A real-time simulation-based optimisation environment for industrial scheduling

Marcus Frantzén

A dissertation submitted in partial fulfilment of the requirements for the degree of:

Doctor of Philosophy

April 2013

Sponsored by the University of Skövde, Sweden

Abstract

In order to cope with the challenges in industry today, such as changes in product diversity and production volume, manufacturing companies are forced to react more flexibly and swiftly. Furthermore, in order for them to survive in an ever-changing market, they also need to be highly competitive by achieving near optimal efficiency in their operations. Production scheduling is vital to the success of manufacturing systems in industry today, because the near optimal allocation of resources is essential in remaining highly competitive.

The overall aim of this study is the advancement of research in manufacturing scheduling through the exploration of more effective approaches to address complex, real-world manufacturing flow shop problems. The methodology used in the thesis is in essence a combination of systems engineering, algorithmic design and empirical experiments using real-world scenarios and data. Particularly, it proposes a new, web services-based, industrial scheduling system framework, called OPTIMISE Scheduling System (OSS), for solving real-world complex scheduling problems. OSS, as implemented on top of a generic web services-based simulation-based optimisation (SBO) platform called OPTIMISE, can support near optimal and real-time production scheduling in a distributed and parallel computing environment. Discrete-event simulation (DES) is used to represent and flexibly cope with complex scheduling problems without making unrealistic assumptions which are the major limitations of existing scheduling methods proposed in the literature. At the same time, the research has gone beyond existing studies of simulation-based scheduling applications, because the OSS has been implemented in a real-world industrial environment at an automotive manufacturer, so that qualitative evaluations and quantitative comparisons of scheduling methods and algorithms can be made with the same framework.

Furthermore, in order to be able to adapt to and handle many different types of real-world scheduling problems, a new hybrid meta-heuristic scheduling algorithm that combines priority dispatching rules and genetic encoding is proposed. This combination is demonstrated to be able to handle a wider range of problems or a current scheduling problem that may change over time, due to the flexibility requirements in the real-world. The novel hybrid genetic representation has been demonstrated effective through the evaluation in the real-world scheduling problem using real-world data.

Acknowledgements

I would like to express my gratitude to my academic supervisors, Professor Philip Moore, Professor David Stockton and Doctor Amos Ng, for their support and guidance during my research study. Special thanks to Amos who made me see the possibilities rather than the difficulties and gave me valuable comments during my research.

Many thanks also to all of my colleagues at the Centre for Intelligent Automation who supported me and gave me a great place at which to work. Among my colleagues, I particularly thank: Martin Andersson for helping me realise many of my ideas; Mats Jägstam who convinced me to continue with research studies; Anna Syberfeldt for her good company and for ensuring that I had the time to complete my thesis; Tehseen Aslam, Matias Urenda Moris, Catarina Dudas, and Leif Pehrsson who shared the research journey with me; Ingemar Karlsson, and Jacob Bernedixen for their good company and for their support, and Vera Lindroos for her language help in this thesis.

My gratitude to the University of Skövde for its financial support and to the partner company for its extensive involvement in this research study. Their contribution made completing the thesis possible.

I would also like to express my gratitude to my parents for supporting me during a hard time in my life. Finally, I would like to dedicate this work to my wonderful wife, Camilla Frantzén, for her unwavering support and understanding during the research study and for her positive approach to life.

Marcus Frantzén
Skövde 2013

Declaration

I declare that this research thesis report is my own work and every effort has been made to clearly point out the contributions and materials from others, by providing references and acknowledgements. I declare that the work was solely conducted during the registration for the degree of PhD and that it has not previously been submitted for any other academic award. I further declare that I obtained the necessary authorisation and permission to carry out this research.

Table of Contents

Abstract i

Acknowledgements iii

Declaration iv

Table of Contents v

List of Figures xii

List of Tables xiv

List of Acronyms xvi

Chapter 1 1

1 Introduction 1

1.1 Research background 1

1.1.1 Challenges of manufacturing industry 1

1.1.2 State of the art: a brief overview 3

1.2 Aim and objectives 6

1.3 Research methodology 7

1.3.1 Qualitative, quantitative, and multi-method research 7

1.3.2 Different research strategies 8

1.4 Scope 11

1.5 Thesis organisation 12

Chapter 2 13

2 Hybrid Flow Shop Scheduling Problems and Scheduling Methodologies 13

2.1 The production scheduling problem 13

2.2 Categories of scheduling problems 13

2.3 Hybrid flow shops 15

2.3.1 Description of hybrid flow shop scheduling problem notation: 15

2.3.2 Machine characteristics 16

2.3.3 Job constraints 17

2.3.4 Objective function 19

2.4 Scheduling methodologies 20

2.4.1 Optimisation-based approaches 21

2.4.2 Artificial intelligence-based approaches 22

2.4.3 Genetic algorithms 24

2.4.4 Dispatching rules approaches 29

2.4.5 Simulation-based approaches 31

2.4.6 GA or dispatching rules for simulation 33

2.5 Assumptions usually made in scheduling research 35

2.6 Scheduling objectives in real-world problems 37

2.7 Concluding remarks 38

Chapter 3 40

3 Rescheduling and System Support 40

3.1 Uncertainty and rescheduling 40

3.1.1 Rescheduling environments 42

3.1.2 Completely reactive approaches 43

3.1.3 Predictive-reactive rescheduling policies 43

3.1.4 Rescheduling methods 44

3.1.5 Direct, indirect and hybrid representation of schedules 45

3.2 The scheduling task 47

3.2.1 Functions of the production scheduling task 47

3.3 System support 48

3.3.1 User-interfaces and human control 49

3.3.2 Flexible objectives over the horizon 51

3.3.3 Feasibility check and fault control 51

3.3.4 Evaluating scheduling systems 51

3.3.5 Commercial software and real-world case studies 53

3.3.6 Proposed architectures 57

3.4 Identified functions to include in a system architecture 60

3.5 Concluding remarks 61

Chapter 4 63

4 A Web Services-based Architecture for Industrial Scheduling 63

4.1 Web-based simulation 63

4.2 Existing Web-based systems for SBO 65

4.3 OPTIMISE: A web services-based SBO platform 66

4.3.1 Optimisation manager and database 70

4.3.2 Simulation components 71

4.4 OPTIMISE Scheduling System (OSS) 73

4.4.1 Scheduling functions and features included in OSS 75

4.4.2 OPTIMISE Information System 78

4.4.3 Scheduling operational steps and system communication 81

4.4.4 Rescheduling procedure 84

4.5 Concluding remarks 86

Chapter 5 87

5 Hybrid Genetic Representations for Industrial Scheduling 87

5.1 Predictive and realised schedule 87

5.1.1 Indirect predictive schedule 88

5.1.2 Direct predictive schedule 89

5.1.3 Predictive schedule and sequence dependent setup times 90

5.1.4 Schedule representation for predictive and realised schedules 91

5.1.5 Permutation versus non-permutation schedules 92
5.2 Optimisation algorithm representation 93
5.2.1 Hybrid genetic representation 94
5.2.2 Scheduling approaches in the GA 95
5.3 Optimisation algorithm steps 98
5.4 Optimisation algorithm operators 101
5.4.1 Initialisation 101
5.4.2 Job sequence crossover 102
5.4.3 Job allocation crossover 102
5.4.4 Dispatching rule crossover 102
5.4.5 Job allocation mutation 102
5.4.6 Dispatching rule mutation 102
5.4.7 Sequence mutations 103
5.5 Optimisation methods 108
5.6 Concluding remarks 109
Chapter 6 110
6 A Full-Scale Industrial Case Study 110
6.1 The industrial case study 110
6.1.1 The scheduling problem 111
6.1.2 The real-world production system 112
6.1.3 The simulation model 114
6.2 The OSS implementation and integration overview 117
6.2.1 Input data module 119
6.2.2 Shop floor module 119
6.2.3 Predictive- and dispatching algorithm library 119
6.3 Scheduling user-interfaces 120

6.3.1 Input data client 121
6.3.2 Scheduling client user-interface 123
6.4 Predictive schedule user-interface 127
6.4.1 Optimisation progress 129
6.4.2 Gantt chart 129
6.4.3 Stock levels 130
6.4.4 Export input and output data 130
6.5 Realised schedule user-interface 130
6.5.1 Schedule execution status 130
6.5.2 Line status 131
6.6 Dispatcher client user-interface 134
6.7 Scheduling scenarios 137
6.8 Real-world deployment stages 140
6.9 Stage 1: simulation model validation 141
6.10 Stage 2: line schedule validation 142
6.11 Stage 3: workstation schedule validation 143
6.12 Stage 4: rescheduling validation 145
6.13 Stage 5: PDA program validation 150
6.14 Stage 6: real-world validation 152
6.15 Concluding remarks 153
Chapter 7 154
7 Quantitative Results and Analyses 154
7.1 General settings 154
7.1.1 Weighted-sum objective function 154
7.1.2 Scheduling scenarios 158
7.1.3 General optimisation settings 159

7.2 Results with setup-reduction mutation ... 159
7.2.1 Analysis ... 161
7.3 Results of the predictive scheduling ... 162
7.3.1 Bottleneck analysis ... 162
7.3.2 Different optimisation methods ... 165
7.3.3 Results of different optimisation methods ... 166
7.3.4 Dispatching rule frequency ... 168
7.3.5 Performance analysis ... 172
7.4 Results with reactive scheduling ... 174
7.4.1 Experimental settings ... 174
7.4.2 Indirect approach ... 177
7.4.3 Direct approach ... 177
7.4.4 Hybrid approach ... 178
7.4.5 Comparison of the optimisation methods ... 179
7.5 Comparison of predictive and reactive scheduling results ... 180
7.6 Discussions ... 181
7.7 Concluding remarks ... 181
Chapter 8 ... 183
8 Conclusions and Future Work ... 183
8.1 Conclusions ... 183
8.2 Contributions to knowledge ... 186
8.3 Future work ... 189
8.3.1 Benchmark problems ... 189
8.3.2 Schedule robustness and hybrid methods ... 189
8.3.3 Integration ... 190
References ... 191

Appendix A: Carrier Flag Client 208

Appendix B: Dispatching Client User-interface 209

Appendix C: Predictive Scheduling Performance Results Scenario 1 217

Appendix D: Predictive Scheduling Performance Results Scenario 2 220

Appendix E: Reactive Scheduling Replication Results 223

Appendix F: Predictive and Reactive Scheduling Hypothesis Tests 225

Appendix G: Simulation model data 231

Appendix H: Simulation model in C# 243

Appendix I: Genetic Algorithm 247

Appendix J: List of Publications 248

List of Figures

Figure 1.1 Research methodology adopted. 10
Figure 2.1 linear order crossover. 27
Figure 3.1 Extended modular architecture (Framinan, Ruiz 2010) 57
Figure 4.1 Web-based distributed simulation system. 65
Figure 4.2 Web-based SBO framework. 66
Figure 4.3 With XML Web services, OPTIMISE can be deployed with high flexibility and scalability. 68
Figure 4.4 The generic OPTIMISE system architecture. 69
Figure 4.5 Architecture of OPTIMISE Scheduling System (see also Figure 4.6 for the information exchanges between the modules). 73
Figure 4.6 Operational steps and system communication (see also Figure 4.5) 82
Figure 4.7 Rescheduling flowcart. 85
Figure 5.1 Gantt chart of a two stage production using FCFS. 89
Figure 5.2 Gantt chart of a two stage production using direct schedule. 90
Figure 5.3 Gantt chart with setup times and FCFS. 91
Figure 5.4 Gantt chart with setup times and direct schedule. 91
Figure 5.5 The hybrid genetic representation. 94
Figure 5.6 Direct, indirect, and hybrid approach. 96
Figure 5.7 Direct representation of an operation schedule. 96
Figure 5.8 Indirect representation of an operation schedule. 98
Figure 5.9 A hybrid of direct and indirect representation. 98
Figure 5.10 Flow chart of the optimisation algorithm steps. 99
Figure 5.11 Reproduction in the optimisation algorithm. 100
Figure 5.12 Mutation into new solution. 100
Figure 5.13 Variant group sequence mutation. 104
Figure 5.14 Setup group sequence mutation. 106
Figure 5.15 Optimisation methods. 108
Figure 6.1 Production stages. 116
Figure 6.2 Structure of a production stage. 116
Figure 6.3 OSS implementation. 117

Figure 6.4 OSS distributed on different servers. 118
Figure 6.5 Screenshot of a schedule in Excel. 122
Figure 6.6 Calendar client. 123
Figure 6.7 The scheduling client. 124
Figure 6.8 Optimisation algorithm settings. 125
Figure 6.9 Advanced optimisation client. 127
Figure 6.10 Optimisation progress. 128
Figure 6.11 Schedule execution status. 131
Figure 6.12 Line status program. 132
Figure 6.13 Personal digital assistant. 135
Figure 6.14 Expert suggestion in the PDA. 136
Figure 6.15 Scheduling horizon and periodic rescheduling. 137
Figure 6.16 The rescheduling point. 139
Figure 6.17 The new predictive schedule. 139
Figure 6.18 The realised schedule. 140
Figure 6.19 Real-world deployment stages. 141
Figure 6.20 Validation of a mock run. 142
Figure 6.21 Comparison of results from OSS with actual output from the real line. 143
Figure 6.22 A workstation schedule. 144
Figure 6.23 The procedure of the simulation environment rescheduling test. 148
Figure 6.24 One production stage in a PDA-program validation test. 151
Figure 7.1 Simulation-based optimisation progress. 157
Figure 7.2 Active duration of scenario 1. 162
Figure 7.3 Shifting bottleneck analysis of scenario 1. 163
Figure 7.4 Active duration of scenario 2. 164
Figure 7.5 Shifting bottleneck analysis of scenario 2. 165
Figure 7.6 Algorithm performance in scenario 1. 172
Figure 7.7 Algorithm performance in scenario 2. 173
Figure A. 1 Carrier flag program. 208
Figure B. 1 Start screen. 209
Figure B. 2 Start a job screen. 210
Figure B. 3 Confirm screen. 210

Figure B. 4 Move a job. 211
Figure B. 5 Confirm screen. 212
Figure B. 6 Block parts. 213
Figure B. 7 Warning when trying to break soft constraints. 214
Figure B. 8 Robustness and PDA-status. 214
Figure B. 9 Stocktaking menus. 215
Figure B. 10 Information menu. 216
Figure C. 1 FHYB performance on scenario 1. 217
Figure C. 2 HYB performance on scenario 1. 218
Figure C. 3 PDR performance on scenario 1. 218
Figure C. 4 PS performance on scenario 1. 219
Figure C. 5 NPS performance on scenario 1. 219
Figure D. 1 FHYB performance on scenario 2. 220
Figure D. 2 HYB performance on scenario 2. 221
Figure D. 3 PDR performance on scenario 2. 221
Figure D. 4 PS performance on scenario 2. 222
Figure D. 5 NPS performance on scenario 2. 222

List of Tables

Table 3.1 The main areas of modular scheduling system architecture 60
Table 5.1 Predictive and realised schedules. 92
Table 5.2 Example of a permutation schedule. 93
Table 5.3 Example of a non-permutation schedule. 93
Table 6.1 Product variants groups 113
Table 6.2 Simulation environment test. 149
Table 7.1 Setup reduction for scenario 1. 160
Table 7.2 Setup reduction for scenario 2. 161
Table 7.3 Results of scenario 1. 166
Table 7.4 Results of scenario 2. 166
Table 7.5 Dispatching rules frequency PDR scenario 1. 169
Table 7.6 Dispatching rules frequency PDR scenario 2. 170

Table 7.7 Dispatching rules frequency HYB scenario 1. 170
Table 7.8 Dispatching rules frequency HYB scenario 2. 171
Table 7.9 Dispatching rules frequency FHYB scenario 1 171
Table 7.10 Dispatching rules frequency FHYB scenario 2 172
Table 7.11 Results of real-world compared to PDR. 177
Table 7.12 Results of real-world compared to PS. 177
Table 7.13 Results of real-world compared to FHYB. 178
Table 7.14 Comparison of different optimisation methods. 179
Table E. 1 Reactive scheduling results for the objective missed target levels. 223
Table E. 2 Reactive scheduling results for the objective makespan. 223
Table E. 3 Reactive scheduling results for the objective shortage hours. 224
Table E. 4 Reactive scheduling results for the objective setup time. 224
Table F. 1 PS setup reduction hypothesis test for scenario 1. 226
Table F. 2 NPS setup reduction hypothesis test for scenario 1. 226
Table F. 3 PS setup reduction hypothesis test for scenario 2. 227
Table F. 4 NPS setup reduction hypothesis test for scenario 2. 227
Table F. 5 Predictive scheduling hypothesis test for scenario 1. 228
Table F. 6 Predictive scheduling hypothesis test for scenario 1. 228
Table F. 7 Predictive scheduling hypothesis test for scenario 2. 229
Table F. 8 Reactive scheduling hypothesis test for the objective setup time. 229
Table F. 9 Reactive scheduling hypothesis test for the objective makespan. 230
Table G. 1 Product variant information 231
Table G. 2 Product variants production stages 231
Table G. 3 Machine settings 232
Table G. 4 Availability of machines 233
Table G. 5 Processing times (seconds) of product variants at different machines 234
Table G. 6 Sequence-dependent setup time production stage 1 235
Table G. 7 Sequence-dependent setup time production stage 5 235
Table G. 8 Machine-dependent setup times 237
Table G. 9 Work in process 238
Table G. 10 Target levels for WIP 238
Table G. 11 Physical maximum of the FGI 239

Table G. 12 Number of parts in FGI at start of simulation 239
Table G. 13 Target levels of the finished goods inventory 239
Table G. 14 Demands Scenario 1 240
Table G. 15 Demands Scenario 2 241
Table G. 16 Production stage buffers 242
Table G. 17 Input and output buffers at the machines (number of carriers) 242
Table G. 18 Number of carriers 242

List of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Networks
CDR	Composite Dispatching Rule
CSMQ	Company Specific Message Queuing
DES	Discrete Event Simulation
EDD	Earliest Due Date
ERP	Enterprise Resource Planning
FCFS	First Come First Served
FGI	Finished Goods Inventory
FHYB	Free HYBrid
GA	Genetic Algorithm
GP	Genetic Programming
GUI	Graphical User Interface
HFS	Hybrid Flow Shop
HNBS	Hybrid Non-Blocking Sequence
HSNBS	Hybrid Setup Non-Blocking Sequence
HYB	HYBrid
IT	Information Technology
LOX	Linear Order Crossover
LPT	Longest Processing Time
MSMQ	MicroSoft Message Queuing
NPS	Non-Permutation Schedule

OCBA	Optimal Computing Budget Allocation
OIS	OPTIMISE Information System
OptDB	Optimisation DataBase
OptEngine	Optimisation Engine
OPTIMISE	OPTIMisation using Intelligent Simulation and Experimentation
OSP	Optimisation Service Provider
OSS	OPTIMISE Scheduling System
OX	Order Crossover
PDA	Personal Digital Assistant
PDR	Priority Dispatching Rule
PMX	Partially Mapped Crossover
PS	Permutation Schedule
SA	Simulated Annealing
SBO	Simulation-Based Optimisation
SEQ	SEQuence
SME	Small to Medium Enterprise
SPT	Shortest Processing Time
SQL	Structured Query Language
TCP-IP	Transmission Control Protocol/Internet Protocol
TS	Tabu Search
TSP	Travelling Salesman Problem
VBA	Visual Basic for Applications
VPN	Virtual Private Network
VSD	Variant Setup Deadline
WBS	Web-Based Simulation
WIP	Work In Process
XML	Extensible Markup Language

Chapter 1

1 Introduction

The introductory chapter presents the research background (Section 1.1) that motivated this research study and thereby the aim and objectives (Section 1.2). The research methodology (Section 1.3) used is explained, the scope of the work (Section 1.4) is then defined and followed by the organisation of the whole thesis (Section 1.5).

1.1 Research background

1.1.1 Challenges of manufacturing industry

Manufacturing organisations are experiencing shortened product life cycles, unpredictable customer demands, and fluctuating production volumes. At the same time, the level of global competition is becoming much stronger. All these changes are forcing manufacturing companies to react more flexibly and swiftly to changes in both product diversity and production volume. In order to meet these challenges, the shop floor control system of a manufacturing system has to be designed to incorporate a high degree of flexibility. Groover (2001) defines different types of flexibility in manufacturing systems as follows:

- *Machine flexibility* means the ability to adapt machines to different production operations and parts.
- *Production flexibility* means the range of different parts that can be produced by the system.
- *Mix Flexibility* means the system's ability to maintain production volume despite a change of product mix.
- *Product flexibility* means the system's ability to cope with design changes and introduction of new products.
- *Routing flexibility* means the system's ability to continue production through alternative workstations if machines are subject to interruptions.
- *Volume flexibility* means the system's ability to economically produce parts of high and low volumes.
- *Expansion flexibility* means the ability for a system to expand for a higher production volume.

In terms of the challenges faced by any manufacturing company, there is no exception for high quantity production (*mass production*) such as that of car manufacturers. In order for them to survive in an ever-changing market, they need to be highly competitive by achieving near optimal efficiency in their operations. However, with the demand for much more flexibility to cope with greater product variety and fluctuating production volumes, as mentioned above, industrial manufacturing systems, in general, and car manufacturers, in particular, are becoming much more complex, viewed from both a technological and management perspective.

In general, scheduling concerns "*the allocation of resources over time to perform a collection of tasks*" (Baker and Trietsch, 1974). In practice, scheduling refers to "*the determination of a set of orders, which will be processed by the resources during a short-term period (day, week, etc.)*" (Kiran, 1998). For a manufacturing company to remain highly competitive, a near optimal allocation of their resources is essential. Furthermore, scheduling may also contribute to the flexibility of a firm (De Snoo et al., 2011). It is therefore not difficult to recognise that efficient scheduling is vital to the success of manufacturing systems in industry today. This makes scheduling an interesting area that has drawn much attention from both academic researchers and industrial practitioners. Nevertheless, with the demand for higher flexibility, the efficient scheduling of a production line has become an extremely difficult task, especially when day-to-day challenges, such as product or order changes, have to be handled efficiently. On a modern manufacturing shop floor, scheduling tasks are undertaken by the Enterprise Resource Planning (ERP) system. Unfortunately, the existing scheduling modules developed for an ERP system are based on deterministic algorithms which are only suitable for operations in a predictable and stable environment. This implies that ERP systems in general do not have the capability to generate detailed schedules for a complex manufacturing system. Therefore, a scheduling decision support that can cope with real-world industrial production systems is needed. Consequently, it is necessary for the research community to explore some new approaches that can make shop floor scheduling tasks capable of handling the complexity and flexibility demands facing today's manufacturing companies.

