):563 – 570, 2007.

Ullah Saif, Zailin Guan, Weiqi Liu, Baoxi Wang, and Chaoyong Zhang. Multi-objective artificial bee colony algorithm for simultaneous sequencing and balancing of mixed model assembly line. *The International Journal of Advanced Manufacturing Technology*, 75(9-12):1809 – 1827, 2014a.

Ullah Saif, Zailin Guan, Baoxi Wang, Jahanzeb Mirza, and Shiyang Huang. A survey on assembly lines and its types. *Frontiers of Mechanical Engineering*, 9 (2):95 – 105, may 2014b.

Ullah Saif, Zailin Guan, Li Zhang, Fei Zhang, Baoxi Wang, and Jahanzaib Mirza. Multi-objective artificial bee colony algorithm for order oriented simultaneous sequencing and balancing of multi-mixed model assembly line. *Journal of Intelligent Manufacturing*, pages 1 – 26, 2017.

Maryam Salehi, Hamid Reza Maleki, and Sadegh Niroomand. A multi-objective assembly line balancing problem with worker's skill and qualification considerations in fuzzy environment. *Applied Intelligence*, pages 1 – 20, 2017.

Melvin E. Salveson. The assembly line balancing problem. *The Journal of Industrial Engineering*, pages 18 – 25, 1955.

Tadeusz Sawik. Monolithic vs. hierarchical balancing and scheduling of a flexible assembly line. *European Journal of Operational Research*, 143(1):115 – 124, 2002.

Armin Scholl and Christian Becker. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168(3):666 – 693, feb 2006.

Armin Scholl and Robert Klein. Balancing assembly lines effectivelyâĂŞa computational comparison. *European Journal of Operational Research*, 114(1):50âĂŞ58, 1999.

Armin Scholl and Stefan Voß. Simple assembly line balancing-heuristic approaches. *Journal of Heuristics*, 2(3):217 – 244, 1997.

Edward C Sewell and Sheldon H Jacobson. A branch, bound, and remember algorithm for the simple assembly line balancing problem. *INFORMS Journal on Computing*, 24(3):433 – 442, 2012.

DY Sha and Cheng-Yu Hsu. A hybrid particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering*, 51(4):791 – 808, 2006.

Wanxing Sheng, Ke-Yan Liu, Yuan Liu, Xiaoli Meng, and Yunhua Li. Optimal placement and sizing of distributed generation via an improved nondominated sorting genetic algorithm ii. *IEEE Transactions on Power Delivery*, 30(2):569 – 578, 2015.

N Shivasankaran, P Senthil Kumar, and K Venkatesh Raja. Hybrid sorting immune simulated annealing algorithm for flexible job shop scheduling. *International Journal of Computational Intelligence Systems*, 8(3):455 – 466, 2015.

Ana S. Simaria and Pedro M. Vilarinho. 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. *Computers & Industrial Engineering*, 56(2):489 – 506, mar 2009.

Ana Sofia Simaria and Pedro M. Vilarinho. A genetic algorithm based approach to the mixed-model assembly line balancing problem of type ii. *Computers & Industrial Engineering*, 47.(4):391 – 407, 2004.

P Sivasankaran and P Shahabudeen. Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology*, 73 (9-12):1665 – 1694, 2014.

Panneerselvam Sivasankaran and Peer Mohamed Shahabudeen. Genetic algorithm for concurrent balancing of mixed-model assembly lines with original task times of models. *Intelligent Information Management*, 5:84 – 92, 2013.

D Sparling and J Miltenburg. The mixed-model u-line balancing problem. *International Journal of Production Research*, 36(2):485 – 501, 1998.

W Stadler. *Recent Advances and Historical Development of Vector Optimization*, chapter Initiators of multicriteria optimization, pages 3 – 47. Springer, 1987.

DJ Stockton, Liam Quinn, and Riham A Khalil. Use of genetic algorithms in operations management: Part 1: Applications. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 218(3):315 – 327, 2004.

Ping Su and Ye Lu. Combining genetic algorithm and simulation for the mixed-model assembly line balancing problem. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 4, pages 314 – 318. IEEE, 2007.

Shyam Sundar, Ponnuthurai N Suganthan, Chua Tay Jin, Cai Tian Xiang, and Chong Chin Soon. A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint. *Soft Computing*, 21(5):1193 – 1202, 2017.

Banu Sungur and Yasemin Yavuz. Assembly line balancing with hierarchical worker assignment. *Journal of Manufacturing Systems*, 37:290 – 298, 2015.

G Suresh and S Sahu. Stochastic assembly line balancing using simulated annealing. *The International Journal of Production Research*, 32(8):1801 – 1810, 1994.

RK Suresh and KM Mohanasundaram. Pareto archived simulated annealing for job shop scheduling with multiple objectives. *The International Journal of Advanced Manufacturing Technology*, 29(1):184 – 196, 2006.

El-Ghazal Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley and Sons, 2009.

Qiuhua Tang, Zixiang Li, and Liping Zhang. An effective discrete artificial bee colony algorithm with idle time reduction techniques for two-sided assembly line balancing problem of type-ii. *Computers and Industrial Engineering*, 97:146 –156, 2016.