1.1.2 State of the art: a brief overview

The scheduling of a real-world production line may be highly complex; sequence dependent setup times, constraints, and long failures could affect the possibility of reaching the production target. Many real-world scheduling problems belong to the class of NP-complete problems, for which finding the optimal solution within an acceptable time period is impossible, due to the size of the problems (Garey and Johnson, 1979). To prove that an optimisation problem is as difficult as an NP-complete problem, the term NP-hard is useful, because it describes that it is not possible to find the optimal solution with available techniques (Baker and Trietsch, 2009). The same could be said about scheduling problems with increasing complexity. Trying to compare all scheduling problems would not be feasible, simply because the combinations of scheduling problems are huge. There are too many different sizes, constraints and objectives in order to solve them optimally, which on the other hand can be done for smaller scheduling problems. At the same time, trying to simplify complex scheduling problems by reducing the number of constraints and characteristics would simply transform them into unrealistic textbook problems that may not be acceptable in a real-world scheduling situation. This claim can be supported by many other researchers. For example, Pinedo (2008) states that advances in scheduling theory have only had a limited impact on scheduling in practice, although the theoretical research has not been a complete waste of time, because it has given insights into the scheduling problem in general. Gupta and Stafford (2006) also claim that theoretical flow shop scheduling problems remain largely unsolved, when the 50 years of research is considered. They state that research within flow shop scheduling seems to have been motivated by what the researchers can achieve rather than what is important, and thereby also suffers from too much abstraction and too little application. Future research in flow shop scheduling should address real-world problems (Jahangirian et al., 2010), in order to avoid spending decades only trying to solve textbook problems. Even though most real-life situations are better represented by models with uniform or unrelated machines, most research has been done on flow shops with identical machines, which is probably due to the fact that identical machines are easier to handle (Ribas et al., 2010). According to

Ribas et al., (2010), most research has been carried out with at most one constraint (e.g., setups, failures, blocking) at a time being studied and only a few researchers have studied all or most of them at the same time. Consequently, in order to diminish the gap between theory and real-world scheduling problems, several constraints need to be considered.

Simulation modelling, i.e., discrete event simulation (DES), has the capability to represent complex real-world systems and their constraints in detail. Simulation-based scheduling approaches are derived from dispatching rule-based methods. In a simulation-based approach, several dispatching rules might be used at different stages, in order to make a decision (Kiran, 1998). Basically, a dispatching rule is a rule of thumb that gives priority to a job among other ones at a specific stage, i.e., at a machine. This is why dispatching rules can also be called priority dispatching rules (PDRs). Generally, a PDR-based approach does not try to find an optimal schedule, but relies on knowing that one scheduling rule statistically performs better than another one, which is sufficient. In comparison, using a meta-heuristic optimiser, such as a Genetic Algorithm (GA), to generate the near optimal schedules directly, may be advantageous if searching for “optimal” solutions is desired. There are many studies that compare these two approaches and some of them provide results showing that the use of GAs to generate detailed schedules can obtain better solutions (Sankar et al., 2003; Kim et. al., 2007) than those obtained by using PDRs. On the other hand, using a GA to select PDRs has shown promising results (Tanev et. al., 2004; Ochoa et al., 2009) compared to conventional GA approaches. Furthermore, hybrids that have a combined representation of these two approaches have shown good results, when uncertainty is considered (Roundy et.al., 1991; Barua et.al., 2005). Robust scheduling (e.g., Leon et al., 1994), reactive scheduling, or rescheduling (Church and Uzsoy, 1992) are also some methodologies that have been successfully used to address scheduling problems with regard to uncertainty.

Regarding uncertainty, McKay and Wiers (1999) claim that researchers and real-world schedulers do not discuss the same problem. While researchers are solving deterministic sequencing problems, real-world schedulers are faced with day-to-day challenges in which uncertainty is believed to be the key characteristic. McKay and Wiers (1999) define a scheduling task as: "*a dynamic and adaptive process of iterative decision making and problem solving, involving information acquisition from a number of sources, and with the decisions affecting a number of production facets in reaction to immediate or anticipated problems*". Wiers (1997) defines production scheduling as a task and the following four types of control are used to further characterise the task: Detailed control, Direct control, Restricted control, and Sustained control. These controls generally mean that the scheduling task deals with short-term decisions regarding what to do next and the situation at hand, answering questions and giving directions. Furthermore, the scheduler monitors schedule execution and carries out necessary changes when needed, in order to fulfil scheduling targets. It is also important to generate a valid schedule, since there is no intermediate control before launching the schedule and there is a risk that the schedules will be adjusted manually (Stoop and Wiers, 1996). Pinedo (2005) also addresses that "*Analyzing a planning or scheduling problem and developing a procedure for dealing with it on a regular basis is, in the real world, only part of the story. The procedure has to be embedded in a system that enables the decision-maker to actually use it. The system has to be integrated into the information system of the organization, which can be a formidable task*". Jahangirian et al., (2010) show that even though scheduling applications have been the most common ones among simulation applications in manufacturing and business between 1997 and 2006, only a small portion of them use both real problems and real data. They also point out that papers addressing real-world problems are important to future research. According to the review of hybrid flow shops by Ribas et al., (2010), only two papers use on-line algorithms for real-time scheduling, when simulation with dispatching rules or realistic decision support systems is considered, and indicate this as an interesting area for future research.

It is not only the scheduling problem that needs to be considered, but also the scheduling task and its integration in the organisation. A real-time scheduling system is not only needed to support the work of the production scheduler, but also the operators on the shop floor, by re-generating feasible schedules when required. With a real-time rescheduling capability, the proposed scheduling system not only solves the sequencing problems, but also provides decision-making support on a day-to-day basis when disturbances, such as machine breakdowns, happen. Based on this research background, the need of a real-time shop floor scheduling system capable of handling the complexity and uncertainty found in real-world problems, when generating near optimal schedules, as well as interacting with users, such as production schedulers and shop floor operators, is identified as the target of this study.

1.2 Aim and objectives

The overall aim of this study is the advancement of research in manufacturing scheduling through the exploration of more effective approaches to address complex, real-world manufacturing flow shop problems. The research hypothesis behind this aim is that existing scheduling approaches and algorithms are believed to be inadequate to address complex, real-world manufacturing flow shop problems because they lack the real-time and reactive support to tackle uncertainty. Therefore, in order to advance the research of manufacturing scheduling, a combination of systems engineering and algorithmic design is needed to tackle the uncertainty issues in real-world environment. The aim of this thesis can be further refined into the following specific objectives:

- Appraise the existing research knowledge and industrial practice to establish the understanding of manufacturing systems and explore the requirements of the scheduling in real-world complex hybrid flow shops.
- Based on the comprehensive literature review, investigate how simulation tools and scheduling techniques can be enhanced to cope with uncertainty, and flexibly cope with different scheduling approaches in order to enhance their performance.
- To design a system framework with real-time and reactive support and then evaluate this framework qualitatively using a real-world industrial case study.
- Design and propose a hybrid meta-heuristic scheduling algorithm for simulation-based optimisation that can flexibly cope with different scheduling approaches in order to be more adaptive to tackle complex hybrid flow shop scheduling problems.

- Validate the performance of the algorithm using empirical experiments based on real-world shop-floor data collected through the system framework implemented in earlier stages.

1.3 Research methodology

Real-world research generally refers to applied research which typically uses projects that are small in scope and scale. Compared to academic research, where the focus is on advancing an academic discipline, real-world research focuses on problems with direct relevance to people or the environment, such as child care and climate change. The real-world researcher needs well-developed social skills and almost always works in the field, e.g., industry, compared to the academic researcher that mainly uses laboratories of some kind. (Robson, 2011)

1.3.1 Qualitative, quantitative, and multi-method research

According to Robson (2011), research can be divided into two main groups: qualitative or quantitative. Whilst quantitative research makes use of numerical data, qualitative data is typically non-numerical (e.g., in the form of words). Myers (1997) defines qualitative research as research that *"involves the use of qualitative data, such as interviews, documents, and participant observation, to understand and explain social phenomena"*. Jabar et al., (2009) argue that qualitative research is significant for information systems research because of its ability to explain what is going on in a real organisation. Quantitative research was, on the other hand, first developed to study natural phenomena in natural sciences (Jabar et al., 2009). Quantitative research involves the collection of quantitative data, the design of which typically used is to exactly determine at an early stage how to carry out the research project before the data is accumulated (Robson, 2011). According to Reswick (1994), the researcher can isolate a problem, e.g., using a laboratory, and can therefore with precision and accuracy define and measure input and output parameters of the study. However, multi-strategy designs have received increased interest because they produce a substantial collection of both qualitative and quantitative data in different parts of a research project. (Robson, 2011)

1.3.2 Different research strategies

Depending on the form and context of the research question, as well as control over behavioural events and focus on contemporary ones, the research strategy used will differ. Yin (2003) defines different types of research strategies:

Table 1.1 Relevant situations for different research strategies (Yin, 2003)

Strategy	Form of research question	Requires control of behavioural events?	Focuses on contemporary events?
Experiment	How, why?	Yes	Yes
Survey	Who, what, where, how many, how much?	No	Yes
Archival analysis	Who, what, where, how many, how much?	No	Yes/No
History	How, why?	No	No
Case study	How, why?	No	Yes

The “who” and “where” questions are common in survey or archival analysis in which the research goal is to be predictive about specific outcomes or when the prevalence of a phenomenon needs to be described. The “what” question is also appropriate in survey or archival analysis which, for example, may provide the answer to the outcomes of a specific type of managerial restructuring. The “how” and “why” questions are typically more explanatory and used for the research strategies: case studies, experiment, and history. In general, the history research strategy is used when no living persons of relevance can report afterwards and therefore historical data needs to be applied. However, the case study strategy can be used when contemporary events need to be examined. In addition to the historical data method, the case study strategy includes the possibilities of interviews with people involved and direct observation of the events being studied. The experiment research strategy is carried out when the researcher can control behavioural events, i.e., can manipulate them directly, precisely, and systematically. (Yin, 2003)

A multi-method research strategy is one that combines different research methods (qualitative and quantitative), in order to provide a greater understanding of the phenomenon of interest and to increase the confidence in the conclusions generated by the research study (Johnson et al., 2007). This can also be referred to as triangulation, for resolving the inherent biases of one measurement technique (Denzin, 2009). Denzin (2009) divides triangulation into four basic categories:

- Data triangulation means using more than one type of data collection method. Different sources can be used to collect the data (observation, interviews or documents), and the data can be collected at different times and different places.
- Investigator triangulation means using multiple observers rather than single observers. For example, different interviewers or data analysts can be used in the study to remove the potential bias connected to one person.
- Theoretical triangulation means using multiple perspectives (theories) on a set of objects rather than a single perspective.
- Methodological triangulation means within-method triangulation or between-method triangulation.

Denzin (2009) suggests that using between-class triangulation, i.e., different methods and measurement strategies, is preferred in comparison to within-class triangulation, in which there are variations of one and the same measurement technique.

A wide range of research methods may be appropriate for systems engineering because it is an interdisciplinary and broad field of engineering dealing with complex projects (Ferris, 2009). According to Yin (2003), one reason why a case-based research approach is appropriate is when contextual conditions must be covered because they are believed to be relevant to the phenomenon of study, which is something that can be characterised with qualitative research. At the same time, a case study may be part of a multi-method research study (Yin, 2003). Consequently, the research approach adopted is a multi-method research strategy in which both theories of current research, experiments (quantitative) and case study (qualitative) are used to achieve the research objectives. Data was collected from three different sources:

- Literature review of existing research to establish the understanding of manufacturing systems and to explore the requirements of the scheduling in real-world complex hybrid flow shops (Chapters 2, 3 and the beginning of Chapter 4).
- Evaluation of the proposed system framework using a real-world industrial case study. Chapter 6 begins with the motivation and selection of industrial case study, and then continues with the implementation and evaluation of the system (proposed in Chapter 4).
- Data generation using simulation-based optimisation with a discrete-event simulation model to investigate how the hybrid meta-heuristic scheduling algorithm (proposed in Chapter 5) can flexibly cope with different scheduling approaches, in order to be more adaptive in tackling complex hybrid flow shop scheduling problems (Chapter 7).

Figure 1.1 provides an overview of the research methodology used to realise the research objectives of the whole study.

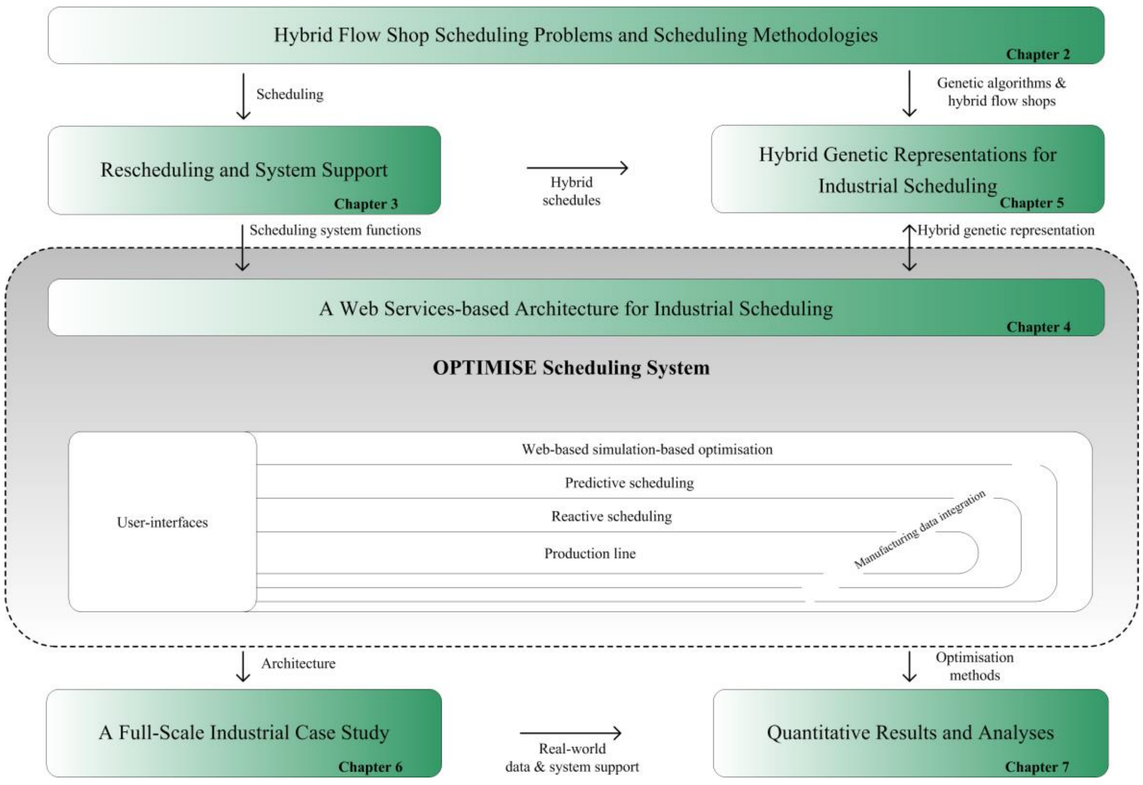

Hybrid Flow Shop Scheduling Problems and Scheduling Methodologies
Chapter 2
Scheduling
Genetic algorithms & hybrid flow shops
Rescheduling and System Support
Chapter 3
Hybrid schedules
Hybrid Genetic Representations for Industrial Scheduling
Chapter 5
Scheduling system functions
Hybrid genetic representation
A Web Services-based Architecture for Industrial Scheduling
Chapter 4
OPTIMISE Scheduling System
User-interfaces
Web-based simulation-based optimisation
Predictive scheduling
Reactive scheduling
Production line
Manufacturing data integration
Architecture
Optimisation methods
A Full-Scale Industrial Case Study
Chapter 6
Real-world data & system support
Quantitative Results and Analyses
Chapter 7

Figure 1.1 Research methodology adopted.

1.4 Scope

Flow shops are generally the type of production lines used for mass production in industry (Groover, 2000). In the classical definition of flow shop problems (Baker, 1974), each production stage consists of only one resource, e.g., machine, and there are at least two production stages. All jobs need to go through the production stages in the same machine order.

In industry, many companies need to increase their production capacity or balance the capacity between different production stages. Companies may also need to manufacture new products, which could mean that the new products are produced using the same machines in most stages but require new ones in others. Consequently, and for other reasons, a flow shop with parallel machines is formed, commonly also known as a hybrid flow shop (Ribas, et al., 2010).

Since “a dear child has many names”, the same scheduling problem could be specified by a number of definitions, e.g., flow shop with multiple machines, flexible flow shop, multiprocessor flow shop, or modified flow shop. However, the hybrid flow shop notation proposed in Ribas et al., (2010) is good for defining real-world scheduling problems, since it handles a broad range of flow shop scheduling problems. A hybrid flow shop consists of at least two production stages and at least one of these stages includes more than one machine (Gupta, 1988).

The scope of this thesis is therefore to address the multi-stage (more than three stages) hybrid flow shops with unrelated parallel machines for discrete parts manufacture, because most real-world flow shops in industry consist of several production stages. Furthermore, in order to diminish the gap between theory and real-world scheduling problems and not make unrealistic assumptions, several constraints and multiple scheduling objectives are addressed as well. Consequently, a review of flow shop scheduling problems and different scheduling methodologies is made. However, the review of scheduling methodologies is not limited to hybrid flow shops, since much scheduling research has been conducted on other complex scheduling problems, e.g., job shops, which could in fact be useful for hybrid flow shops as well. Nevertheless, in the design of the hybrid genetic representation, the focus is only put on hybrid flow shops. As mentioned, because uncertainty is a key characteristic in real-world scheduling, it needs to be addressed in order to realise a schedule in a production line. However, uncertainty is only one part of the scheduling task in which day-to-day challenges need to be handled by production schedulers. Consequently, methodologies that handle uncertainty as well as scheduling system functions and features, in order to support the tasks of the scheduler, are also studied.

1.5 Thesis organisation

Chapters 2, 3, and parts of Chapter 4 feature the literature review. In short, Chapter 2 describes the background of scheduling theory and scheduling methodologies. Chapter 3 reviews rescheduling and identifies the main functions and features to be included in a system to support the scheduling task. Chapter 4 begins with a brief introduction and literature review of Web-based simulation and some existing platforms found in the literature. Furthermore, this chapter describes the overall system architecture of the web services-based industrial scheduling system, i.e., OPTIMISE Scheduling System (OSS), which is designed to be software architecture to solve the limitations of existing scheduling software used in industry. Chapter 5 describes a new novel hybrid genetic representation which is based on a mixture of dispatching rules and genetic encoding the entire schedule. The design and implementation of the hybrid genetic representation into an SBO algorithm for handling various real-world, complex hybrid flow shop scheduling problems is then addressed in detail. In order to prove the system architecture, optimisation methods and techniques proposed in this thesis, a full-scale industrial case study of a machining line was completed in this study and is presented in Chapter 6. Chapter 7 presents the experimental results of applying the hybrid genetic representation to the real-world case study. All the results in this chapter were obtained from the OSS implementation on the real machining line. Finally, the thesis conclusions, contributions to knowledge, and identified future research areas are presented in Chapter 8.

Chapter 2

2 Hybrid Flow Shop Scheduling Problems and Scheduling Methodologies

This chapter describes the background of scheduling theory, classification of scheduling problems, and what kinds of assumptions are usually made in scheduling research. Furthermore, different approaches to solve scheduling problems and new advances to solve complex hybrid flow shop scheduling problems are reviewed. Finally, the review is concluded and recommendations regarding how to solve real-world, complex hybrid flow shop scheduling problems are proposed.

2.1 The production scheduling problem

There are many different definitions of scheduling problems from the research communities and still they may differ from the understanding of scheduling problems faced daily in industry. The classical definition is more limited to “sequencing”, which can be found in Conway et al., (1967), who define sequencing in terms of one machine and scheduling as the sequencing of operations on several machines. In general, scheduling concerns “*the allocation of resources over time to perform a collection of tasks*” (Baker and Trietsch, 1974). In practice, scheduling refers to “*the determination of a set of orders, which will be processed by the resources during a short-term period (day, week, etc.)*” (Kiran, 1998).

2.2 Categories of scheduling problems

Graves (1981) introduced a broad classification that covers the general characteristics of both scheduling theory and scheduling practice. The classification divides production scheduling problems into the following three dimensions:

1. Requirements generation

2. Processing complexity
3. Scheduling criteria

The first dimension, requirements generation, means that a manufacturing facility can be either an open shop when items are produced to order or a closed shop when orders are filled from existing inventory. In an open shop, the scheduling is simply described as a sequencing problem in which open orders are sequenced at the production facility. In a closed shop, both the sequencing problem and the lot-sizing decisions connected to the inventory replenishment process need to be considered. The second dimension, processing complexity, refers to the number of production stages and type of flow and can be further classified into:

1. One stage, one processor
2. One stage, parallel processors
3. Multistage, flow shop
4. Multistage, job shop

In a one stage, one processor problem, all jobs require one production stage and only one single resource or machine needs to be scheduled. The one stage, parallel processor problem means that all jobs only require a single production stage, but there is more than one resource that can process the job. In the multistage, flow shop problem, all jobs require processing by the same set of resources and there is a common route for all jobs. The multistage, job shop problem means that there are no restrictions on the production stages for a job and alternative routes can be chosen for a job. The third dimension, scheduling criteria, describes the scheduling objectives. These include, to mention a few, to minimise tardiness, minimise work-in-process, maximise production rate, and maximise the utilisation level of resources, which are just some of the objectives commonly used in production scheduling problems.

According to Graves (1981), there are two additional dimensions that could have been included: the requirement specification and the scheduling environment. The requirement specification shows the degree of uncertainty of the scheduling problem which can be defined as deterministic or stochastic. Stochastic scheduling problems may include random variables such as distributions of processing times, failures, and so on. The scheduling environment defines whether the scheduling problem is static or dynamic. A static scheduling problem is when the number of jobs and their ready times are available, while a dynamic scheduling problem is when the number of jobs and related characteristics change over time.

2.3 Hybrid flow shops

Flow shops are generally the type of production lines used for mass production in industry (Groover, 2000). In the classical definition of flow shop problems, each production stage consists of only one resource, e.g., machine, and there are at least two production stages (Baker, 1974). All jobs need to go through the production stages in the same machine order.

In industry, many companies need to increase their production capacity or balance the capacity between different production stages. Companies may also need to manufacture new products, which could mean the new products are produced using the same machines in most stages but require new ones in others. Consequently, a flow shop with parallel machines is formed, commonly also known as a hybrid flow shop (Ribas, et al., 2010). As mentioned earlier in Chapter 1, there are some other names to describe a hybrid flow shop: e.g., flow shop with multiple machines, flexible flow shop, multiprocessor flow shop, or modified flow shop. In the remainder of this thesis, the term hybrid flow shop is continuously used, because its formal definition, as introduced in the next section, has captured the essence of the scheduling problems that can be found on real-world shop floors.

2.3.1 Description of hybrid flow shop scheduling problem notation:

A HFS (hybrid flow shop) consists of at least two production stages and at least one of these stages includes more than one machine, which has proven to be NP-complete, even for this basic HFS case (Gupta, 1988). In the structure $\alpha|\beta|\gamma$ proposed by Graham et al., (1979), α stands for the machine characteristics, β for the job constraints and γ for the objective considered. Ribas et al., (2010) refers to the specific notation proposed by Vignier et al., (1999) which follows the same structure proposed by Graham et al., (1979), but divides α into four terms, i.e., $\alpha_1\alpha_2\left(\alpha_3\alpha_4^{(1)},...,\alpha_3\alpha_4^{(\alpha_2)}\right)$. The first term α_1 specifies the problem considered, i.e., “HF” for a hybrid flow shop. The second term α_2 specifies the number of production stages, while the third term α_3 specifies the type of machines at a stage, i.e., identical (P), uniform (Q), unrelated (R), or one machine (0). Finally, the term α_4 specifies the number of machines at a stage. Furthermore, when there are several subsequent stages with the same type and number of machines, the terms α_3 and α_4 can be grouped as $\left(\left(\alpha_3\alpha_4\right)^l\right)_{l=s}^k$, where *s* stands for the first stage in the index and *k* for the last stage in the index (Ribas et al., 2010).

2.3.2 Machine characteristics

Identical parallel machines mean that all machines within each production stage are considered to be identical, and therefore the processing time of a job does not depend on which of the machines it is assigned to. According to Ribas et al., (2010), most research focuses on the hybrid flow shop problems with identical machines, e.g., Gupta et al., (1997) and Zhang et al., (2005) have studied the $HF2\left(PM^{(1)},0\right)$ problems, i.e., two-stage hybrid flow shop problems with several parallel identical machines in the first stage and one machine in the second. However, uniform or unrelated machines represent real-life situations in a better way. Uniform parallel machines mean that each machine within a production stage has its own speed and therefore has an individual completion time for a job. However, unrelated parallel machines mean that the processing times of a job on a production stage depend on each one of the parallel machines. Some of the machines might be better suited to some jobs whilst others are not, which may be due to physical differences in the machines, such as old machine equipment or newly bought machines. The reason for machine eligibility, i.e., when machines are dedicated to certain jobs, can be due to the technological differences between machines in the same stage or because some jobs have some special characteristics. On the other hand, it can also happen that a job can only be assigned to machines that are physically nearby. The last cause is still valid for defining a production stage with identical parallel machines, but if there are technological constraints, the production stage should be defined as a stage with unrelated parallel machines (Ribas, et al., 2010).

Ribas et al., (2010) also further categorise scheduling problems according to the specifics of the production system. For example, $HF2\left(RM^{(1)},0\right)| fmachs$, means a two stage hybrid flow shop with several unrelated parallel machines in the first stage, one machine in the second stage, and dedicated machines. Some constraints of production systems are:

- $fmachs$, represents jobs that at some or all stages are dedicated to specific machines (machine eligibility).
- nw, stands for "no wait" which means that the operations of a job have to be processed from the start to the end without any interruption on or between machines.
- brk, means that unavailability periods (failures) may happen in some or all machines in the production system.
- $size$, stands for multiprocessor task, which means that more than one machine is required in order to perform an operation at a certain stage.
- $blck$, stands for blocking and means that jobs may be blocked for transportation to the next production stage. Blocking can occur for several reasons, but downstream machine failures with limited buffer capacities are a common cause.

There is no agreed set of benchmark tests for the standard HFS, which makes it difficult to compare different algorithms (Ruiz and Vázquez-Rodríguez, 2010).

2.3.3 Job constraints

Job constraints can be classified as hybrid-specific or non-hybrid-specific. Hybrid-specific job constraints are those that are to be found exclusively in a hybrid flow shop.

The non-hybrid-specific job constraints are more general ones that can be found in any manufacturing environment. An example of a non-hybrid-specific job constraint is the setup time required to be able to process a job. If the setup time is machine dependent, i.e., the required time will depend on which machine it is assigned to, the job constraint is hybrid-specific. However, the differences between hybrid- and non-hybrid-specific job constraints can be quite hard to define, and it is argued in this thesis that the distinction between hybrid- and non-hybrid-specific job constraints has made no contribution to resolve complex scheduling problems. Therefore, hereafter, job constraints will not be distinguished as hybrid- or non-hybrid-specific, but simply as job constraints. Some common job constraints and characteristics follow:

- Job pre-emption: this means that a job currently being processed on a machine may be put on hold in its processing in favour of another job. When the job that had been put on hold continues, it need not restart its entire processing operation, but can continue where it left off. (Pinedo, 2008)
- Job precedence: is a predefined sequence or order of jobs that must be preserved. The reason for job precedence might be that certain sequences are prohibited due to technological constraints or because of a policy decision. An example of when job precedence rules are created is when there are long, sequence dependent setup times. (Conway, et al., 1967)
- Sequence dependent setup times: this means that a setup on a machine, in order to start a job, depends on the differences between the last and the current job (Pinedo, 2005).
- Transportation times: this means the time it takes to move a job between different locations (Pinedo, 2005).
- Missing operations (Ribas, et al., 2010), bypass (Pinedo, 2005), or by-passing move (Groover, 2000): all of these refer to the jobs which do not need to go through all production stages and can thus disregard some of them.
- Lot splitting means that a lot can be split over parallel machines in at least one production stage. If lot splitting is not allowed, it means that a lot cannot be started at the next production stage until the whole batch is finished in the current production stage.

- A lot sizing and scheduling problem is common in a closed shop and means that not only the sequencing problem is considered, but also the lot-sizing decisions associated with the inventory replenishment process (Graves, 1981).
- Re-entrant hybrid flow shop means that some jobs need to revisit some previous production stages.
- Rework means that some jobs might need to revisit a previous production stage because of quality problems.

2.3.4 Objective function

The objective function of a scheduling problem is what determines whether a schedule is good or bad. The definition of an objective function that represents the scheduling and production system goals is crucial in order to find the best schedules. The "minimax" criteria, simplified as "max", are frequently used in the literature to denote the time of the latest job to some criteria, e.g., the time of the latest job (T'Kindt et al., 2002). In the same manner, the "minisum" criterion $\overline{f}$ designates objectives based on all jobs, usually averages or sums of some kind (T'Kindt et al., 2002). Pinedo (2005) sorts objectives into three main groups: (1) throughput and makespan objectives; (2) due date related objectives, and (3) cost related objectives. In the throughput and makespan objectives, the following aims can be included:

$C_{\max}$ Maximum job completion time. The objective is to decrease the makespan, i.e., the time required for the last job to be finished. (T'Kindt et al., 2002)

$\overline{Th}$ Throughput rate. The objective is to increase throughput rate (average), e.g., throughput per hour. However, the throughput rate is usually unnecessary when decreasing the makespan, because maximizing $C_{\max}$ tends to increase the throughput rate (see Pinedo, 2005).

$\overline{C}$ Average completion time or total completion time of jobs. The objective is to decrease the average completion time of all jobs or the total completion time of jobs (T'Kindt et al., 2002).

In the due date related objectives the following aims can be included:

$L_{\max}$ Maximum lateness. The objective is to decrease the lateness of the latest job. Lateness can be less than zero and often there are no benefits in finishing earlier than the deadline. Therefore, it is often more appropriate to work with tardiness instead (see Baker and Trietsch, 2009).

$T_{\max}$ Maximum tardiness. The objective is to decrease the tardiness of the tardiest job. Tardiness for each job can be zero (on time) or larger than zero (late). Tardiness can never be less than zero (T'Kindt et al., 2002).

$\overline{T}$ Average tardiness or total tardiness of jobs. The objective is to decrease the average tardiness of the tardy jobs or the total tardiness of the tardy jobs.

$\overline{U}$ Number of late jobs. The objective is to decrease the total number of late jobs.

Examples of cost related objectives are setup costs, work-in-process inventory costs, finished goods inventory costs and transportation costs. However, there are other costs, such as those related to personnel and equipment, which may also depend on the schedule, but are perhaps not necessarily proportional to other objectives, e.g., makespan.

2.4 Scheduling methodologies

According to the classical definition of the scheduling problem, the goal is to find the best possible schedule (sequences). Makespan is probably the most common objective and means the maximum job completion time. However, it has to be clarified that there might be several objectives and constraints that make the problem itself difficult. The methodology used to solve the problem will differ, depending on what kind of scheduling problem it is and the requirements of the solution, e.g., optimality and time requirements.

Different scheduling methodologies for different scheduling problems are presented in this chapter, based on a classical definition of scheduling problems, i.e., sequencing of operations on several machines (Conway et al., 1967). When all numeric quantities (processing times, due dates etc.) are known in advance, the scheduling problem can be classified as a deterministic scheduling problem. In contrast, numerical quantities are stochastic in a stochastic scheduling problem. A static problem is when jobs are assumed to be available at time 0, and a dynamic problem is when a subset of jobs has a non-zero release or ready time. According to Kiran (1998), scheduling problems can be defined into four different categories: static stochastic, static deterministic, dynamic deterministic and dynamic stochastic and can be addressed by three basic approaches:

- Optimisation-based approaches
- Artificial intelligence-based approaches
- Dispatching rules and simulation-based approaches

2.4.1 Optimisation-based approaches

Optimisation-based approaches attempt to find the optimal schedule mathematically. There are different techniques that may be used according to the problem to be solved. Approaches based on optimal scheduling rules create schedules using a set of rules that are based on the characteristics of the schedule and mathematical properties of the problem. Once it has been proven that a scheduling rule can find optimal solutions for most general causes of a scheduling problem, it can be used for all other problems in this problem class. Examples are that the priority dispatching rule’s shortest processing time (SPT) and earliest due date (EDD) can prove their optimality for minimising the total flow time and the maximum tardiness respectively for the single machine sequencing problem (Baker and Trietsch, 2009). Another example is the adjacent pairwise interchange technique, which can be used for static deterministic problems, to evaluate different sequences by swapping adjacent jobs and checking the objective function to find optimal schedules. Compared to a total or complete enumeration, where all sequences need to be evaluated, the adjacent pairwise interchange technique has an obvious advantage, according to Kiran (1998).

While approaches based on optimal scheduling rules might be inappropriate, due to the huge solutions space when considering larger problems, or even unfeasible with regard to dynamic scheduling problems, implicit enumeration techniques can be used. The functionality of enumeration-based approaches is to find optimal schedules faster, by reducing the computational burden using mathematical analysis and mathematical programming. Implicit enumeration uses the simultaneous evaluation of alternatives and, compared to total enumeration, not all possible combinations need to be evaluated, because promising solutions are kept and unpromising ones are deleted. These algorithms are also called branch and bound, proposed by Land and Doig (1960). Implicit enumeration may, on the other hand, not be used for constrained optimisation problems. Mathematical programming, also referred to as linear programming or integer programming, can represent many quite different scheduling problems and is mainly used to solve constrained optimisation problems. For example, linear programming can be used for scheduling optimisation problems, given that the objective function and the constraints can be defined as linear equations. Another major drawback with the mathematical approaches is that they take a long time to solve even moderately sized problems.

As Laguna and Marti (2003) put it, "*Many real world optimization problems in business, engineering and science are too complex to be given tractable mathematical formulations*". Furthermore, Kempf et al., (2000) also conclude that using a mathematical model with abstractions of the problem directly in a production line and expecting it to work is unrealistic. Accordingly, complex real-world scheduling problems would be impossible to solve using mathematical programming without making huge simplifications, and with these simplifications it may not provide valid solutions.

2.4.2 Artificial intelligence-based approaches

Artificial intelligence (AI)-based approaches are used to generate schedules that satisfy the constraints, so called constraint-based scheduling. AI-based approaches can be divided into three main groups:

- Rule/knowledge-based approaches.
- Artificial Neural Networks (ANNs).
- Meta-heuristic approaches, such as using Tabu Search (TS), Simulated Annealing (SA), or Genetic Algorithms (GA).

Rule/knowledge-based approaches, also called expert systems, rely on rules that evaluate and develop schedules in a manner similar to human experts. These systems need to have input-output components that have information regarding orders, applicable rules stored in a database and a logic component that processes the data by using the rules in the database. There might be rule conflicts that different systems handle differently, e.g., by weighing up the importance of the rules (Kiran, 1998).

According to Jones (2009), these systems can successfully cope with both quantitative and qualitative knowledge. They can handle complex heuristics, cope with huge amounts of information that may directly or indirectly affect the scheduling problem, capture complex relationships in new data structures, and create algorithms that can manipulate those data structures in new and novel ways. The drawbacks are that they can be difficult to build and manage and they become tied to the system for which they are built. Furthermore, they only generate feasible solutions, making it hard to know how close to the optimum any given solution is.

The basic idea of using ANNs for scheduling relies on their power of pattern recognition in “good” schedules. An ANN is trained by feeding data to it from a set of training problems and their acceptable solutions. The trained network can then be presented to a new problem and, depending on how it is built, can generate the answer of a recommended solution. However, using ANNs would be difficult with regard to more complex scheduling problems.

Neighbourhood search techniques mainly consist of the following steps: (1) create an initial solution and evaluate it according to the objective, (2) generate new solutions in the neighbourhood and evaluate them, and (3) select the best solution in the neighbourhood and let it be the new “seed” or terminate the search, if there are no solutions better than the previous best solution. The generating mechanism uses the seed solution to create new solutions according to a predefined pattern, e.g., the adjacent pairwise interchange technique could be used to generate neighbourhood solutions. Examples of widely used neighbourhood search methods are some of the meta-heuristic algorithms introduced earlier, such as TS, SA and GA (Baker and Trietsch, 2009).

TS can be regarded as a modified form of neighbourhood search in its basic form. Stopping at local optima is a well-known problem of neighbourhood search and TS tries to avoid that by occasionally moving to worse solutions. A number of already evaluated solutions are stored in a “tabu list” which makes sure that the same sequences are not re-evaluated. The method used for selecting the neighbourhood solutions and the size of the neighbourhood seems to have a major influence on the quality of the solution obtained (Kiran, 1998; Baker and Trietsch, 2009).

SA selects neighbouring solutions randomly, whilst TS selects the best non-taboo solution in the neighbourhood. The better the value of a neighbouring solution, the higher the probability it will be chosen as the next starting solution. Annealing comes from the physical process of cooling down material slowly. At the beginning of the optimisation process, the value of the objective function tends to fluctuate quite a lot, but at the end the value does not fluctuate significantly (Kiran, 1998; Baker and Trietsch, 2009). Since a GA-based approach is adopted in this thesis, GA is described in more detail.

2.4.3 Genetic algorithms

Genetic algorithms (GAs), originally described by Holland (1962, 1975), may be viewed as a neighbourhood search procedure (Baker and Trietsch, 2009). It can also be classified as a population-based meta-heuristic and belongs to the class of evolutionary algorithms. GAs are based on the Darwinian theory of natural selection, i.e., the survival of the fittest. The first initial solutions are usually randomly generated into a population of solutions. Each of the solutions is then evaluated, after which a new population is generated. The new population is based on current solutions and in the selection strategy a good solution usually has a higher probability of being chosen as the parent to form new solutions. The new solutions called children or offspring of the parent solutions are formed through reproduction, i.e., crossover and mutation. This process is discontinued when the stopping criterion is met, e.g., time or number of iterations. The success of the search for optimal or near optimal solutions is largely determined by the problem structure and the design of the genetic algorithm (Kiran, 1998; Talbi, 2009). GAs can be used for both manufacturing design and planning decisions, such as decisions concerning aggregate planning, material requirements planning, assembly line balancing and facilities layout, as reviewed and tested in Stockton et al., (2004a, 2004b). Khalil et al., (2012) proposed a framework with discrete-event simulation, drum-buffer-rope and GA, and demonstrated an improvement when simultaneously changing the buffer sizes and batch sizes for a multi-objective optimisation problem, i.e., maximising the throughput and minimising the queue length. However, in this review of GA, the focus is on solving scheduling problems which include changing batch sizes, but exclude design parameters such as buffer sizes.

2.4.3.1 Population

A GA is a population-based algorithm and in the conventional GA a generation-based approach is used where the entire population is replaced simultaneously (Rogers and Prugel-Bennet, 1999). A shortcoming of this method is that if several computers are being used in parallel all the computers may not be utilised if the population size is not divisible by the number of computers or if there are more computers than the size of the population. On the other hand, a steady state GA can utilise parallel evaluations in a better way, because the populations overlap.

2.4.3.2 Representation

A permutation is the arrangement of jobs into a row, hence there are $n!$ permutations totally out of n unique jobs (Whitley, 1997). A permutation representation can be used for resource scheduling where the permutation represents a priority queue of jobs. The classical GAs called canonical GAs use a binary string representing the decision variables (Bäck, 1997b), although a real number representation is possible and is probably more intuitive (Davis, 1991). For example, a permutation representation can be used for an actual sequence, and a vector of real values could be used for the capacity size of a buffer. The former is the focus of the review in the following sub-sections on the two most important GA operators: crossover and mutation.

2.4.3.3 Initialisation

The task of the initialisation process is to create an initial population of solutions. This is usually done randomly, but domain-specific knowledge or other information can be used to create the initial solutions (Sastry et al., 2005).

2.4.3.4 Selection

The main task of the selection process is to select parents for mating, in order to generate new offspring. A main feature of this process is to let a better solution obtain a higher probability of being chosen as parent. A common method is the roulette wheel selection that uses a biased roulette wheel which is proportional to the fitness of the different solutions. However, a conventional roulette wheel method may get a premature convergence at the beginning of the search process and, therefore, methods such as tournament selection may be used (Talbi, 2009). Tournament selection simply selects a number of individuals and the best one of these is chosen as a parent.

2.4.3.5 Crossover operators

A well-known scheduling problem is that of the travelling salesman (TSP), which is NP-complete. In short, TSP represents a problem in which a salesman starts at a given city and has to visit each of *n* cities only once while making a round trip. The target is to find the shortest possible path for the salesman. This problem has similarities to other scheduling issues, such as the job shop scheduling problem, and many of its applications can be used for production scheduling as well. The partially mapped crossover (PMX) was introduced by Goldberg and Linge (1985) for the TSP and has been compared to other crossover operators in various GA scheduling studies (Kellegöz et al., 2008; Engin et al., 2011). A great review of crossover operators applied to GAs for scheduling problems can be found in Aytug et al., (2003). A popular crossover operator in scheduling is the linear order crossover (LOX) (Falkenauer and Bouffouix, 1991), which can be applied to both simple and complex scheduling problems (Pinedo, 2008). The LOX is a modified version of the order crossover (OX) (Davis, 1985) and is quite similar to both OX and PMX. However, it maintains the relative order of the positions that need to move due to the insertion of new genetic material. The LOX works in the following way, redrawn from Pinedo (2008) which is based on Liaw (2000) in Figure 2.1:

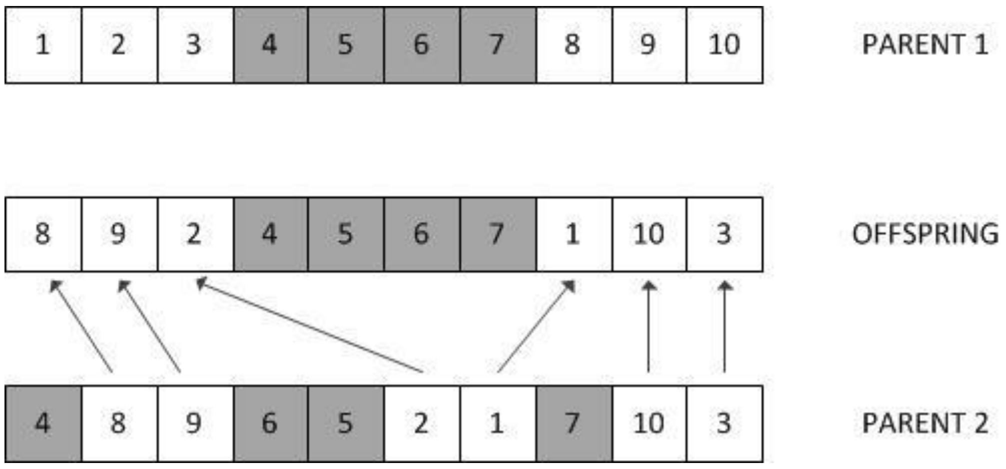


Figure 2.1 linear order crossover.

Basically it works in the following way: a range or a substring is selected from one of the parents, exact positions of which are transferred to the offspring solution, and then the remaining solutions are transferred to the offspring from the other parent. The LOX keeps the internal order of the parent two numbers: 8, 9, 2, 1, 10 and 3 in Figure 2.1, which is different when compared to the OX and PMX, where this internal order could vary.

2.4.3.6 Mutation operators

Mutation operators are used to provide a random diversity in the population of solutions. According to Deep and Thakur (2007), the proportion of the population undergoing the mutation and the strength of the mutation is of great importance when applying a mutation operator. According to Talbi (2009), there are three points that must be taken into account when designing or using a mutation operator:

- Ergodicity: all solutions of the search space should be able to be reached by the mutation operator.
- Validity: valid solutions must be generated by the mutation operator.
- Locality: a small change should be generated by the mutation operator.

There are different types of mutation operators applied to different types of problem representations. Furthermore, there are different techniques when mutation is applied to binary strings, real-valued vectors, permutations, finite-state machines, parse trees and other representations such as hybrid representations. A mutation applied to a permutation must result in a solution that represents a permutation. Most mutation operators for permutations are related to and can be applied for local neighbourhood search strategies (Bäck et al., 1997a).

The 2-opt, 3-opt and k-opt mutation operators generally mean that cut points are selected, between which the sequence is reversed. The following is an example of a sequence of ten elements [A, B, C, D, E, F, G, H, I, J] in which a 2-opt mutation operator is used. If the segment [D, E, F, G], i.e., two cut points, is selected this would result in the complete sequence [A, B, C, G, F, E, D, H, I, J], which would be a minimal change with regard to the TSP, but a larger change for resource scheduling where the permutation represents a priority queue of jobs. Therefore, in order to make a smaller change when considering a resource scheduling problem, it is possible to use *insert, swap* or *scramble*. Insert simply means to select a job and insert it at a random position in the list of jobs. A similar approach, *position-based mutation*, describes a variant of this mutation that randomly selects two jobs and allows the second job to be inserted before the first one (Syswerda, 1991). Another way is to select two jobs and swap their positions (Bäck et al., 1997a) or, in other words, *order-based mutation* described by

Syswerda (1991). Syswerda (1991) also defines a scramble mutation operator that randomly re-orders jobs in a sub-list of jobs (Bäck et al., 1997a).

To further distinguish the different types of mutation operators, there is also *adjacent exchange mutation, displacement mutation* and *inversion/displacement mutation* (Nearchou, 2004). The adjacent exchange mutation, also described as *swap of adjacent elements* in (Bäck et al., 1997a), means that two consecutive jobs swap their positions. A variant of the insert mutation is the displacement mutation that takes a range of subsequent jobs and inserts them into a new position. The inversion/displacement mutation is similar to the latter, but uses the reversion/inversion for the subsequent range of jobs being inserted into a new position.

However, some of the more conventional mutation operators may not be suitable for real-world scheduling problems in which several constraints may make it difficult for them to create valid solutions. To prevent previously good solutions being cast into unfeasible regions of the search space, a domain-specific directed mutation operator that follows the rules of the constraints can be used. Berry and Vamplew (2004) propose Pointed Directed (PoD) mutation in which each gene is tightly coupled to a bit that decides the mutation direction possible for that gene. Korejo et al., (2010) propose a similar approach in which the directed mutation makes an individual shifting based on statistical information, in order to guide the search into a promising area.