Pınar Tapkan, Lale Özbakır, and Adil Baykasoğlu. Bee algorithms for parallel two-sided assembly line balancing problem with walking times. *Applied Soft Computing*, 39:275 – 291, 2016.

Seren Ozmehmet Tasan and Semra Tunali. A review of the current applications of genetic algorithms in assembly line balancing. *Journal of Intelligent Manufacturings*, 19(1):49 – 69, 2008.

Joc Cing Tay and Nhu Binh Ho. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering*, 54(3):453 – 473, 2008.

Nick T Thomopoulos. Line balancing-sequencing for mixed-model assembly. *Management Science*, 14(2):B – 59, 1967.

Li-Hui Tsai. Mixed-model sequencing to minimize utility work and the risk of conveyor stoppage. *Management Science*, 41(3):485 – 495, 1995.

Mohammad Kamal Uddin and Jose Luis Martinez Lastra. Assembly line balancing and sequencing. In *Assembly Line-Theory and Practice*. InTech, 2011.

Mohammad Kamal Uddin, Marian Cavia Soto, and Jose L Martinez Lastra. An integrated approach to mixed-model assembly line balancing and sequencing. *Assembly Automation*, 30(2):164 – 172, 2010.

Berna Ulutas and A Attila Islier. Dynamic facility layout problem in footwear industry. *Journal of Manufacturing Systems*, 36:55 – 61, 2015.

European Union. European commission, the eu footwear industry. `https://ec.europa.eu/growth/sectors/fashion/footwear/eu-industry\textunderscoreen`, July 2018.

Pedro M. Vilarinho and Ana Sofia Simaria. A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research*, 40.(6):1405 – 1420, 2002.

Pedro M Vilarinho and Ana Sofia Simaria. Antbal: An ant colony optimization algorithm for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research*, 44(2):291 – 303, 2006.

S Voss, V Maniezzo, and T Stützle. Matheuristics: Hybridizing metaheuristics and mathematical programming (annals of information systems). 2009.

Lei Wang, Jingcao Cai, Ming Li, and Zhihu Liu. Flexible job shop scheduling problem using an improved ant colony optimization. *Scientific Programming*, 2017, 2017.

Ling Wang, Gang Zhou, Ye Xu, Shengyao Wang, and Min Liu. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 60(1-4):303 – 315, 2012.

Masato Watanabe, Kenichi Ida, and Mitsuo Gen. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. *Computers & Industrial Engineering*, 48(4):743 – 752, 2005.

Tse Chiu Wong, Felix TS Chan, and LY Chan. A resource-constrained assembly job shop scheduling problem with lot streaming technique. *Computers & Industrial Engineering*, 57(3):983 – 995, 2009.

Zhao Xiaobo and Katsuhisa Ohno. A sequencing problem for a mixed-model assembly line in a jit production system. *Computers & Industrial Engineering*, 27 (1-4):71 – 74, 1994.

Mehdi Yazdani, Maghsoud Amiri, and Mostafa Zandieh. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, 37(1):678 – 687, 2010.

Olga Yugay, Insoo Kim, Beomjune Kim, and Franz IS Kot. Hybrid genetic algorithm for solving traveling salesman problem with sorted population. In *2008 Third International Conference on Convergence and Hybrid Information Technology*, pages 1024 – 1028. IEEE, 2008.

Andrea Zangiacomi, Liao Zhijian, Marco Sacco, and Claudio Roberto Boër. Process planning and scheduling for mass customised shoe manufacturing. *International Journal of Computer Integrated Manufacturing*, 17(7):613–621, 2004.

Chao Yong Zhang, PeiGen Li, YunQing Rao, and ZaiLin Guan. A very fast ts/sa algorithm for the job shop scheduling problem. *Computers & Operations Research*, 35(1):282 – 294, 2008.

ChaoYong Zhang, PeiGen Li, ZaiLin Guan, and YunQing Rao. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, 34(11):3229 – 3242, 2007.

Guohui Zhang, Lingjie Zhang, Xiaohui Song, Yongcheng Wang, and Chi Zhou. A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem. *Cluster Computing*, pages 1 – 12, 2018.

Rui Zhang and Cheng Wu. A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing*, 10(1):79 – 89, 2010.

Rui Zhang, Shiji Song, and Cheng Wu. A hybrid artificial bee colony algorithm for the job shop scheduling problem. *International Journal of Production Economics*, 141(1):167 – 178, 2013.

Yudong Zhang, Shuihua Wang, and Genlin Ji. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015, 2015.

Boxuan Zhao, Jianmin Gao, Kun Chen, and Ke Guo. Two-generation pareto ant colony algorithm for multi-objective job shop scheduling problem with alternative process plans and unrelated parallel machines. *Journal of Intelligent Manufacturing*, 29(1):93 – 108, 2018.

Erming Zhou, Jin Zhu, and Ling Deng. Flexible job-shop scheduling based on genetic algorithm and simulation validation. In *MATEC Web of Conferences*, volume 100, page 02047. EDP Sciences, 2017.

Qiong Zhu and Jie Zhang. Ant colony optimisation with elitist ant for sequencing problem in a mixed model assembly line. *International Journal of Production Research*, 49(15):4605 – 4626, 2011.

Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*. PhD dissertation, Swiss Federal Institute of Technology Zurich, 1999.