2.4.4 Dispatching rules approaches

When it takes longer to actually solve a scheduling problem optimally than to actually execute the work in the shop with any given sequence, there is an NP-hard situation (Baker and Trietsch, 2009). Therefore, in practice, using heuristics such as dispatching rules is often the rule rather than the exception (Baker and Trietsch, 2009). According to Kiran (1998), the scheduling objective is not directly considered when using a dispatching rule. Basically, a dispatching rule is a rule of thumb that gives priority to a job among other jobs at a specific stage, i.e., at a machine. This is why dispatching rules can also be called priority dispatching rules (PDRs). Generally, a PDR-based approach does not try to find an optimal schedule, but relying on knowing that one scheduling rule statistically performs better than another one is sufficient. According to Panwalkar and Iskander (1977), PDRs can be classified into:

- Simple priority rules: simple dispatching rules, combinatorial dispatching rules, weighted priority rules.
- Heuristic scheduling rules.
- Other rules.

Simple dispatching rules, such as shortest processing time (SPT), earliest due date (EDD), first come, first served (FCFS), among others, are quite simple and intuitive rules. When using combinatorial dispatching rules, also referred to as composite dispatching rules (CDR), a ranking expression is used to create a function of attributes of jobs and/or machines (Pinedo, 2005). An example of this approach can be found in (Tay and Ho, 2008), in which CDRs are generated by genetic programming (GP). In their simulation study they found that CDRs generated by GP outperformed several simple PDRs, when minimising tardiness and makespan objectives. A similar approach is the weighted priority indexes that use a combination of PDRs with assigned weights to each PDR, e.g., Jayamohan and Rajendran (2004) who assign specific weights according to the importance of different objectives. They also take the weighted priority rules one step further, when the weighted dispatching rules have different weights due to more important jobs.

Heuristic scheduling rules are rules that may use human experience expertise together with both simple PDRs and CDRs (Panwalkar and Iskander, 1977).

Other scheduling rules may be those designed for a specific shop, rules based on mathematical functions, and so on. Barman (1997) reveals that combining different priority rules at different production stages is appealing, because it is more practicable and less complex than many of the combinatorial rules. Furthermore, he points out that it is an excellent strategy for achieving better results, when several performance measures are considered. They claim that the consensus of researchers is that in some way a combination of dispatching rules is better than using simple dispatching rules.

The main disadvantage of PDRs is their myopic nature (Tanev et. al., 2004; Tay and Ho, 2008), because local PDRs, at a stage, are far from optimal and no single PDR is likely to perform highly on a range of complex scheduling problems (Pierreval and Mebarki, 1997). In order to improve overall performance, both combined dispatching rules at different stages, CDRs and combined GA with PDRs, e.g., approaches found in Tanev et al., (2004) and Ochoa et al., (2009) have been demonstrated to perform better than simple PDRs.

2.4.5 Simulation-based approaches

A complex real-world scheduling problem comes with many constraints that cannot be ignored if a valid schedule is to be created. To find a good and feasible schedule is much more important than attempting to find a mathematical optimal schedule for near–term production scheduling practice (Sivakumar and Gupta, 2006). At the same time, production facilities tend to exist in an ever-changing environment which also affects the problem structure of the scheduling problem, while at the same time, flexibility is the key to the success of any production system (Groover, 2000). McKay et al., (2002) conclude that flexible and configurable algorithms need to be researched further.

Simulation modelling has the capability to represent complex real world systems in detail, which is its main advantage compared to other methods. It is also very useful for communicating details, such as a scheduling situation, due to the visual aids provided by most simulation software. According to Koh et al., (1996), a simulation model built for scheduling is quite different compared to an ordinary simulation model which is generally used for the design and analysis of an existing or proposed system. Simulation-based scheduling, on the other hand, is used for the on-going operation and control of the system, and the ultimate output is a detailed operation plan. Hence, models built for simulation-based scheduling need to be more detailed compared to typical simulation models. Typical simulation models are usually stochastic when analysing design, and so on, whilst scheduling simulation models are usually deterministic. Koh et al., (1996) also identifies a number of important requirements for discrete event, simulation models used for scheduling, namely, flexibility, speed, and details. A model needs to be flexible enough to cope with changes in the physical configuration, fast enough so a schedule can be generated in an acceptable time, and detailed enough with an appropriate level of simplification.

Simulation-based scheduling approaches are derived from the group of dispatching rule-based approaches. In a simulation-based approach several dispatching rules might be used at different stages, in order to make a decision (Kiran, 1998). Many real-world optimisation problems can only be treated by simulation models (Laguna and Marti, 2003), but the problem is that simulation is not an optimisation in itself (Law and McComas, 2000). Therefore, simulation-based scheduling may include much user intervention, in order to manually test different schedules, which would be unfeasible with regard to larger optimisation problems. In order to automatically search for near optimal solutions, a scheduling problem can be solved by using the simulation-based optimisation (SBO) approach in which the simulation model is integrated with meta-heuristic search methods, such as TS or GA (Laguna and Marti, 2003).

In this approach the simulation model is viewed as a black box function evaluator which evaluates a set of input parameters generated by the meta-heuristic optimiser. The response or output is used by the meta-heuristic optimiser to generate new values of the inputs. Simulated annealing may be viewed as a sort of random search procedure, but its main disadvantage is the computational time it takes to find a good solution. The main advantage of evolutionary approaches, such as GAs, compared to those that use neighbourhood search-based methods on a single solution, e.g., simulated annealing, is that fewer evaluations are needed in order to search a larger area of the solution space. Finding good solutions early in the search process is particularly important regarding SBO (April et al., 2003).

The weakness of simulation is that it is time consuming, which can be somewhat compensated by SBO, because it does not try to evaluate all solutions, but rather a fraction of the whole search space. Furthermore, it is possible to parallelise the simulation evaluations (e.g., Li and Wang, 2008) and to use a steady state GA (Rogers and Prugel-Bennet, 1999) in order to speed up the optimisation process. The weakness of GAs and other meta-heuristic search methods is that they may not find the optimal solution for larger scheduling problems. On the other hand, the question is whether any method would find an optimal solution in an acceptable period of time for an NP-hard, complex real-world scheduling problem? It is important to note that an optimal solution is usually not the target in a complex real-world scheduling problem; it is instead to achieve a relatively high performance for many problems, which is a characteristic of GA (Sankar et al., 2003).

2.4.6 GA or dispatching rules for simulation

In the last decade, there has been extensive research in the field of production scheduling using simulation. Simulation modelling has the capability to represent complex real-world systems in detail, and several dispatching rules can be used at different stages to make decisions about what parts to select for the next scheduling period. The number of rules can be infinite, because it is possible to define new scheduling rules as the combinations of several other dispatching rules (Holtaus, 1997). Generally speaking, a PDR-based simulation scheduling approach does not attempt to find an "optimal" schedule, but relies on knowing that one rule, or a combination of rules, performs better than another one. In comparison, using a meta-heuristic optimiser, such as a Genetic Algorithm (GA), to generate the near optimal schedules directly, which is referred to as a direct approach in this thesis, may be advantageous if searching for "optimal" solutions is desired. Nevertheless, to generate a complete schedule using a GA-based SBO may require very long computing time. This is usually impractical or even unacceptable, if the result is needed to control the system in "real-time".

There are many studies that compare these two approaches and some of them provide results showing that the use of GAs to generate detailed schedules can obtain better solutions than those obtained by using PDRs. For example, Sankar et al., (2003) use a GA for the scheduling of a job shop with five production stages, parallel machines in each stage and 43 jobs to be scheduled. Several objectives, including customer satisfaction, machine utilisation and total elapsed time, are integrated into a single combined objective function. A GA is coded in such a way that the chromosomes represent the job sequences which the manufacturing system has to follow in order to achieve the best schedule. The results obtained with the GA are then compared with the results obtained using six different dispatching rules including SPT, LPT, EDD, largest batch quantity (LBT), smallest batch quantity (SBQ), and highest penalty (HP). It has been found that the solutions generated by GA outperform the solutions obtained by using PDRs, for this specific production system.

The most common form of hybridisation is combining a GA with local search procedures or using domain specific knowledge. Hybrid genetic algorithm and memetic approaches have achieved good results in complex real-world application areas, but there has been limited work developing a theoretical basis for genetic algorithm hybridisation (Sastry et al., 2005). Kim et al., (2007) made a comparison between the use of PDRs and GAs for solving the scheduling problem in a real factory that manufactures standard hydraulic cylinders. More specifically, it was a job shop of six machines and nine jobs. Different dispatching rules were used in this study, namely SPT, LPT, most work remaining (MWKR), and least work remaining (LWKR). When using GAs, different jobs to be performed by different machines are codified into an individual chromosome, and then the different individuals are selected following the "natural selection", in order to minimise makespan. Again in this study, the researchers found that the GA-based approach outperforms the PDR-based one. At the same time, the researchers state in their conclusion that better results could be found if the two techniques for the scheduling of orders are used in combination. An example is Kianfar et al. (2012) that propose a hybrid GA procedure that uses PDRs to generate initial solutions. Overall, the algorithm was shown to be better than some common dispatching rules, when compared in four flow shop scheduling scenarios.

A method that combines GA and PDR can be found in Tanev et al., (2004), where a hybrid evolutionary algorithm for the scheduling of a plastic injection machines factory was developed. The system was a job shop with four machines and 50-400 jobs. In their approach, the researchers proposed a hybrid GA combined with the use of PDRs; a GA was used to evolve the different combinations of dispatching rules and to finally find which one provides the best schedule. The solutions were then evaluated by means of a fitness function conformed by the different parameters (flow times, setups, makespan, tardiness, etc.) to be optimised. They found that letting the GA select PDRs generated better solutions compared to a conventional GA, in a shorter time period. At the same time, the computational effort/job to be scheduled seems to decrease with an increasing number of jobs, making it particularly appropriate for complex real-world problems compared to a conventional GA. Another study has been carried out by Ochoa et al., (2009), in which the hybrid flow shop was considered as $HFk\left(QM^{l}\right)_{l=1}^{k}$, where k is 5-30 stages and M is four to five machines. A conventional GA creating a permutation schedule was compared to a GA that selected dispatching rules, and the latter approach was demonstrated to be advantageous compared to the conventional GA.

These methodologies that use some sort of meta-heuristic, e.g., GA, in order to select other heuristics, e.g., PDRs, may also be referred to as a hyper-heuristic approach, in which some meta-heuristics are used to select the appropriate heuristics (Burke at al., 2003). This kind of approach in which a GA chromosome is used to represent different combination of PDRs, is referred to as the indirect approach in this thesis. The reason is that the actual sequence itself is only indirectly handled by the GA using the PDRs. Burke at al., (2003) reveal that current meta-heuristic search methods tend to solve and be customised for a particular problem type, whilst hyper-heuristics are able to handle a wider range of problems and may lead to more general systems. Algorithms' ability to adapt and learn has been identified as future research issues (McKay et al., 2002).

2.5 Assumptions usually made in scheduling research

Even moderately sized scheduling problems tend to become complex. Gupta and Stafford (2006) state that research within flow shop scheduling seems to have been motivated by what the researchers can achieve rather than what is important, and thereby also suffers from too much abstraction and too little application. According to Pinedo (2008), advances in scheduling theory have only had a limited impact on scheduling in practice, but the theoretical research has not been a complete waste of time, because it has given insights into the scheduling problem. Still, looking at 50 years of research, theoretical flow shop scheduling problems remain largely unsolved (Gupta and Stafford, 2006). Generally, scheduling problems include many restrictive assumptions to be solved (Kiran, 1998). Many of these assumptions are valid for different scheduling problems, but it would not be true to say that these assumptions can be used for all different kinds of scheduling problems. Assumptions are usually made about scheduling problems and some of the most general ones include the following (e.g., Baker and Trietsch, 1974; Ramasesh, 1990; Kiran, 1998; Baker and Trietsch, 2009):

- All jobs to be scheduled are available at time zero.
- Machines can only process one job at a time.
- Setup times are sequence independent, i.e., there are no sequence dependent setup times.
- Setup times are included in the processing times.
- There are not any breakdowns of machines, i.e., the machines are continuously available for production.
- Jobs are processed without any disruptions.
- There is no alternative routing of jobs, i.e., jobs have strictly ordered operation sequences.
- No parallel machines can do the same type of operation.
- An operation may not start before the preceding ones are finished.
- There is no pre-emption of jobs, i.e., once started jobs must be processed until completion.
- A job may not be started before it is finalised in previous operations.
- There is no variation of processing times.
- Jobs are moved directly between production stages, i.e., there are no transfer times between machines.
- Buffer sizes (queue lengths) are not limited.
- There are no assembly operations.
- Jobs are carried out on a machine only once.
- There is no rework of jobs.

When considering complex, real-world production scheduling problems, only a few of these assumptions can possibly be made without changing the original issue into a completely different scheduling problem, i.e., a theoretical scheduling problem that is not of much use in practice. In a review of flow shop scheduling research, Ribas et al., (2010) states that even though most real-life situations are better represented by models with uniform or unrelated machines, most research has been done on flow shops with identical machines, which is probably due to the fact that identical machines are easier to handle. According to Allahverdi et al., (2008) who reviewed 300 papers on scheduling with setup times, between 1999 and 2008, there has been a significant increase in scheduling problems involving setup times. The reason is that substantial savings can be made, when setup times are considered for real-world industries. The majority of papers dealt with sequence independent setup times, because this is easier to handle compared to sequence dependent setup times. Again, according to Ribas et al., (2010), most research has been carried out with, at most, one constraint (e.g., setups, failures, blocking) being studied at a time and only a few studies dealt with all or most constraints at the same time. Consequently, in order to diminish the gap between theory and real-world scheduling problems, several constraints need to be considered simultaneously.

2.6 Scheduling objectives in real-world problems

Most real-world scheduling problems have more than one objective of interest (Gary et al., 1995; Yang and Chang, 1998), commonly defined as multi-objective scheduling problems. However, most of the theoretical literature addresses single objectives only (Graves, 1981; Allahverdi et al., 2008; Ribas et al., 2010).

There are different ways to address multi-objective scheduling problems, of which some can be found in Kempf et al., (2000). One way is to use the primary objective as the one to optimise and a secondary objective as a constraint. Another strategy is to use a multi-objective approach and let the user decide from a set of Pareto (non-dominated) solutions. For example, when using the Elitist non-dominated sorting genetic algorithm (NSGA-II), the Pareto front consists of all the solutions that are not dominated by other solutions of at least one objective (Deb et al., 2000). Another common approach is to combine different objectives into a single one by using weights for the different objectives of interest (Kempf et al., 2000). Finally, a similar approach to the latter one is to use a cost-based objective (see Section 2.3.4) where all the objectives are measured in cost (Kempf et al., 2000). Real-world scheduling problems usually have multiple objectives. Whilst a Pareto set of solutions of multiple objectives may be beneficial when analysing a production system, it would require much time from the production scheduler and is probably better suited for other types of SBO problems, such as optimal buffer allocation ones. A weight-based objective function was adopted in the thesis, because the production scheduler needed to obtain the result quite quickly and the user had no time to study separate sub-targets.

Regarding real-world problems, different organisations have different objectives and therefore the scheduling metrics will vary from case to case. At a higher company level, profit is the important long-term objective, along with customer satisfaction; however, the importance of customers may vary depending on the customer. As a matter of fact, on the production floor, the supervisor might want high overall machine utilisation and throughput rate by having bigger batch sizes, as demonstrated in Stockton et al., (2012), and an operator might want homogenous batches, in order to avoid setups on a certain machine.

2.7 Concluding remarks

In summary, it has been emphasised in this chapter that even a moderately sized scheduling problem tends to be too complex to solve by any analytical approaches and many real-world problems, such as the hybrid flow shop, belong to the class of NP-complete problems. In other words, it could be possible to solve real-world scheduling problems using mathematical programming, but it would require huge simplifications, as reviewed in this chapter. Flow shop scheduling seems to have been motivated by what the researchers can achieve rather than what is important, and thereby also suffers from too much abstraction and too little application. Discrete event simulation has the capability to represent complex real-world systems in detail, as well as cope with several constraints and multiple objectives, which have been identified as important factors. By using the simulation-based optimisation (SBO) approach in which the simulation model is integrated with meta-heuristic search methods, such as genetic algorithms, the search for optimal or near optimal solutions can be done automatically. A main advantage of using genetic algorithms is that quite a few evaluations are needed, in order to search a large area of the solution space. Furthermore, combining GA with dispatching rules (hyper-heuristics) seems to be a promising research direction, according to several researchers reviewed in this chapter. Therefore, a hybrid genetic representation is proposed in this study and presented in Chapter 5. The scheduling problem from the perspective of uncertainty is dealt with in Chapter 3.

Chapter 3

3 Rescheduling and System Support

This chapter describes the functions and features needed in order to support production scheduling in real-world problems which are subjected to disturbances such as machine breakdowns. It begins with a brief introduction of how uncertainty affects the execution of a schedule and continues with rescheduling methods and policies to handle uncertainty. Furthermore, functions of the scheduling task are presented, which is followed by a review of important scheduling system functions identified in the research society. Finally, based on the literature reviews in chapters 2 and 3, a summary of the most important functions of a production scheduling system is presented.

3.1 Uncertainty and rescheduling

In the research of the higher levels of production control, there have been successful practical implementations of research, such as Enterprise Resource Planning (ERP) systems used more often nowadays by companies in industry (McKay and Wiers, 1999). However, there have been very few successful practical implementations or usable optimisation methods in dynamic job shops and detailed dispatching (McKay and Wiers, 1999; Stoop and Wiers, 1996). In fact, McKay and Wiers (1999) depict that the underlying principles of scheduling research are insufficient and should be reassessed. A common opinion is however that the theoretical techniques are actually applicable, but people in industry do not use them, because they do not know how to apply them or simply because they are not aware of their existence (McKay and Wiers, 1999). The traditional definition of scheduling is more about sequencing, while the impact of uncertainty is systematically underestimated by academic research. Frequent schedule interruptions may occur during the execution of a schedule in a production system, due to the variability present in these systems (Stockton et al., 2012). According to McKay and Wiers (1999), a common approach to uncertainty is to react and reschedule. In some way, it is possible to reduce uncertainty by taking precautionary actions, such as preventive maintenance, but it is hard to remove uncertainty completely. Some researchers who address specific scheduling problems do include uncertainty in the scheduling problem with stochastic arrival and/or processing times, e.g., Daniels and Kouvelis (1995) and Leon et al., (1994). However, with regard to hybrid flow shop scheduling problems, most research papers do not consider uncertainty or other related constraints, or they simply handle only one constraint at a time (Ribas et al., 2010).

Graves (1981) identified that scheduling robustness is an important area of future research and vividly stated that *"A frequent comment heard in many scheduling shops is that there is no scheduling problem but rather a rescheduling problem"*. Production scheduling research can be divided into two groups, namely, deterministic scheduling research, in which the problems are defined with deterministic terms, and stochastic scheduling research, whereby at least some randomness is modelled for the problems. Aytug et al., (2005) reveal that many of the stochastic scheduling research efforts have focused on local control policies, such as priority dispatching rules, aimed at minimising some measure of performance. Most of these methods do not use any information about the global state of the shop floor and create the schedules during executions. The deterministic scheduling research is more focused on creating an optimal or near-optimal schedule, according to a single or multiple objectives, usually with regard to a single or multiple machines. The problem with the deterministic solutions obtained is that it is assumed they can be exactly executed in the real machine/line/shop for which they are created. However, many researchers have recognised that uncertainty is always part of the problem and therefore put effort into extending the deterministic approaches to enable them to handle some form of uncertainty.

The predictive schedule could be described as the forecasted "optimal" schedule found by the scheduling approach used and may be updated with a new predictive schedule when required. When this predictive schedule is used in the real world, very often with regard to disturbances, it is called the realised schedule. Stoop and Wiers (1996) have found that the expected performance of a (predictive) schedule often deviates from the (realised) actual performance which, in most cases, is worse than the expected performance. Three categories of disturbances that cause these performance deviations have been classified (Stoop and Wiers, 1996): (1) *capacity disturbances*, such as machine breakdowns; (2) *order disturbances*, such as rush orders; (3) relates to the measurement of data, such as estimated processing times used in the scheduling process. The quality of data affects the uncertainty and is very important, but a high quality measurement of data could be difficult to obtain in some production systems. Therefore, these three types of uncertainties are included in the reactive scheduling experiments presented in Section 7.4. Viera et al., (2003) further present a framework to classify rescheduling research in which uncertainty plays some key role. Such a rescheduling framework includes *rescheduling environments*, *rescheduling strategies*, *rescheduling policies* and *rescheduling methods,* which is discussed in more detail in the following sub-sections.

3.1.1 Rescheduling environments

The rescheduling environment refers to the problem instance to be rescheduled, i.e., whether it is a finite set of jobs (static) or an infinite set of jobs (dynamic). In a static and deterministic environment (instance), nothing is unknown and a rescheduling is not necessary. In a static and stochastic environment, there is a finite set of jobs but some uncertain variables exist, such as the processing times of the jobs. When there is no arrival variability of the jobs in a dynamic environment, a cyclic schedule that is executed repeatedly could be used. On the other hand, when there is an arrival variability of the jobs, but all the jobs have the same route, the sequence cannot be reused if a direct representation of the schedule is used. Finally, process flow variability and arrival variability of the jobs may co-exist, which is mostly characterised in job shops, where a great variability of job arrivals is very common.

In terms of rescheduling strategies, two common categories of approaches can be identified: (1) dynamic - completely reactive approaches, and (2) predictive-reactive approaches, which are discussed below.

3.1.2 Completely reactive approaches

In a dynamic approach, the schedule itself is not generated beforehand, but jobs are dispatched at the machines in real-time. Dispatching rules, or other types of heuristics or control policies, characterise dynamic scheduling (Viera et al., 2003). This group is also called completely reactive approaches (McKay and Wiers, 1999), as the dispatching rules actually react to the events that are taking place and dynamically generate the sequences. Different approaches using dispatching rules are reviewed in Chapter 2, such as simple priority dispatching rules (PDRs) (Panwalkar and Iskander, 1977), combined dispatching rules at different stages (Barman, 1997), composite dispatching rules (Tay and Ho, 2008), and combined GA with PDRs (Tanev et. al., 2004; Ochoa et al., 2009). Dispatching rules have the capability to keep the machines utilised, as long as there is material waiting in the queue, but it is nonetheless hard to know the performance of the realised sequence order in the presence of uncertainty. The realised sequence order may have a significant impact on the performance of the schedule, if sequence-dependent setup times are present (Allahverdi et al., 2008).

3.1.3 Predictive-reactive rescheduling policies

When the schedule is generated beforehand, i.e., direct representation of the schedule, there are different policies to decide when to reschedule, in order to update the predictive schedule. Church and Uzsoy (1992) present a rough taxonomy of the existing approaches, namely: periodic, continuous, and event-driven rescheduling. Periodic rescheduling is when rescheduling takes place periodically with a predetermined time interval. The event-driven rescheduling is triggered as soon as a “big enough” disruption occurs. In other words, if the realised schedule deviates too much from the predictive schedule by some measure, then a rescheduling will be executed. An example is Kianfar et al. (2012) that use an event-driven triggering based on the arrival of new jobs and reschedules if the number of jobs or time elapsed since last rescheduling is big enough. Continuous rescheduling is an extreme case in which each event starts a new rescheduling. Periodic rescheduling may also be seen as a form of event-driven rescheduling policy. Additionally, in hybrid rescheduling policies, periodic rescheduling is combined with event-driven rescheduling (Herrmann, 2006). Church and Uzsoy (1992) studied one stage, one machine and a parallel machines’ problem with dynamic job arrivals, for the purpose of decreasing maximum lateness, and show how the rescheduling frequency affects the schedule performance. Suwa and Fujiwara (2007) propose a new hybrid rescheduling policy based on the cumulative delay of jobs, i.e., differences between the predictive and realised schedule that outperform a combined periodic and event-driven rescheduling policy for a single machine scheduling problem and a parallel machines’ scheduling problem, which showed positive results. Actually, periodic and hybrid rescheduling policies seem to be the most common ones in practice (Herrmann, 2006). Since the main approach used in this thesis is react and reschedule, a hybrid rescheduling policy has been adopted, as described in Section 4.4.4.

3.1.4 Rescheduling methods

While a predetermined sequence created by a direct approach could be re-sequenced when a disruption occurs, a more novel approach is to generate the sequences to be robust enough to handle uncertainties. Robust scheduling approaches, also called proactive approaches, focus on creating a schedule that, when implemented, will be robust enough to handle different disruptions and minimise their effects with respect to some performance measure. These approaches can be further classified: (1) optimising the worst possible scenario; (2) minimising differences in objective function, subject to disturbances, and (3) to include the effects of machine failures, subject to a given rescheduling method. Daniels and Kouvelis (1995) develop a procedure for creating robust schedules, by analysing worst case scenarios. Leon et al., (1994) create a schedule approach that shows robustness for processing time variability and machine failures with makespan as the minimisation objective. Leon et al., (1994) develop robustness measures that are used with a GA to find robust schedules. Another approach to optimise buffer allocation in a job shop was proposed by Al-Aomar (2002). In this method, the author achieves robustness by integrating it into the GA search engine through assigning a Signal-to-Noise ratio (S/N) to each simulation outcome. The method has been applied to a hypothetical job shop example with buffer sizes as the discrete factors (Al-Aomar, 2006).

When rescheduling is necessary, e.g., due to the deviations of the initial plan, there are different ways of repairing a schedule. According to Herrmann (2006) and Viera et al., (2003), there are three ways to reschedule: (1) complete regeneration, (2) right-shift scheduling, and (3) match-up scheduling or so called partial rescheduling. Complete regeneration means that the whole schedule is regenerated, i.e., all the jobs that have not been executed by the time of rescheduling will be rescheduled. A complete rescheduling may lead to schedule nervousness (Stoop and Wiers, 1996), which, according to McKay and Wiers (1999), can be overcome in most real-world situations, if small changes are continuously updated and only partial solutions are generated. Right-shift scheduling means that the remaining jobs are postponed by the time needed to obtain a feasible schedule. Right shift scheduling may be seen as a simple form of match-up scheduling, since the jobs are shifted to the right in the Gantt chart, without any re-sequencing being done. Match-up scheduling means the necessary actions to be able to get “back on track” with the predetermined schedule. The match-up point indicates what part of the schedule has to be rescheduled. Bean et al., (1991) propose a match-up heuristic method that begins with incrementally searching for the appropriate match-up point with regard to machine disruption. Jobs are rescheduled for the machine, or machines, with the disruption, using several dispatching rules. If jobs can be rescheduled without exceeding the threshold for the tardiness costs, the search stops. If a schedule cannot be found for a given, maximum match-up time point for the machine(s), then the search is extended by scheduling several machines. Akturk and Gorgulu (1999) propose a match-up heuristic procedure that determines the match-up point and does the rescheduling for a modified flow shop. Since both the match-up point and the new schedule for that period are determined simultaneously, a heuristic procedure was chosen, involving different dispatching rules, in the creation of a new schedule. In this thesis, all three ways to reschedule have been adopted, as described in Section 4.4.4.

3.1.5 Direct, indirect and hybrid representation of schedules

Several researchers, e.g., Sankar et al., (2003) and Kim et al., (2007), have shown that global scheduling, i.e., a direct representation of schedules, using GA, has the potential to improve the performance of complex shops, compared to dispatching rules. Lawrence and Sewell (1997) also compare dynamic heuristics, e.g., dispatching rules, with static algorithms such as shifting bottleneck heuristic, for several job shop scheduling problems with makespan objective and different degrees of processing time variability. They found that simple dynamic (real-time/on-line) scheduling heuristics yield equally good or better results compared to complex static (off-line) algorithms, especially when complexity and uncertainty are increased. Wan (1995) shows similar results to the latter, in which a dynamic dispatching rule yields equally good or better results compared to static methods, when subjected to processing time variability. Regarding scheduling problems with high uncertainty, many studies have confirmed that an indirect representation of schedules, such as dispatching rules, can produce better solutions compared to a direct representation of schedules (Lawrence and Sewell, 1997, Matsuura et al., 1993; 1997; Wan, 1995). In order to show that the predictive-reactive approach using a direct representation could be better, even when the uncertainty is quite low, Matsuura et al., (1993) propose a hybrid approach called switching. In such a hybrid approach, a predictive schedule is created for the shop which uses a periodic rescheduling policy. If the realised schedule deviates significantly from the predicted one, the system switches to using a dispatching rule for the remainder of the period. Another hybrid approach, which includes a global scheduler and a dispatching module for a job shop with variable processing times, is proposed by Roundy et al., (1991). In this approach, the dispatching module selects a job, which is based on the outcome of deriving the costs associated with performing a job at a particular time, from the global schedule. With increasing shop complexity, this method has been shown to perform well in comparison to dispatching rules. A similar hybrid approach to the latter, called SB-DIS, was proposed by Barua et al., (2005). A global schedule is created for the shop which uses a periodic rescheduling policy. The global schedule is implemented directly, but serves to provide a priority index for the jobs. Compared to the latter approach, the global schedule does not need to be feasible, but serves as a priority index for jobs used by the dispatching procedure. SB-DIS was tested on both a deterministic and a stochastic, hypothetical multi-stage shop problem and generally showed that it outperformed different dispatching rules.

3.2 The scheduling task

Conway et al., (1967) state that a scheduling problem taken out of its context gains in generality, since it approximates many situations, but does not represent a solution to any real-world sequencing problem. This information is only a partial assessment of the real problem. McKay and Wiers (1999) claim that researchers and real-world schedulers are not discussing the same problem, since researchers are solving the sequencing problems and real-world schedulers are faced with day-to-day challenges, such as communicating with personnel about events of the previous night. A critical task of a scheduler is also to check the current status of the plant with regard to demand, machines, material, and personnel. Another task is to anticipate and plan future events, such as machine maintenance and repair issues, processing changes, and new product samples. When planning what has to be done, where, and by whom, there is almost always a compromise, due to the wide range of options faced by the scheduler. This is why McKay and Wiers (1999) define the scheduling task as: "*a dynamic and adaptive process of iterative decision making and problem solving, involving information acquisition from a number of sources, and with the decisions affecting a number of production facets in reaction to immediate or anticipated problems*", which this work is based on.

3.2.1 Functions of the production scheduling task

Wiers (1997) proposes that four types of control can be used to further characterise a scheduling task: Detailed control, Direct control, Restricted control and Sustained control. In Detailed control, the scheduling is very detailed in order to deal with the short-term dispatching decisions that determine what to do next (Wiers, 1997). It is important that a valid schedule for a short-term scheduling horizon is generated, because there is no intermediate control before the schedule is launched and there is a risk that the schedules have to be adjusted manually (Stoop and Wiers, 1996).

Direct control means that the scheduler has direct control to answer questions and give directions, as the schedule has been created without any intermediate control before its launch (Wiers, 1997). Methods such as completely reactive approaches or predictive-reactive approaches are possible solutions, but in real-world situations these procedures need to have some system support, in order to be able to provide the direct control functions.

With regard to Restricted control, schedulers have to deal with the situation at hand, with material availability and requirements usually beyond their control (Wiers, 1997). McKay and Wiers (1999) explain that the decisions made regarding various problems may differ, depending on the kinds of situation, such as the beginning of a day or a Friday afternoon. The scheduling process needs to be able to answer questions in a limited amount of time and small changes to the schedule must be made continuously throughout the day, even if there is not a complete set of data available. Instead of a complete rescheduling, some sort of partial rescheduling could possibly reduce the risk of schedule nervousness.

Finally, Sustained control refers to the scheduler that monitors schedule execution and carries out necessary changes when needed, in order to fulfil scheduling targets (Wiers, 1997). Consequently, a solution for real-world scheduling problems would have to include detailed monitoring capability.

3.3 System support

As Pinedo (2005) vividly states, "*Analysing a planning or scheduling problem and developing a procedure for dealing with it on a regular basis is, in the real world, only part of the story. The procedure has to be embedded in a system that enables the decision-maker to actually use it. The system has to be integrated into the information system of the organization, which can be a formidable task*". Therefore, in order to be able to handle the scheduling task that takes uncertainty into account, a scheduling system, not only a scheduling algorithm is needed. Framinan and Ruiz (2009) believe that scheduling research needs to increase studies in areas such as user interfaces, data management, scheduling monitoring, as well as in more tools and methods for the design and implementation of scheduling systems for manufacturing facilities. Hence, this review identifies important functions and features that need to be handled by a scheduling system.

3.3.1 User-interfaces and human control

Improvements can usually only be made through the scheduling process in practice (McKay and Wiers, 1999), and the success of a particular technique is greatly determined by its human users (Stoop and Wiers, 1996). In the field study of McKay et al., (1995) at a printed circuit board (PCB) factory, an analysis of a scheduler's task was made to find out which decisions were taken due to uncertainty. The analysis indicates that the scheduler was more of a problem solver and used more than 100 heuristics in order to take precautionary actions and to anticipate problems. Furthermore, Stoop and Wiers (1996) rightly note that humans often rely on their own judgement with regard to the application of techniques and common sense tells them that these techniques are imperfect. The only way to increase the use of new procedures is to have a great deal of transparency, i.e., letting the user see what happens and to offer monitoring support.
User interfaces to support both model input manipulation and schedule manipulation are believed to be an important research area (McKay et al., 2002). Gantt charts are probably the most common way to present schedule information (e.g., McKay and Buzacott, 2000) and there are real-world case studies that allow the user to modify the predictive schedule through a Gantt chart-based interface (McKay and Black, 2007). Higgins (1996) observes that the jobs screen, which displays the attributes of the available jobs, is central to the interactive decision-making and thereby presents a system architecture for human-computer interaction. A jobs screen is both made up of assigned jobs at machines and unassigned ones. Although this approach is possible using the dispatching clients or monitoring programs of the proposed system (see Chapter 6), it is not used in this work because a schedule, generated by the SBO, is used to suggest jobs for the operators in a production line. Consequently, the scheduling system will support the operators with a schedule, in contrast to jobs screens (Higgins, 1996) that would leave this decision to the operators themselves. Scheduling rules/heuristics can be used to test different policies and the knowledge-based adviser will indicate if any constraints are infringed. Higgins also notes that human decision-

making with its ability of pattern recognition and setting things into a context is part of an interactive process for creating the Gantt chart. In a similar vein, McKay et al., (2002) also maintain that the monitoring schedule execution status and evaluation performance is important. Furthermore, they identify the research opportunity of task design, i.e., what functions should be automated and what should be left to human control.

McKay et al., (1999) also rightly point out that disturbances in the process and the environment can be anticipated, reacted to, and adjusted in the scheduling process. A manufacturing system is exposed to uncertainty in many forms, i.e., varying machine processing times, machine failures, quality problems, personnel on sick leave, late supply deliveries, and so forth. Although some uncertainties cannot be predicted, there are some “surprises” that can be foreseen. For example, they mention that the humidity during the summer months is higher and may affect the production line and quality of products, but can be taken into account since it is known in advance. Therefore, a resource calendar interface (Pinedo, 2005) can be used for this reason and also for short-term conditions, such as planned maintenance and shift schedules.

Additionally, Pinedo (2005) provides other examples of various, important user-interfaces that may be used in a scheduling system: plant layout, routing table, capacity buckets, and throughput data interfaces. Plant layout and routing table interfaces are simple user-interfaces for the input data. The capacity buckets interface is used when the time axis is divided into buckets or periods of time, e.g., days, weeks, or months, in order to show the utilisation of the line capacity, when jobs are assigned to these buckets. The benefit of such information is that the decision-maker can be proactive and make sure that the resources can be utilised efficiently over time, e.g., to avoid generating schedules that would require additional work on weekends some weeks, when the extra work could, in fact, be balanced over several weeks of production. The throughput data interface shows information about material waiting to be processed, products delivered, WIP-levels, FGI-levels, utilisation of machines, and so forth. Finally, a column editor could be useful for the scheduler, because it displays lists of jobs in scheduled order divided over the machines (Pinedo et al., 1994).

3.3.2 Flexible objectives over the horizon

McKay et al., (1999) describe that everything changes over time and one day is not like another, e.g., Monday morning is different to Friday afternoon. Consequently, the scheduling function must be able to handle both absolute and relative time. Absolute time is calendar-based information, such as the planned maintenance of machines, while relative time refers to the decisions on the rolling horizon and affects the level of detail and type of constraints used to make decisions. For example, in the next few weeks all the constraints may be relevant when specific, production target levels are to be met, but since the scheduling strategy might be changed, due to a future machine installation, infinite loading may be used to enable preparation for production line maintenance. Stoop and Wiers (1996) state that the scheduling horizon must be determined long enough, in order to avoid generating sub-optimal schedules due to a too-short scheduling horizon. Hence, it is natural that the productivity fluctuates over time.

3.3.3 Feasibility check and fault control

McKay and Wiers (2003) observe that checking the consistency of input data is important, since data may come from many different sources. In addition, Blazewicz et al., (2001) propose a feasibility analysis to ensure that resources, e.g., machines and raw material, are available for scheduling the jobs. Framinan and Ruiz (2009) point out that a standard language, such as XML standard, is needed to facilitate system integration for scheduling systems. However, it may also be important to control scheduling dispatching in real-time, similar to the knowledge-based adviser proposed by Higgins (1996), in order to indicate if any constraints are violated. Hence, breaking soft constraints could give a warning and breaking hard constraints will be prohibited.

3.3.4 Evaluating scheduling systems

Kempf et al., (2000) conclude that one of the problems with implementing systems in industry is the difficulty evaluating the effectiveness of production schedules. An *absolute measurement* may be used to ascertain whether a schedule is good or not on its own. Furthermore, a benchmark result is needed to be able to obtain an absolute measurement. Most real-world problems are NP-hard (Garey and Johnson, 1979), i.e., finding the optimal solutions for them is computationally difficult or not possible. An alternative is to theoretically compute a result with regard to a stable state and compare it against that value. A *relative comparison* means that two or more schedules are available and the best among them may be determined. However, if the system is to be evaluated against a real-world production system, real-world *historical data* can be used for the comparison in a relative or an absolute approach. One way could be to use the historical data as it is and another could be to use the trends of the historical data. Manufacturing facilities are subject to an ever-changing environment and therefore the historical data needs to be updated. *Static measurement* is when the predictive schedule is measured without considering the dynamics of the real system, while a *dynamic measurement* is when the predictive schedule is tested in the real environment with regard to disturbances. The result of the dynamic test would be the realised schedule. A *schedule measurement* is when the schedule itself is evaluated against some objectives, but a good schedule might still leave a production line in a bad state at the end of the horizon. For example, leaving a production line in a WIP status that is too low may lead to a problem later on, and therefore the *state measurement* is of importance as well (Kempf et al., 2000).

When comparing the results of different optimisation methods, one replication is not enough, if the model or the algorithm is stochastic, such as GAs. Comparing average results between various optimisation methods will almost always generate different outcomes, and it may be tempting to proclaim that the method with the better average results is the better one. However, it may be an erroneous conclusion, because there is a risk that the randomness is the cause of the difference between them. A common method is to use a hypothesis test for testing claims:

- H_0: Optimisation method A (OMA) is not better than optimisation method B (OMB).
- H_1: OMA is better (lower) than OMB.

The hypothesis H_1 is the hypothesis an experimenter wants to prove correct, but hypothesis H_0 cannot be rejected until H_1 has been proven statistically correct. Two common statistic methods that can prove whether hypothesis H_1 is true are the t-test and the Mann-Whitney test. An unpaired t-test is based on the difference between the averages of the two groups divided by the standard deviation of the two populations, and if this fractional number is large it is possible to reject hypothesis H_0 and state that hypothesis H_1 is true. Observe that the t-test assumes the data sets in comparison are normally distributed (Lövås, 2006).

The Mann-Whitney test is called a non-parametric test, since it does not need any parameters, such as standard deviation and average, and hence does not assume the data sets are normally distributed (Lowry, 2012). Hypothesis tests, i.e., Mann-Whitney test and unpaired t-test, for the experimental results in Chapter 7 have been used in this work and can be found in Appendix F.

3.3.5 Commercial software and real-world case studies

3.3.5.1 Commercial software

A generic job shop scheduling system named “LEKIN” is presented in Pinedo (2005). Built mainly for education and research, it has also been used in real-world implementations. The system is able to handle many different environments from single machine to flexible flow- and job shops. The machine environment is modelled directly in the software which guides the user to set the necessary settings. Different predefined algorithms as well as user-developed algorithms can be used. A problem with using this software which is related to the validity of the schedule is that the constraints necessary for many real-world scheduling problems cannot be modelled, due to the fact that no real, discrete-event simulation software or language is used. Furthermore, it is not designed to be part of an on-line reactive scheduling system and would need to be re-designed (if possible), in order to handle on-line data. Another type of software which is also used mainly for learning scheduling and comparing algorithms is “Parsifal”

(Morton and Pentico, 1993). However, the software seems to be outdated as it runs only on MS-DOS. A commercial scheduling tool that has the possibility to use discrete event simulation is Delfoi Planner (Delfoi, 2012). The simulation-based version, Delfoi Planner Simu, is a web-based scheduling software primarily used for the analysis of a scheduling situation and possibly also to generate detailed schedules, but without the possibility of on-line scheduling and monitoring. The other version, Delfoi Planner Lite, is without discrete event simulation support, focuses more on the integration with other systems, and only supports a simple, finite capacity planning function. The two systems together could possibly support on-line and reactive scheduling, but it is not possible at this time. Furthermore, no information is available about the optimisation algorithms used to generate schedules, so its capability of handling complex scheduling problems is uncertain.

ILOG (2012) is the name used for an umbrella of products supplied by IBM, and the ILOG solver is the most common commercial tool for constraints programming (Gusikhin et al., 2007). For example, a system based on products within ILOG, called Centralized Vehicle Scheduler (CVS), was developed for the sequencing in a paint shop at DaimlerChrysler (CVS, 2012). Many real-world applications use ILOG products (ILOG, 2012), but the main problem is that the system is primarily based on mathematical programming techniques.

3.3.5.2 Real-world case studies

In a statistical review of flow shop scheduling research between 1952 and 1994 (Reisman et al., 1997), it has been shown that only 5 out of 184 papers dealt with true applications, which is much less compared to other areas within the science of operations research/management. The study carried out by Jahangirian et al., (2010) illustrates that even though scheduling applications have been the most common ones among simulation applications in manufacturing and business between 1997 and 2006, only a small portion of them use both real problems and real data. They also point out that papers addressing real-world problems are important future research. On the other hand, Kumar and Nottestad (2006) present a real-world, decision support system for the scheduling of a plastic parts manufacturing line, using discrete event simulation. The application uses the discrete-event simulation software WITNESS and a heuristics to generate job allocation for two different lanes in a manufacturing line for plastic parts. The heuristics used in the simulation includes many different constraints, although the model itself is deterministic. It seems that a predictive-reactive approach is used in which a manual periodic and event-driven rescheduling strategy is applied. Furthermore, Excel, Microsoft Access, and Visual Basic for Applications (VBA) are used to present input and output data for decision support. The result of a scheduling cycle is the output data report which can be printed and delivered to the shop floor.

Dangelmaier et al., (2006; 2007) present a simulation-based scheduling system with real-time control. Although the experiments in their research are not based on real-world data, the system idea of online reactive scheduling is based on a realistic problem. The system is divided into two parts: predictive scheduling and reactive scheduling. In the predictive scheduling part, the schedules are generated in two steps. In the first step, an optimisation algorithm generates a semi-feasible schedule, because not all constraints are considered, e.g., buffer sizes. A simple heuristics sequences each job with the longest tail order at the earliest available machine based on the bottleneck stage and then applied for all the stages. The schedule generated from the optimisation algorithm is simulated in a discrete event simulation model in order to obtain a valid schedule. Another simulation is started with the activated Flow Analyzer Module that may override the current schedule, by using rules mostly based on the waiting times of jobs in the system. The schedule from the predictive phase is executed on the manufacturing floor. Once there is a process disturbance, the rescheduling mechanism is activated. The real-time monitoring and control module starts the simulation evaluation function when a disturbance occurs. In order to generate a new schedule, two algorithms are used: an optimisation rescheduling algorithm that reschedules as few jobs as possible and the match-up rescheduling algorithm which tries to get the current schedule back on track. Thereafter, a simulation is started together with the Flow Analyzer, and the user may decide whether to apply the new schedule or whether the current one is preferred. The proposed system was implemented in the discrete event simulation software Tecnomatix eM-Plant (Plant Simulation) and partly tested in Dangelmaier et al., (2006) for a hypothetical flexible flow shop with parallel and identical machines at three stages. The results show that the combination of simulation and optimisation is better than only using optimisation or random scheduling alone, when the number of jobs increases. The system proposed by Dangelmaier et al., (2006; 2007) has only been tested for theoretical problems and uses simple optimisation heuristics. Furthermore, the “system” is implemented as a module inside discrete event simulation software, which would limit its general use in other applications. However, to our best knowledge, it is one of few simulation-based scheduling systems that deal with online reactive scheduling.

McKay and Buzacott (2000) describe two different, industrial real-world case studies, one with a high volume low product mix and the other with a low volume high product mix. The first case study revealed that the decisions of the scheduler were too difficult to handle in computerised scheduling software, which was therefore stopped before implementation. However, if the product mix had been higher, the need for scheduling software would have been desirable. In the second case study, a production planning system using an evolutionary approach was implemented. The tool was built in Excel with VBA and produced Gantt charts. In addition, the necessary reports were printed and delivered to the shop floor. The scheduling tool generates schedules in a short horizon of two days, but deals with various real-world constraints.

McKay and Black (2007) describe the evolution of a real-world scheduling system that supports the tasks of the scheduler in a job shop environment. In some ways, the shop may be defined as a re-configurable flow shop, since several machines were put together in order to form a line without intermediate inventory between production stages. Initially, a two-week cyclic schedule was desired, and one of the key issues was setup reduction and workforce constraints. A first prototype was built in Excel and VBA, used simple heuristics and presented the result in a Gantt chart. However, the system has been developed over a ten year period into to a small mini-MRP system with finite capacity. The scheduling or sequencing task has been divided into two parts, namely, scheduler and dispatcher. The scheduler handles the long-term (weeks) scheduling and the dispatcher handles the short-term (two days) reactive dispatching using heuristics. The scheduling system has a number of different functionalities to support the work of the scheduler, such as user interface for modifying the predictive schedule through a Gantt chart and various kinds of output reports.

3.3.6 Proposed architectures

To briefly summarise the above detailed review, company confidence in existing software tools is reduced, because most tools do not provide a fair representation of a company's scheduling problems, due to the vast simplifications (Tolio, et al., 2010). Furthermore, Pinedo (2005) also states that many of the commercial systems claim that their systems can be used with only minor modifications, however, in reality, the changes required are often substantial. Nonetheless, if a system were designed to be highly modular, it would increase the possibility for the users to expand their functionality and save development time. Framinan and Ruiz (2010) present a system architecture with such a modular attribute that entails both production scheduling and shop floor control, as shown in Figure 3.1 which illustrates a simplified version of their proposed architecture.

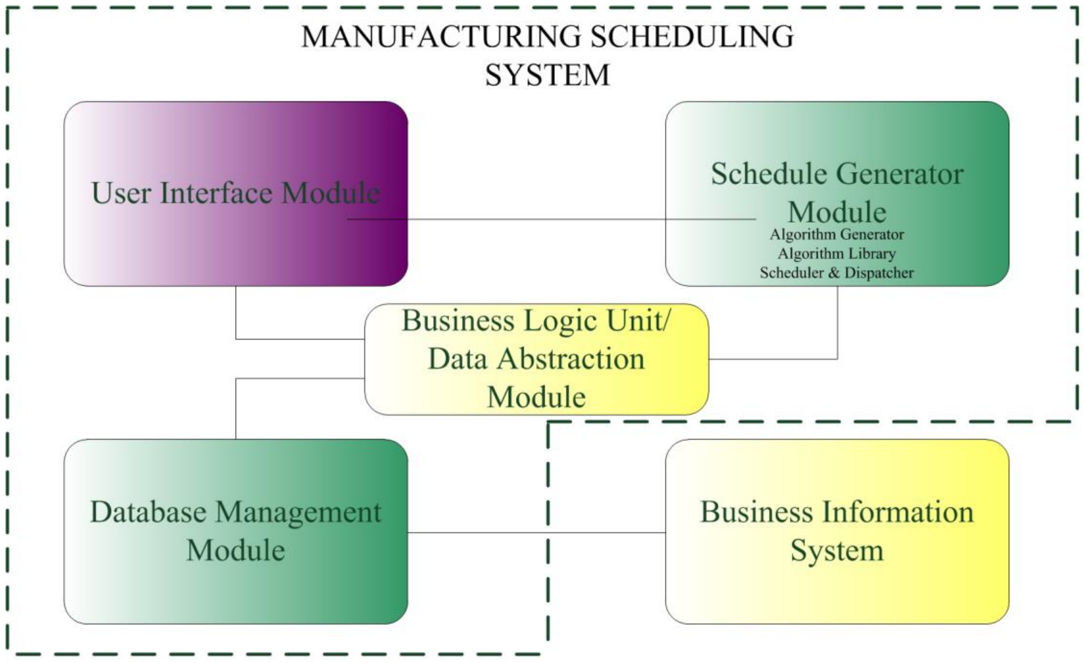

MANUFACTURING SCHEDULING SYSTEM
User Interface Module
Schedule Generator Module
Algorithm Generator
Algorithm Library
Scheduler & Dispatcher
Business Logic Unit/ Data Abstraction Module
Database Management Module
Business Information System

Figure 3.1 Extended modular architecture (Framinan, Ruiz 2010)

The first module, Business Logic/Data Abstraction Management Module, makes sure that data needed is at the required abstraction level. The second, the Database

Management Module, stores the data and handles the import/export of data and production monitoring data from the business information system. The third, the User Interface Module, handles the necessary user-interfaces, while the fourth one, the Schedule Generator Module, handles the functions in order to generate schedules.

The user-interface module consists of five parts, of which the first is the output sub-module that presents necessary Gantt charts and other information. The second part is the scenario management sub-module which can answer what-if questions that may arise, e.g., what happens when the night shift is cancelled? The third part is the system maintenance sub-module which handles the shop configuration and product information. The fourth part is the scheduling control sub-module that handles real-time data from the production. It checks feasibility and input data with each new scenario and warns the user when needed. The fifth part is the algorithm generator interface that allows users to create new algorithms through a user-friendly interface.

The schedule generator module consists of an algorithm library, algorithm generator, scheduler & dispatcher, and pre-processor. The algorithm library contains the different optimisation algorithms, while the algorithm generator sub-module is mainly an object that generates algorithms, based on information in the algorithm generator interface in the user interface module. The Scheduler & Dispatcher use algorithms from the library. In addition, a two-step schedule generation is proposed in which the schedule(s) from the first step takes major constraints into consideration and the second step also incorporates those minor constraints that have been ignored in the first step. The main task of the pre-processor sub-module is to find out which algorithms are suitable for the scheduling problem at hand.

Framinan and Ruiz (2010) further claim that the “non-essential constraints” could be separated from the schedule generation process and used when the actual schedule is constructed. This is due to the fact that the architecture’s purpose is mainly to use mathematical optimisation methods and meta-heuristics and there is no support for discrete-event simulation. To exclude some “non-essential” constraints is common when building a discrete-event simulation model, but a simulation model may include many more constraints. In fact, the effect of excluding different constraints can be tested in the simulation model by some validation and sensitivity tests.

Li et al., (2012) propose a new modular design for a simulation-based scheduling system for semiconductor manufacturing lines. The architecture is divided into a software layer, a simulation layer, and a data layer. They use a modular approach in which simulation models, algorithms, etc., are divided into different modules. Both predictive (dispatching rules) and reactive algorithms are used, but the main focus is on the automatic generation of simulation models based on the integrated and modular approach of simulation data. Furthermore, there is no information whether on-line reactive scheduling is possible, when integrated into a real-world system. A similar approach of automatically generating discrete-event simulation models is proposed by Horn et al., (2006). A successful implementation in a real-world system employed a five-step, simulation-based optimisation procedure using different heuristics, allowing the user to modify the schedule and finally generating a detailed operational plan, i.e., a Gantt chart. Sivakumar and Gupta (2006) propose an “implementation concept” for another similar system using the automatic generation of simulation models. They state that a simulation model would require much maintenance, as the circumstances change if the model itself is not generated automatically. The system includes the generation of schedules in a predictive-reactive manner, but the output reports produced appear to be static, i.e., not updated until a rescheduling is carried out. The system was implemented at a real-world facility and it allows the user to use both the “what-if” scenario experiments and the scheduling function.

Son et al., (2003) describe the structure and architecture of a simulation-based real-time shop floor control system for discrete part manufacturing. Discrete-event simulation models in ARENA are automatically generated, by using a model generator and a resource model, i.e., the database in MS Access 97, and a Message-based Part State Graph (MPSG) shop level execution model. Most of the software tool has been developed in VBA. The control system can be used for either flow shops or job shops and its purpose is to work on automatic systems, but it may also operate at manual workstations, as long as feedback is sent back to the system. The simulation communicates with a shop level executor and interacts with different external databases, such as master production schedule. The scheduling function is similar to other hybrid methods (e.g., Matsuura et al., 1993; Barua et al., 2005). A commercial scheduler was used to find good schedules without employing any simulation-based optimisation technique. Each resource, e.g., machine to be scheduled, is associated with a dispatch list which is a sequence of jobs to be scheduled in the order which is to be kept, but the simulation model is able to run in a FCFS mode as well. The real- time simulation is used as the central controller that keeps track of the current status of the system and sends required messages. The simulation sends messages to the lower level controllers and then receives feedback from the system.

3.4 Identified functions to include in a system architecture

Following the comprehensive literature review presented in chapters 2 and 3, a complete list of the necessary functions that can be included in the architecture of a scheduling system capable of handling real-world production scheduling problems is provided, in Table 3.1.

Table 3.1 The main areas of modular scheduling system architecture

Identified main areas	Identified functions
Discrete-event simulation (DES)	DES for complex problems (Laguna and Marti, 2003): multiple constraints (Ribas et al., 2010), uniform or unrelated machines (Ribas et al., 2010) multiple objectives (Gary et al., 1995). Automatic model generation (Sivakumar and Gupta, 2006). Model properties: Flexibility, Speed, Details (Koh et al., 1996).
Simulation-based optimisation	Genetic Algorithms (GA) + DES (April et al., 2003). Steady state GA (Rogers and Prugel-Bennet, 1999) and parallel evaluations (Li and Wang, 2008).
Scheduling	Schedule representation: Global scheduling (direct) (Sankar et al., 2003; Kim et. al., 2007), Dispatching rules (Baker and Trietsch, 2009) and other heuristics (indirect) (McKay and Wiers, 1999), Hybrid solutions (Roundy et al., 1991; Barua et al., 2005). Algorithm generator (Framinan and Ruiz, 2010). Automatic algorithm selection (Framinan and Ruiz, 2010). Meta-heuristics (Laguna and Marti, 2003). Hyper-heuristics (Burke at al., 2003). Setup-time reduction (Allahverdi et al., 2008). Flexible and configurable algorithms (McKay et al., 2002). Adaptive and learning by algorithms (McKay et al., 2002).

Dispatching (on-line)	List of jobs to be dispatched (Son et al., 2003). Switching (Matsuura et al., 1993) Priority index hybrids (Roundy et al., 1991; Barua et al., 2005)
Rescheduling	Rescheduling policies: Periodic rescheduling, Event-driven rescheduling, Hybrid rescheduling (Church and Uzsoy, 1992; Herrmann, 2006) Rescheduling methods: Robust schedules , Complete regeneration, Right-shift scheduling, Match-up scheduling (Viera et al., 2003).
Scheduling algorithms	Scheduling algorithms library (Framinan and Ruiz, 2010).
Objectives	Multiple objectives (Gary et al., 1995). Flexible objectives (McKay et al., 1999).
Experimentation module	Validation experiments (Kempf et al., 2000). What-if scenarios (Framinan and Ruiz, 2010; Sivakumar and Gupta, 2006).
Integration with other systems	Integration and import/export (Pinedo, 2005; Framinan and Ruiz, 2010)
Database	Input data (Framinan and Ruiz, 2010). Output data (Framinan and Ruiz, 2010). Monitoring data (Framinan and Ruiz, 2010).
Fault control	Data feasibility (Blazéwicz et al, 2001) and consistency (McKay and Wiers, 2003) check. Real-time dispatching fault control: Soft constraints, Hard constraints (Higgins, 1996)
User-interfaces	Model input data: Plant layout (Pinedo, 2005), Shop configuration (Framinan and Ruiz, 2010), Routing table (Pinedo, 2005), Product information (Framinan and Ruiz, 2010). Scheduling input data: Scheduling horizon (Stoop and Wiers, 1996), User interactivity (Higgins, 1996; McKay and Black, 2007), Resource calendar (Pinedo, 2005; McKay and Wiers, 1999), e.g., Planned maintenance, Machine repairs, Product samples. Scheduling output data: Capacity buckets (Pinedo, 2005), Gantt charts (McKay and Buzacott, 2000), Column editor (Pinedo et al., 1994), output reports (Kumar and Nottestad, 2006). On-line data: Schedule execution status, Production status, Performance measures (McKay et al., 2002), WIP levels (Pinedo, 2005), User interactivity (Higgins, 1996). Algorithm generator (Framinan and Ruiz, 2010).

3.5 Concluding remarks

As a general conclusion, the impact of uncertainty is systematically underestimated by academic research and a common approach to uncertainty is to react and reschedule (McKay and Wiers, 1999). Different methods with which to react and reschedule and create schedules that are robust or reactive to real-world disturbances have been identified in this chapter. However, solving the sequencing problems is not enough, since real-world schedulers are faced with day-to-day challenges. In order to handle the scheduling task including uncertainty, an integrated scheduling system and not only an intelligent algorithm is needed. This review has identified the most important functions and features that need to be handled by such an integrated scheduling system, such as simulation-based optimisation, flexible algorithms, system integration capability, within a modular architecture. Chapters 4 and 5 further address the internal details of such scheduling system architecture for handling most of the important functions identified.

Chapter 4

4 A Web Services-based Architecture for Industrial Scheduling

This chapter describes the overall system architecture of the Web services-based industrial scheduling system, which is designed to be a software architecture to solve the limitations of existing scheduling software used in industry. This architecture is based on the generic simulation-based optimisation platform, OPTIMISE, introduced in (Ng et al., 2008), and is customised and extended for industrial scheduling. Hence, the architecture is called OPTIMISE Scheduling System, or OSS. Since a Web services-based simulation system like OSS is closely related to Web-based simulation applications, this chapter begins with a brief introduction and literature review of Web-based simulation (Section 4.1), as well as some existing platforms found in the literature (Section 4.2). A short introduction of the OPTIMISE architecture is presented in Section 4.3, after which the chapter focuses on OSS and its core components (Section 4.4).

4.1 Web-based simulation

The internet has grown considerably in the last two decades and it is not only a platform for information sharing, but also for new applications within many different areas. Simulation applications have started using the concept of Web-based simulation (WBS) moving from more traditional local desktop solutions. Fishwick (1996) states that WBS "*represents the connection between the web and the field of simulation*", and Byrne et al., (2010) define WBS as "*the use of resources and technologies offered by the World-Wide-Web (WWW) for interaction with client and server modelling and simulation tools*". Compared to desktop systems, some advantages can be identified when a Web-based system approach is used:

- Accessibility: A Web-based system enables users at different locations to access the data from any computer that has internet available. Furthermore, a Web-based system is also accessible off-hours (Veith et al., 1999).
- Cross-platform capability: Such a solution is flexible because the web applications can be independent of computer type or operating system (Jin et al., 2010; Byrne et al., 2010).
- Controlled access: A Web-based system can use passwords and user-accounts to restrict the access of the system. (Veith et al., 1999)
- Licensing: The cost of simulation software and computer hardware can be high for a company (Fishwick, 1996), especially if there are requirements for parallel or distributed evaluations. Using a Web-based approach means licenses can be used when these are required from within a company or an external service provider. The total cost of simulation projects can be substantially reduced (Wiedemann, 2001).
- Maintenance: The maintenance is carried out on the server and the changes take effect without needing to involve actual client applications. (Byrne et al., 2010)

However, there are drawbacks to Web-based systems, some of which follow:

- Graphical user interface limitations: Interfaces supported by the web are limited (Suh, 2005), and it may require too much effort (Wiedemann, 2001) creating complex Web-based interfaces compared to desktop-based interfaces.
- Security vulnerability: Web-based applications are vulnerable to malicious Internet attacks (Suh, 2005).
- Licensing: Some software vendors may only allow a single place usage (Wiedemann, 2001).
- Network traffic delays: Luo et al., (2000) state that distributed simulation clients may take longer to execute compared to local simulation because of network traffic delays.

Byrne et al., (2010) claim that the research within WBS is still in its infancy and the number of real-world applications is still low. When it comes to Web-based SBO systems, there are only a few publications, which are briefly reviewed in the following sub-section.

4.2 Existing Web-based systems for SBO

Luo et al., (2000) describe a Web-based distributed SBO system that is based on Java. The system consists of three parts: a management console, a web server and central controller, and the simulation clients, see Figure 4.1 freely redrawn from Lou et al., (2000).

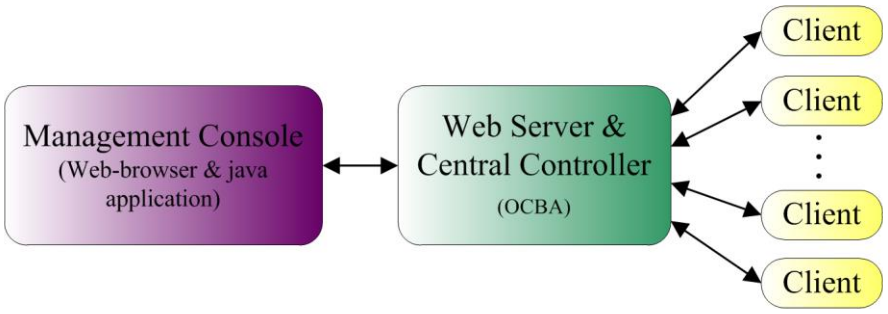

Management Console
(Web-browser & java application)
Web Server & Central Controller
(OCBA)
Client
Client
Client
Client

Figure 4.1 Web-based distributed simulation system.

A Web-browser is used to start a java application, i.e. the management console which is used to set up and start experiments for real-time monitoring of the clients and to show the simulation results of present and past optimisations. The web server and the central controller handle the assignment of jobs to be evaluated by the clients and use a sequential optimisation algorithm for Optimal Computing Budget Allocation (OCBA). The clients are the computing resources used for the simulation evaluations.

Another Web-based SBO system has been proposed by Yoo et al., (2009). Their framework for Web-based SBO uses a distributed platform, Parallel Replicated Discrete-Event Simulation (PRDES), to execute the simulation evaluations, see Figure 4.2 freely redrawn from Yoo et al., (2009).

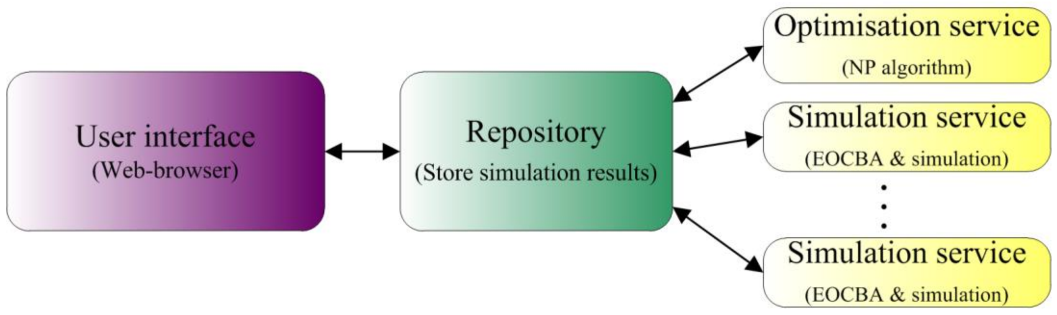


Figure 4.2 Web-based SBO framework.

A Web-page is used as a user interface, making it accessible through a Web-browser. The optimisation service uses an optimisation engine based on an NP-algorithm and, when a promising solution is found, it is sent to the simulation service through the repository. The simulation service is not entirely an evaluative client, since it uses a variant of the OCBA algorithm called EOCBA, which takes the computing power into account as well. The results are stored in the repository, i.e., database.

4.3 OPTIMISE: A web services-based SBO platform

OPTIMISE (OPTIMisation using Intelligent Simulation and Experimentation) is conceived as a generic Internet computing platform that tightly integrates different Discrete-Event Simulation (DES) systems with Artificial Intelligence-based optimisation tools in a Web services-based platform that can be integrated with other industrial/business information systems for valid simulation and optimisation runs (Ng et al., 2008). By generic, it is designed to be a computing platform that can be used to: (1) address a wide range of real-world optimisation problems commonly found in manufacturing and logistic applications; (2) facilitate the combined use of various search algorithms (e.g., Genetic Algorithms (GA) and local search); (3) be able to connect to different types of simulators and Discrete-Event Simulation (DES) packages through the Sim-Agent concept (see Figure 4.4), and (4) support inherently parallel and distributed simulation to significantly reduce the time spent on simulation evaluations. The platform is designed to be multi-tier client/server based in which all complex components, including various meta-heuristic search algorithms, neural network-based meta-models, deterministic/stochastic simulation systems and the corresponding database management system are integrated in a parallel and distributed platform and made available to general users for easy access, anytime, anywhere, through Web Services technology (Ng et al., 2007).

Even though the term cloud computing was not in popular use when OPTIMISE started to be implemented in 2006, it actually bears many common features that a cloud infrastructure should provide. Particularly, the concept of dual parallelism in cloud computing is supported by OPTIMISE, because it supports (1) multiple users from several companies/institutions that may be geographically distributed; and (2) running multiple simulations for different simulation models developed using various simulation languages/packages.

To support these goals, OPTIMISE was designed with the following important features:

- Web services: using a Web page inside a Web browser as user interface, which Yoo et al., (2009) adopt, is advantageous with regard to accessibility and cross-platform capability. Although, approaches using client applications, e.g., .NET-applications are supported as well.
- Distributed simulations: distributed simulations facilitate the simulation evaluations that are to be run in parallel on different computing nodes/cores, which is important to reduce the total execution time for SBO. Network traffic delays, identified by Luo et al., (2000), are believed not to cause any problems, due to the improvements in the network technologies over the years.
- Remote database – using a database to store optimisation results similar to Lou et al., (2000) supports the storage and access to experimental data.
- Modularity: The system is designed to be highly modular, since it would increase the possibility to expand the functionality and save development time (Pinedo, 2005).
- Security: The security is taken into account in order to avoid unauthorized access, e.g., by using security certificates.
- Users: Different user accounts are needed in order to handle users' privileges, so that an ordinary user does not have administrator privileges. Furthermore, it is important that multiple users can use the system at the same time.

- Research – to facilitate further research on SBO using hybrid search methods for real-time decision making and/or weekly/daily scheduling.

Resembling the system architecture commonly used for cloud computing, the OPTIMISE systems architecture (Figure 4.3) is composed of multiple server components (cloud components) communicating with each other over a loose coupling mechanism such as a messaging queue. OPTIMISE fulfils the definition of cloud computing, as it incorporates the ideas of virtualisation and distributed computing using Web services technologies. With the XML Web services platform, OPTIMISE can be deployed as a three-tier architecture that consists of the following three layers: 1) OPTIMISE client; 2) OPTIMISE server; 3) data sources. This is a highly flexible and scalable solution and the separation is intended to support industrial IT service providers in delivering and supporting both computing services and technical consultancies to a wide range of industries, national and global, from SMEs to multi-national enterprises.

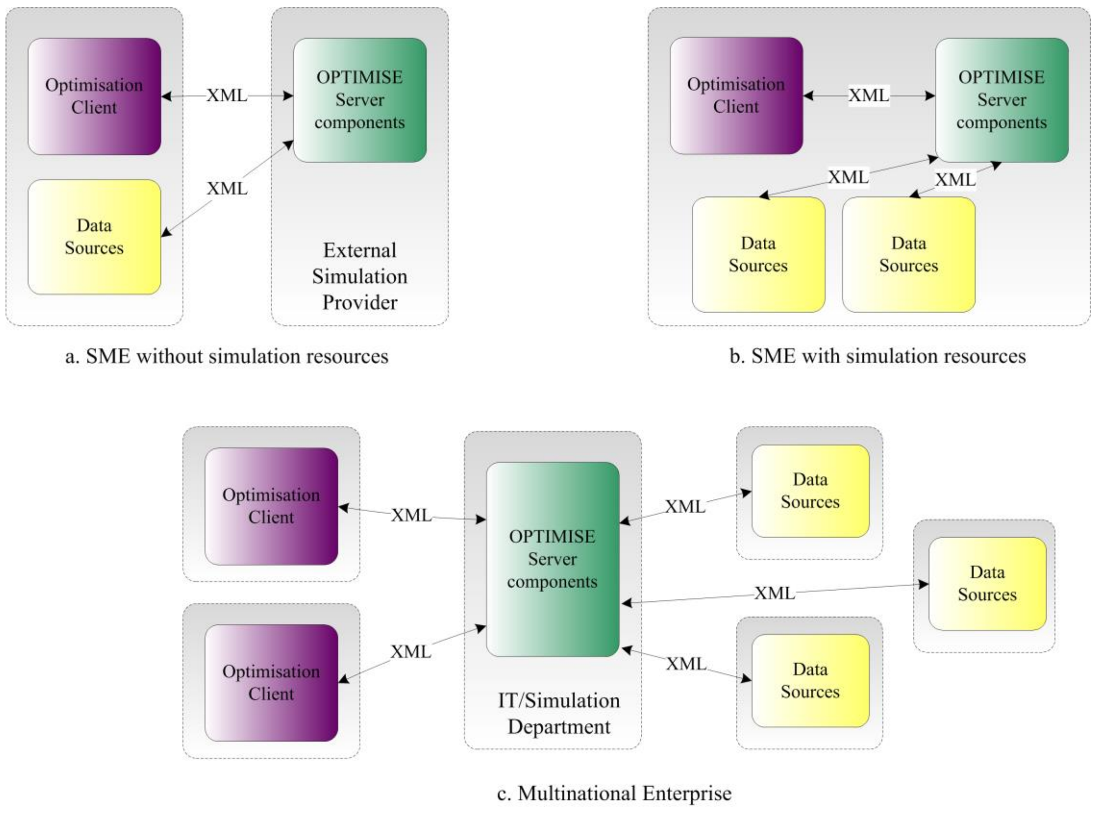


Figure 4.3 With XML Web services, OPTIMISE can be deployed with high flexibility and scalability.

For example, as illustrated in Figure 4.3a, a SME that does not possess its own simulation resources and required computing capacity can run the OPTIMISE client and data sources layer locally and connect to a remote OPTIMISE server that houses the DES systems and optimisation engines, by contracting an Optimisation Services Provider (OSP). The same kind of configuration can also be applied to a multi-national enterprise in which multiple OPTIMISE clients can connect to the optimisation services supplied by a central IT department, which acts as an internal OSP (Figure 4.3c).

As shown in Figure 4.4, the OPTIMISE architecture consists of a number of optimisation engines, surrounded by a set of OPTIMISE Server Components divided into three tiers: (1) Web Server; (2) Optimisation, and (3) Simulation subsystem. The optimisation engine (OptEngine) in the optimisation tier is the most important component for an SBO application, because it provides the core functionality for a optimisation/experiment and acts as the hub for coordinating other functions.

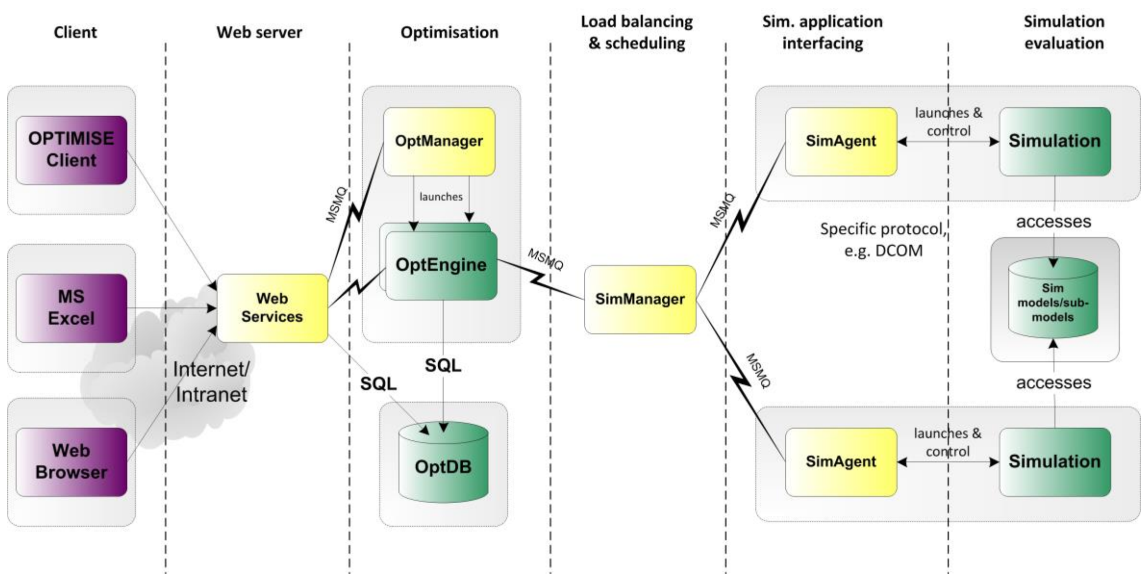


Figure 4.4 The generic OPTIMISE system architecture.

The web services function, hosted by the Webserver, listens to the XML requests from the client tier, such as start an SBO (through OptManager) or read data from the optimisation database (OptDB). The implementation of OPTIMISE started in 2006. While several extensions to support new technologies and applications have been made over the years, the core components have not been changed. Since these core components are used or extended in OSS, they are briefly introduced in this section and described in detail in the following sub-sections.

4.3.1 Optimisation manager and database

The Optimisation Manager (OptManager) is a Windows process that listens to the request from the Web Server to launch different OptEngines, according to the settings specified in the client applications. Data required to start an SBO procedure may include: (1) simulation settings (e.g., warm-up time, simulation horizon, number of replications and production line configuration), (2) objective function, (3) list of input variables, (4) list of output variables, (4) constraints to input variables, (5) choice of optimisation algorithm, and (6) optimisation parameters (e.g., population size, crossover rate, and stop criterion). Currently, OPTIMISE supports several optimisation algorithms, such as meta model-assisted hill climbing and evolutionary algorithms. However, meta model-assisted hill climbing algorithms are not used in the implementation of OSS (Chapter 6) or in the genetic algorithm (Chapter 5). Furthermore, new algorithms can be added easily, by compiling the modified algorithm core with the Object-Oriented libraries which OPTIMISE supplies. Generic algorithm software or templates needs research in its own area (Voß and Woodruff 2000). OPTIMISE has an Object-Oriented class library that allows new algorithms to inherit or override class methods for selection, crossover, and mutation operations which are commonly used in any evolutionary algorithms. There are also common function libraries for training meta-models, data normalisations, and communication with other components. These enable new algorithms to be quickly developed or customised and fit into the OPTIMISE framework by reuse. Such generic support of SBO algorithms' development and ease of launch during optimisation runs is a very important feature for the experiments in comparing different genetic representations (Chapter 5) and have generated the experiment results presented in Chapter 7.

By letting all *OptEngines* save their optimisation settings and other experiment results in a central database, i.e., OptDB, OPTIMISE supports the following features:

- Initial solutions, their quality and diversity, have a huge impact on the performance of an optimisation run, especially when a GA or other population based algorithms are used. All the experiment results are stored in the OptDB, to enable a user to choose the set of initial solutions from previous experiments, when starting a new optimisation run. This can also be used in combination with other experimental designs, e.g., Design of Experiments (DoE), provided in the OPTIMISE client applications. However, DoE is not used in the implementation of OSS (Chapter 6) or in the genetic algorithm (Chapter 5).
- Dynamical changes to meta-heuristic algorithms during the optimisation run are especially useful when global search methods, e.g., GAs, are used for exploration in a first stage followed by local search methods, e.g., hill-climbing algorithms, in order to further improve the optimisation result. However, local search methods are not used in the implementation of OSS (Chapter 6) or in the genetic algorithm (Chapter 5).
- Fault tolerance – Faults in a simulation evaluation can easily be detected and recovered by re-starting the run with another SimAgent using time-outs for the communication. If a simulation model returns invalid results, due to model deficiencies, it will be shown in OPTIMISE Browser, with which it is possible to browse new and historical optimisation data from OptDB. If the OptEngine crashes due to software faults, OPTIMISE indirectly facilitates error-recovery, by allowing a user to start an OptEngine and re-load the previous simulation records saved in OptDB.

4.3.2 Simulation components

Parallel simulation evaluations may be needed to speed up the SBO process. Therefore, the simulation components are located in a tier of their own, decoupled from the server components, to offer a modular solution that enables them to be widely distributed. Different simulation systems, e.g., commercial software or developed .Net applications, are connected to SimManager homogenously by using SimAgents in the SimAgent tier. To launch the simulation software used in a particular optimisation run, the SimAgents use the software specific BackEnd objects that support Distributed Component Object

Model (DCOM) and Socket communications for connecting to different simulation systems. Furthermore, BackEnd protocols are used to be able to communicate with the simulation software, e.g., load model, start simulation run, and collect output data. A standard format, XML, is used to return the output data via SimManager, to the OptEngine for evaluation and storage.

Unlike the SimManager described in Biles and Kleijnen (2005), which needs the software for the statistical methodology and optimisation techniques to be able to analyse the simulation results, the SimManager in OPTIMISE is a generic and light-weighted job dispatcher. Several SimAgents can be started at the same computer, depending on the computing capacity, i.e., number of processor cores. The SimManager registers all of the SimAgents that have been started, which means that it can dispatch several jobs received from OptEngines to multiple simulation systems running in parallel. The SimManager will send a job that is pending in the message queue to the first available SimAgents that fulfil the correct software requirements. The SimAgent will be marked as busy until the result is sent back to SimManager.

Any applications that use the Web services provided by OPTIMISE can be called an OPTIMISE client application. In order to supply the data needed to run SBO for the industrial scheduling problems, the GUI was extended to connect to the OPTIMISE Web services to launch SBO for industrial scheduling applications. On the other hand, there are some generic applications which have been developed for the monitoring/control of the OPTIMISE Server Components and management of optimisation project data. With generic, it means that they have not been specifically developed for a particular application. OPTIMISE Browser is useful for many optimisation projects and is an example of such a generic application. OPTIMISE Browser reads the data from OptDB, presents the data in tables and graphs, and can be used to analyse the data. How the OPTIMISE framework and client applications are customised for industrial production scheduling is the topic of the next section.

4.4 OPTIMISE Scheduling System (OSS)

The architecture of the OPTIMISE Scheduling System (OSS) (Frantzén et al., 2010; Frantzén et al., 2011) can be seen in Figure 4.5.

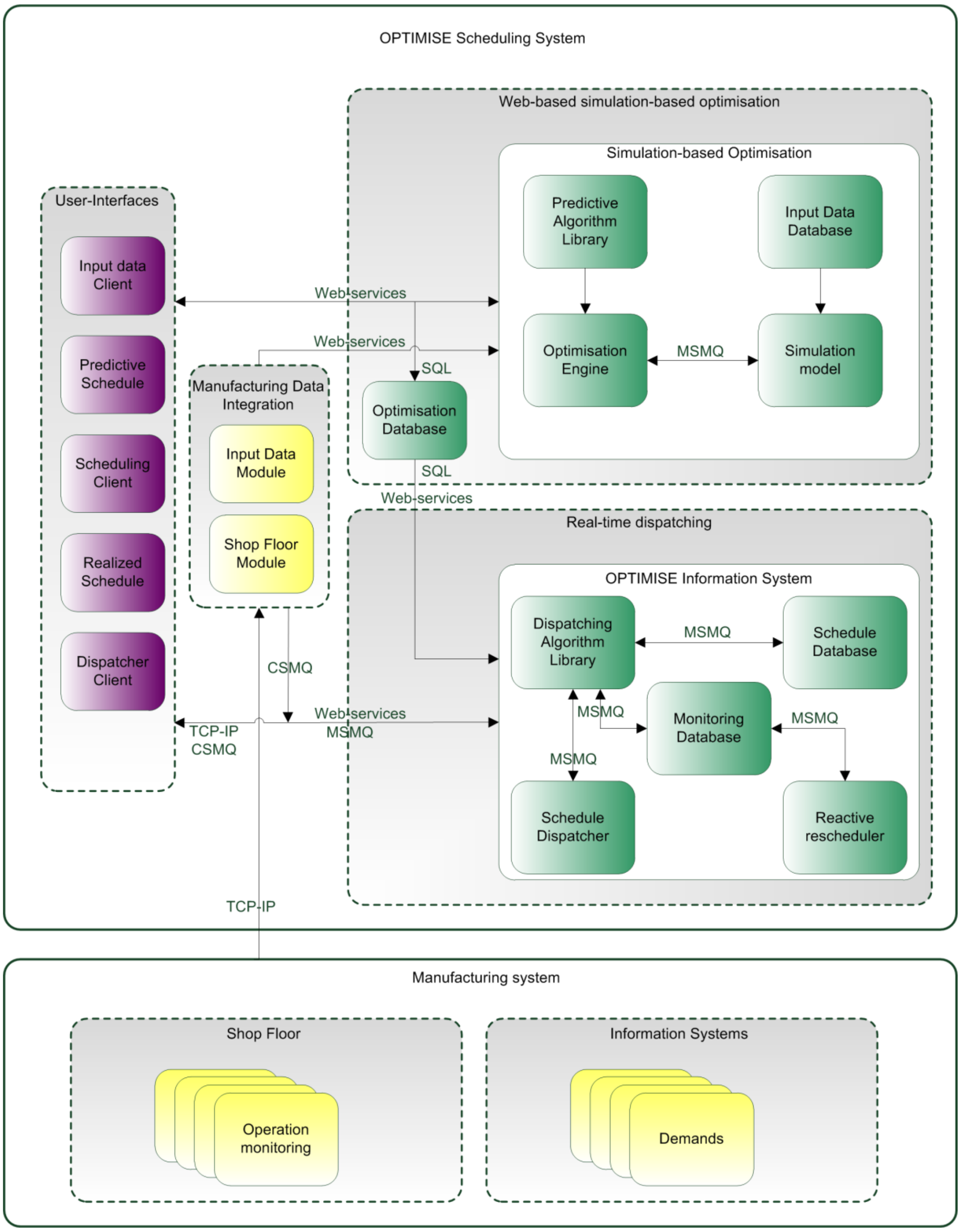


Figure 4.5 Architecture of OPTIMISE Scheduling System (see also Figure 4.6 for the information exchanges between the modules).

It is divided into two main parts, namely: (1) Web-based simulation optimisation (Andersson et al., 2007) and (2) Real-time dispatching. OSS is, in essence, a system implemented on top of the generic OPTIMISE platform to support near optimal and real-time scheduling with the help of SBO. OSS utilises the existing core components to support running the parallel and distributed simulation evaluations via the Internet. The connections and communications between the different parts in OSS are explained in detail in Section 4.4.3.

Apart from this, OPTIMISE directly offers the following advantages for the research advancement of OSS:

- Simulation models developed for a specific scheduling application, irrespective of the simulation languages/packages used for the development, can be connected seamlessly to OSS through the SimAgent technology (see Section 4.3.2).
- All the OSS scheduling and optimisation data is stored in OptDB, using the existing generic database structure. In other words, OSS inherits all the advantages offered by OptDB, in terms of the remote accessibility, security, fault tolerance, and flexibility, as described in Section 4.3.1.

On the other hand, OSS extends and customises the OPTIMISE platform, in order to support the real-time scheduling, identified as important in Chapter 3. In the following, the major OPTIMISE extensions, for the research advancement of OSS, are summarised:

- OSS adds a further layer of information management sub-system to the existing OptDB to specifically support reactive scheduling applications. This sub-system is called OPTIMISE Information System, or OIS hereafter. Unlike OptDB, OIS contains many data tables that store both shop floor and scheduling data which is related to other optimisation data in OptDB. Based on the real-time data collected from the shop floor and scheduling data derived from the optimisation data in OptDB, the Scheduling Dispatcher module can generate an expert proposed solution for the operator, when selecting the next job for an idle machine. This is done through dispatcher clients, which could be computer terminals on the shop floor or other devices like PDA (see Chapter 6), depending on the implementation of the Shop Floor Interfacing Module. Unlike other scheduling systems which can integrate with the company's Enterprise Resource Planning (ERP) system, the uniqueness of OIS is that the Scheduling Dispatcher processes the jobs with reference to the optimised schedules from OptDB.

- The Reactive Re-scheduler in OIS is a technique to tackle unforeseeable events, such as machine breakdowns and/or demand fluctuations in machine scheduling problems. This technique applies on-line data collection from the factory shop floor and on-line transient simulation with the same model used in SO to continuously monitor the production status. The near optimal plan is expected to yield a certain output performance. Also, there can be two problems: (i) the output is not met or (ii) the near optimal plan cannot be executed (e.g., major breakdown). In terms of machine scheduling, the Reactive Re-scheduler allows engineers to have high fidelity prediction and to determine when it is necessary to run re-scheduling, if a large deviation is anticipated. While this technique bears some similarity to other approaches, such as DES-based control (Smith et al., 1994) and on-line simulation (Davis, 1998), it is argued that such a reactive scheduling approach is novel in the sense that: (1) the performance index is determined by the deviation of the predicted performance from the near optimal plan generated from SBO and (2) integrated SBO support for short-term re-scheduling.
- OSS includes a specific module called Shop Floor Module which interfaces the data collection systems on the shop floor.

4.4.1 Scheduling functions and features included in OSS

OSS has a modular design and supports most of the functions and features that are presented in Chapter 3. How the following functions are supported by the architecture is briefly described for clarification:

- *Multiple constraints, uniform or unrelated machines, multiple objectives* are supported by a discrete event simulation model.
- The GA proposed in Chapter 5 supports: *Steady state GA, Meta-heuristics, Hyper-heuristics, Setup-time reduction.*
- *Parallel evaluations* are supported by the Web-based simulation optimisation when using the OPTIMISE platform.
- *Different schedule representations,* including *priority index hybrids,* are supported by the algorithm libraries.
- *List of jobs to be dispatched* is supported by the schedule database in OIS.
- *Different rescheduling policies* are supported by the reactive re-scheduler in OIS and manual rescheduling.
- *Robust schedules* are supported by a DES-model that includes stochastic, such as failure distributions, together with robust optimisation objectives.
- *Complete regeneration* is supported by the SBO.
- *Match-up scheduling* is supported by the reactive re-scheduler and realised through a schedule reconfiguration program.
- *Right-shift scheduling* is naturally supported by the list of jobs to be dispatched (direct representation), because it does not constrain the start time of a job.
- *Scheduling algorithms library* is supported by both the predictive and reactive libraries.
- *Multiple objectives* are easily handled by a DES-model.
- *Flexible objectives* can be handled by a DES-model with user-interface settings.
- *What-if scenarios* are supported by the OPTIMISE platform.
- *Integration and import/export* is supported by the manufacturing data integration modules.
- *Input data* is supported by the input data database.
- *Output data* is supported by the optimisation database, OptDB.
- *Monitoring data* is supported by the monitoring database.
- *Data feasibility* is supported by input data client and other user-interfaces.
- *Data consistency check* is supported by manufacturing data integration modules.
- *Real-time dispatching fault control* is supported by the schedule dispatcher.

- *Model input data and scheduling input data* is supported by the input data client, the input data database and schedule database.
- *Scheduling output data* is supported by the predictive schedule user-interface.
- *On-line data* is supported by the realised schedule user-interface and dispatcher client user-interface.

On the other hand, the following functions are only indirectly supported by OSS:

- *Automatic algorithm selection,* as proposed in Framinan and Ruiz, (2010), is partly supported through the use of hyper-heuristics (Burke at al., 2003) for both indirect and hybrid representation of schedules.
- *Adaptive, flexible and configurable algorithms* (McKay et al., 2002) are partly supported through the use of SBO, which is divided into two parts and can easily adapt to different types of problems.
- *Validation experiments* are partly supported by the OPTIMISE platform.

The following functions are not directly supported, mainly because they are outside the scope of this thesis:

- *Algorithm generator* (Framinan and Ruiz, 2010) is outside the scope of this research, although it would be possible to add it to the architecture. Simple heuristics, such as PDRs, could easily be automatically generated, but with regard to advanced algorithms, such as meta-heuristics, it could be very difficult.
- *Learning algorithms* (McKay et al., 2002) are indeed interesting, but are outside the scope of this thesis, due to the complex nature of real-world scheduling problems.
- *Capacity buckets* (Pinedo, 2005) is not considered part of the scheduling function, since it is more a planning activity connected to manpower planning.
- *Automatic model generation* has been identified as important by several authors (e.g., Li et al., 2012; Horn et al., 2006; Son et al., 2003). Sivakumar and Gupta (2006) assert that a simulation model would require much maintenance, as the circumstances change, if the model itself is not generated automatically. However, input data for real-world problems is rarely built directly into a

simulation model without using an external user-interface, such as Excel. Koh et al., (1996) describe that a model needs to be flexible enough to cope with changes in the physical configuration, fast enough so a schedule can be generated in an acceptable period of time, and detailed with a limited number of simplifications. Consequently, a model built for a complex real-world scheduling problem that satisfies these requirements would already be flexible enough. Generating a simulation model completely automatically for such a scheduling problem is probably difficult and may affect the speed of the model generated. However, an extensive review that maps general and specific properties of such scheduling problems resulting in a generic interface could be of great use, but is outside the scope of this work.

The Web services-based simulation optimisation is a part of OSS that is supported by the OPTIMISE platform. The real-time dispatching is an extension of OPTIMISE and is therefore described in the following sub-section.

4.4.2 OPTIMISE Information System

OPTIMISE Information System (OIS) holds the information that is relevant for the schedule execution. When a job is started or ended, the worker/operator uses the dispatcher client to report it and the information is sent to OIS, which keeps track of the WIP status and the schedule execution status. Job expert suggestions are sent to the dispatcher clients in real-time based on the current near optimal schedule.

The optimisation data that contains the near optimal schedule and scheduling scenario is sent to OIS by the scheduler, from using the scheduling program. The scheduling scenario contains all the information needed to start an optimisation, i.e., shift forms, variants processing steps, machine processing and setup times, and so forth. Some of this information is necessary for use in other programs, for example, “Update current status” of WIP uses the variant processing step information to update correct WIP area and PDRs might be using processing times, and so on. The near optimal schedule contains different sequences and/or PDRs that are selected. The implementation (see Chapter 6) of OIS (see Figure 4.5) consists of the following parts:

- Dispatching algorithm library
- Schedule database
- Monitoring database
- Schedule dispatcher
- Reactive re-scheduler

4.4.2.1 Dispatching algorithm library

In the dispatching algorithm library, it is possible to add different kinds of scheduling rules. Such rules can be direct, indirect, or hybrid scheduling rules, which handle the actual real-world dispatching at the workstations. The scheduling rules in the dispatching algorithm library will create the realised schedule and can differ from those rules used to create the predictive schedule. See Chapter 5 for further clarification on the different types of schedules and Chapter 6 for the actual implementation of different sorts of scheduling rules. In the algorithm library, the following types of representation rules can be added:

- Direct representations, such as permutation schedule and non-permutation schedule.
- Indirect representations first-come-first-served and other types of PDRs.
- Hybrid representations that mix direct- and indirect representation in the same rule.

4.4.2.2 Schedule database

The schedule database contains the current predictive schedule generated from the SBO. When a rescheduling has been carried out, i.e., when a new predictive schedule updates the current schedule, a schedule reconfiguration is carried out. The schedule reconfiguration is the process that brings the active schedule in line with the new schedule. The schedule reconfiguration program is part of the reactive re-scheduler and is executed after a rescheduling has been carried out in order to match the current state against the new schedule. When an optimisation is started, the status of WIP and schedule execution is used to create a scheduling scenario for optimisation. When the optimisation is finished, the new schedule will be based on the WIP status and the schedule execution status that was, in fact, correct information at the start of the optimisation, but may not necessarily still be true at the conclusion of the optimisation. Consequently, WIP changes during the optimisation need to be incorporated into the new schedule to make it valid. If necessary, the schedule reconfiguration program will take care of the activities that have occurred during this time period and merge them with the new schedule. This information is taken from the monitoring database.

4.4.2.3 Monitoring database

The monitoring database may contain information such as schedule execution and machine status. To keep track of the WIP status is one of the main tasks of the system. When a WIP message is sent to OIS, the status of WIP will be updated. The dispatcher clients could be programmed to enable them to report the start or end of jobs and to retrieve information about the next job to be started. The WIP of a production stage is the number of available jobs to be executed at the workstations belonging to that production stage. The schedule execution status monitors production fulfilment of the current schedule. In order to know which job to select next, the task of storing information is important so that the production rate fulfilment and additional Gantt chart for both visualisation and analyses can be monitored. The task of checking off jobs in each machine’s predictive sequence (list of jobs to be dispatched) is necessary, when a direct approach is used. Direct approach means following an exact sequence at the machines, while the indirect approach uses a set of PDRs for the local decision about the jobs to be started at the machines. When a job has been moved from one WIP area to another, this information will be stored with the job, so that it can be deleted, if necessary (i.e., revert job), to maintain a valid Gantt chart. The implementation of the monitoring database, described in Chapter 6, keeps track of two things only: (1) WIP status and (2) schedule execution status. This is due to a limitation regarding integration with the manufacturing systems.

4.4.2.4 Schedule dispatcher

The schedule dispatcher is the program that communicates with the dispatcher client when it asks for the next job in line to be scheduled at a specific workstation. The schedule dispatcher works as follows:

- Receives a request for the next job to dispatch to a specific workstation.
- The schedule dispatcher uses the designated scheduling rule (direct, indirect or hybrid) for the workstation together with WIP status and schedule execution status to determine the next job to be dispatched.
- Information is sent back to the dispatcher client to determine whether the next job is available or not.

4.4.2.5 Reactive re-scheduler

The reactive re-scheduler is the module that initiates not only a complete rescheduling, but also a partial rescheduling. A complete rescheduling can be initiated periodically, due to periodic events such as demand forecast updates. The event-driven rescheduling is initiated when a certain event, such as machine failure, occurs. Since most real-world rescheduling policies are a hybrid of the two methods, (Herrmann, 2006) the schedule dispatcher would need to support both these policies. Some events may, in fact, require small changes to a schedule, i.e., when operators scrap parts, block parts, or send parts for rework. When this happens, the current schedule will automatically be out of date and, in order to cope with this, the match-up scheduling will be initiated to bring the current schedule back on the original course. The reactive re-scheduler also initiates a schedule reconfiguration program each time the predictive schedule is updated.

4.4.3 Scheduling operational steps and system communication

The detailed communication of the system architecture connected to the operational steps of the scheduling procedure is described in this subsection. In Figure 4.6, the “Scheduling steps” (left column) show the different steps necessary to realise the scheduling support of the proposed OPTIMISE Scheduling System. “Related system architecture functions", presented in the middle column of Figure 4.6, show what parts of the system architecture (Figure 4.5) are used in relation to the scheduling steps.

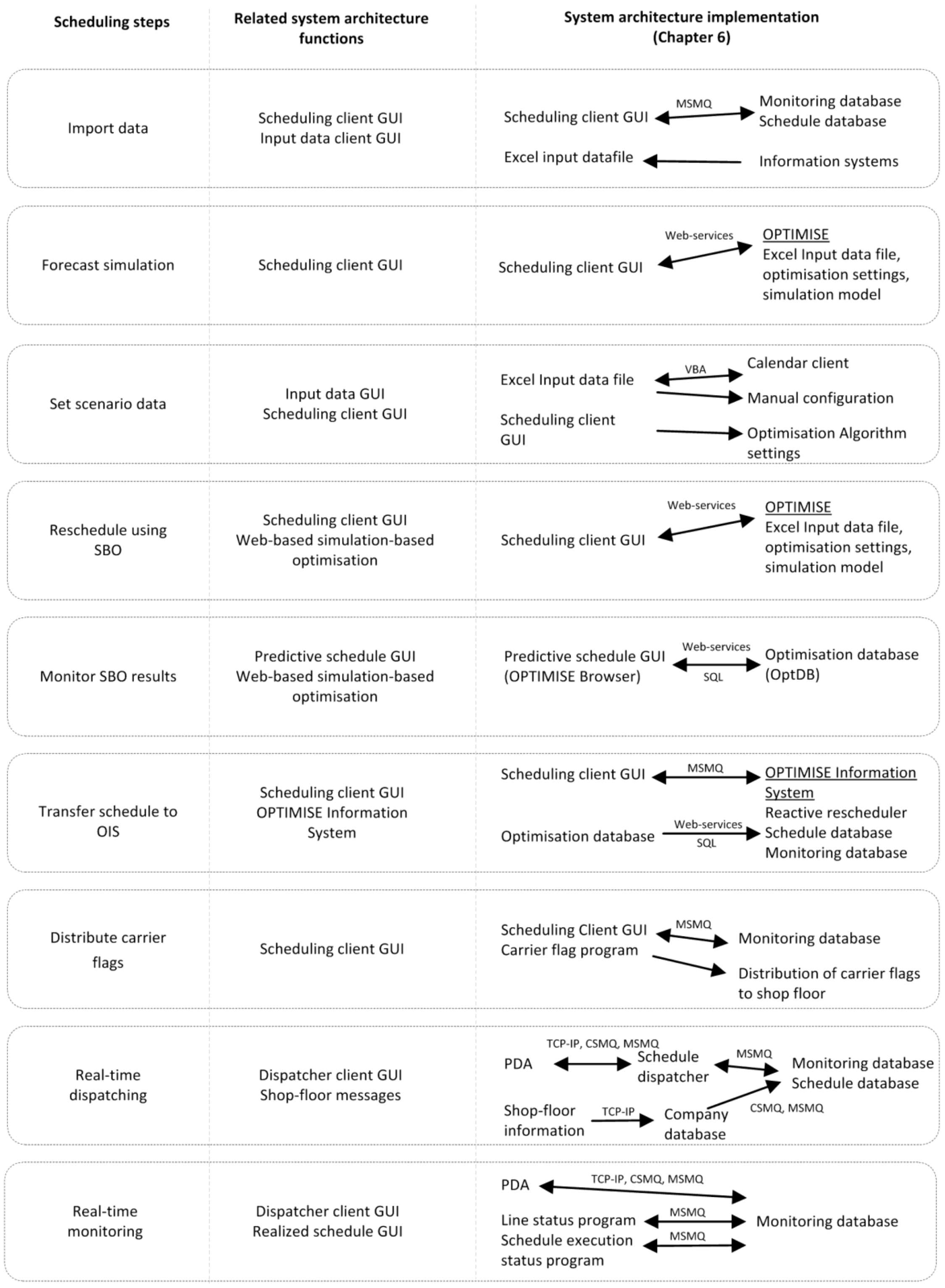


Figure 4.6 Operational steps and system communication (see also Figure 4.5)

“System architecture implementation” (right column) shows the detailed communication between the realised system architecture functions. Microsoft message queuing (MSMQ), internet communication (web-services), Visual Basic for Applications (VBA), Structured Query Language (SQL), Transmission Control Protocol/Internet Protocol (TCP-IP) and company specific message queuing (CSMQ) are used in different parts of the system to enable system integration and communication. The different steps of the scheduling procedure are as follows:

- Import data: Scheduling client graphical user-interface (GUI) imports current schedule from the schedule database in OIS and the current work in process (WIP) information from the monitoring database in OIS. The Excel input file is the realised input data client and it imports demand data from the information system. See Section 6.2.1 for further information.
- Forecast simulation: The OPTIMISE platform is used to run a simulation evaluation. The data sent by the scheduling client is the Excel input data file (see Section 6.3.1), the optimisation settings (see Section 6.3.2.1) and the simulation model (See Section 6.1.3).
- Set scenario data: Excel input data file imports calendar data from the Calendar client (See Section 6.3.1.1) and a manual configuration (see Section 6.3.1) is made to the file. If necessary, optimisation settings will be made (see Section 6.3.2.1).
- Reschedule using SBO: The OPTIMISE platform is used to run a simulation-based optimisation. The data sent by the scheduling client is the Excel input data file (see Section 6.3.1), the optimisation settings (see Section 6.3.2.1) and the simulation model (See Section 6.1.3).
- Monitor SBO results: Predictive schedule GUI, i.e., OPTIMISE Browser is used to monitor the SBO progress (see Section 6.4).
- Transfer schedule to OIS: The new schedule is sent from the optimisation database to the schedule database initiated by the scheduling client (see Section 6.3.2). When the new schedule has been matched-up with the current status of the system by the reactive rescheduler (see Section 4.4.2.5), a message is sent back to the user of the scheduling client.

- Distribute carrier flags: Carrier flags are printed using the carrier flag client (See Appendix A) which also marks the jobs as “ready” in the monitoring database. The carrier flags are physically distributed to the production line.
- Real-time dispatching: PDA (dispatching client) communicates with the schedule dispatcher (see Section 4.4.2.4) and shop floor information, i.e., parts finished to enter Finished Goods Inventory (see Section 6.2.2), updates or retrieves information from the monitoring database and the schedule database.
- Real-time monitoring: PDA can be used to obtain information about a job ID (see Appendix B), the line status program to monitor machine status (see Section 6.5.2) and schedule execution status program to monitor the schedule execution (see Section 6.5.1).

4.4.4 Rescheduling procedure

Schedulers have to deal with partial data, i.e., the situation at hand even if there is not a complete set of data available (See Section 3.2.1). Small changes to the schedule must be made continuously throughout the day and instead of a complete rescheduling, some sort of partial rescheduling (match-up scheduling and right-shift scheduling presented here) could possibly reduce the risk of schedule nervousness. Due to the uncertainties in the real-world, a rescheduling procedure was adopted based on the periodic- and event-driven rescheduling policy, i.e., hybrid rescheduling policy (see Section 3.1.3). The Rescheduling flowchart presented in Figure 4.7 shows the three different events, “Dispatcher message” (see Section 6.6), “Machine status update” (e.g., failure) and “Periodic rescheduling” (e.g., demand data), that may initiate a rescheduling. The implementation of the system in Chapter 6 includes the periodic rescheduling and event-driven rescheduling, although the event-driven rescheduling is not automatically carried out and is based on the manual decision of the two decision points: “Deviation too big?” and “Too many unscheduled jobs”. The schedule deviation is based on the comparison of actual dispatching times, i.e., the starting and stopping of jobs, with the current schedule in the schedule database in OIS. The deviation may be the result of the uncertainties in a flow shop, such as machine failures. The number of unscheduled jobs is increased when jobs have been measured and adjusted in a quality control and thereafter sent for rework. When jobs are sent to an upstream production stage for rework, the jobs will be flagged in the monitoring database so that they will be included in the next rescheduling. If there are too many jobs, they may constrain the other jobs by allocating carriers (see Section 6.1.1) that could be used for other jobs or are important to include in a new schedule (reschedule) because of their deadlines. When a rescheduling is to be carried out, the first step is to import necessary data and run a forecast simulation. The forecast simulation will indicate how the current schedule will perform if it continues with the same settings (possibly with new demand data) without performing a rescheduling (explained in detail in Section 6.7). Based on the results of the forecast simulation, the decision is made whether a rescheduling is necessary.

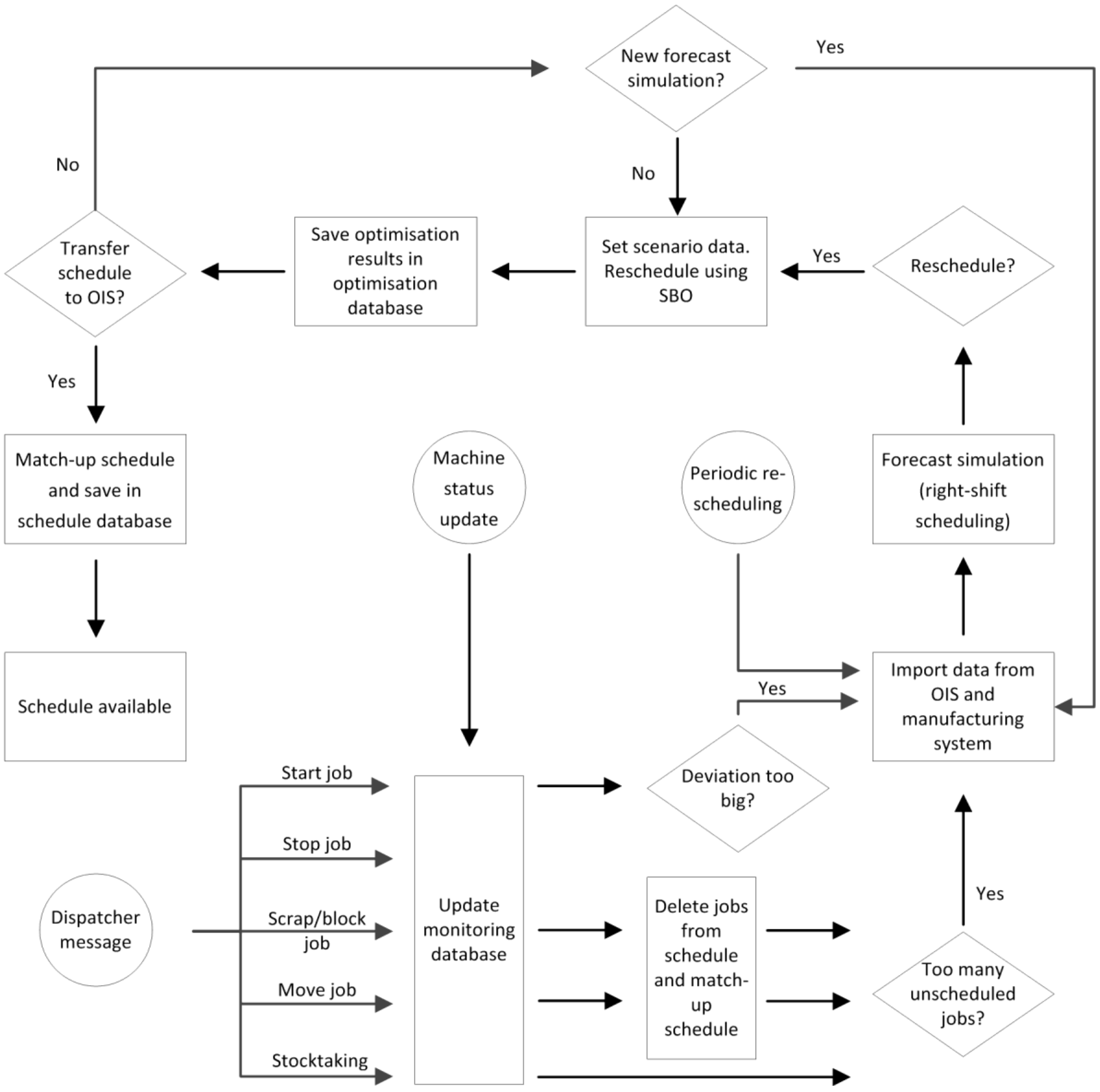

New forecast simulation?
Yes
No
No
Transfer schedule to OIS?
Save optimisation results in optimisation database
Set scenario data. Reschedule using SBO
Yes
Reschedule?
Yes
Match-up schedule and save in schedule database
Machine status update
Periodic re-scheduling
Forecast simulation (right-shift scheduling)
Schedule available
Yes
Import data from OIS and manufacturing system
Start job
Deviation too big?
Stop job
Yes
Dispatcher message
Scrap/block job
Update monitoring database
Delete jobs from schedule and match-up schedule
Too many unscheduled jobs?
Move job
Stocktaking

Figure 4.7 Rescheduling flowcart.

The simulation-based optimisation is started and when it is transferred (launched) to the shop floor it is matched-up (see Section 6.7) against the events that happened during the simulation-based optimisation run.

4.5 Concluding remarks

By adding new components, specific for industrial scheduling applications, to OPTIMISE, a new Web services-based industrial scheduling system called OSS has been designed in this study. OSS supports most of the important functions and features identified in Chapter 3. As reviewed in this chapter, OSS is unique compared to other Web-based SBO systems, as it has the capability to collect real-time data from the shop floor for reactive scheduling. On the other hand, since the optimisation algorithms can be developed independent of the applications, OSS can facilitate the evaluations of various optimisation algorithms. Specifically, a new genetic representation has been proposed in this study, which is presented in Chapter 5. Implemented on top of OPTIMISE, OSS is also designed to have a generic architecture. Components in OSS, particularly OIS and the client applications, can be customised for specific applications, which are shown in the full-scale industrial case study in Chapter 6.

Chapter 5

5 Hybrid Genetic Representations for Industrial Scheduling

It has been reviewed in Chapter 2 that SBO using GA is a promising approach to solve real-world scheduling problems, compared to classical scheduling methods. Nevertheless, because of the deficiencies of existing genetic representations in using GA, there is a need to propose some innovative representations to improve the performance of GA-based SBO. Therefore, this chapter includes a description of a hybrid genetic representation which is based on a mixture of dispatching rules and encoding the entire schedule. The chapter starts with an in-depth introduction to various representations used in scheduling. The design and implementation of the hybrid genetic representation into an SBO algorithm for handling various, real-world complex, hybrid flow shop scheduling problems is then addressed with details. Quantitative results from applying the hybrid representation to the full-scale industrial case study are provided in Chapter 7.

5.1 Predictive and realised schedule

Most scheduling research has considered the problem of finding an optimal or near optimal predictive schedule for various scheduling problems. The predictive schedule could be described as the forecasted “optimal” schedule, and when the predictive schedule is executed in the real-world the outcome is the realised schedule. There are different ways to find the predictive schedule, as described in Chapter 2. The way to find the schedules in the optimisation algorithm can be done at the lowest level, by handling the sequences directly, i.e., direct representation, or by using heuristics such as priority dispatching rules (PDR), i.e., indirect representation. With regard to scheduling using SBO, it is an iterative process in which the optimisation is separated from the simulation. The schedules found by the optimisation algorithm are tested in the simulation model, in order to estimate the result of the predictive schedules. The global schedules are often referred to as off-line schedules, since they may be executed strictly until completion, regardless of events that may occur, such as the arrival of a new job (Wan, 1995). However, a global schedule could be completely reactive if a continuous rescheduling policy is used, but the output of the algorithm is still a direct representation of a schedule. Furthermore, using dispatching rules is often referred to as an on-line, a real-time, or a completely reactive approach, which is true when the dispatching rules are used directly in the production line. However, when dispatching rules are used to create the predictive schedule in a simulation model before execution in a real production line, it is still an off-line schedule. Therefore, an algorithm that handles and creates sequences directly is referred to as having a direct representation of schedules and an algorithm that does not explicitly create sequences directly is referred to as having an indirect representation. Another dimension is whether the algorithm is used off-line or on-line (soft real-time). Hence, a predictive schedule may be created (in a simulation model) in the off-line mode, by using a direct or indirect representation, and the realised schedule is created by some sort of on-line dispatching that could be indirect (dispatching rule or other heuristic) or direct (predetermined sequences). In the following sub-sections, the direct and indirect schedule representations are introduced in more detail.

5.1.1 Indirect predictive schedule

We begin with an example to introduce the use of a simple PDR - First Come First Served (FCFS). Consider the Gantt-chart in Figure 5.1, which can be regarded as the output from a simulation model with two production stages.

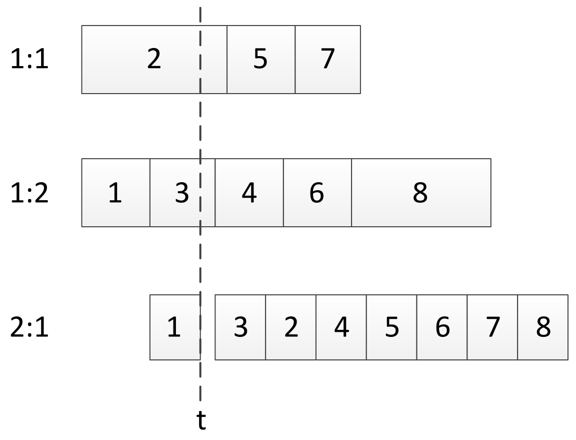


Figure 5.1 Gantt chart of a two stage production using FCFS.

Production stage one has two parallel machines and production stage two has one machine. 1:1 refers to production stage one and machine number one, 1:2 refers to production stage one and machine number two, and 2:1 refers to production stage two and machine number one. As mentioned, this simulated production line uses a dispatching rule that makes a decision based on local information at each machine. The area of interest here is production stage two which has to choose a job from those available after the completion of each job. When job one is finished, there will be no available job and therefore it must wait until job three arrives and starts that job. If several jobs are available after job completion, the PDR will make a choice depending on its setting, which in this case is the order they arrive in at production stage two. There are many different types of dispatching rules, and they may be classified as static or dynamic rules, global or local rules (Kiran, 1998). Static rules determine a priority value of a job that does not change over time, while dynamic rules determine priority values that change over time. Local or global rules refers to the data they use, i.e., whether it is global information (e.g., line status) or local information. As mentioned previously, the use of PDRs will be defined as an indirect schedule, since the actual sequence is not handled directly but rather the choice of dispatching rules.

5.1.2 Direct predictive schedule

A direct schedule is where each machine has a list of jobs it must process in a strict order and possibly also the start time. It is not allowed to deviate from this order and, if the machine is available but the next job on the list is not, it has to wait for the job to become available before it can continue. Figure 5.2 illustrates the same example as the previous one, but with the difference that a direct schedule is used where the jobs are in order from the lowest to the highest number, i.e., a permutation schedule.

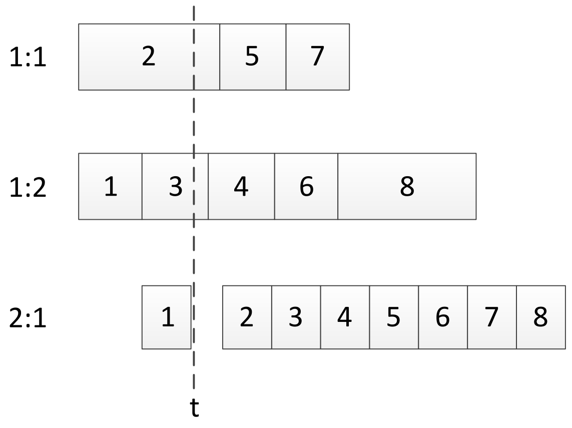


Figure 5.2 Gantt chart of a two stage production using direct schedule.

The difference with this method is that the schedule will wait for the next job according to the decided sequence. In machine 2:1, it waits for job two even though job three is available before job two. There are advantages and disadvantages with both methods. There is a risk that a direct schedule can create deadlocks in a model where a machine waits for a particular job which cannot move due to various constraints, for example.

5.1.3 Predictive schedule and sequence dependent setup times

Direct schedules are controlled and the sequence in each machine is known beforehand, while the indirect schedules generate the sequences as the simulation is executed. Since sequence-dependent setup times may have a large impact on the performance of a schedule (Allahverdi et al., 2008), the following example is similar to the previous one but with the difference that a sequence dependent setup time occurs in machine 2:1. There is a sequence dependent setup time between two groups of products, group A (jobs 1-2) and group B (jobs 3-8). Figure 5.3 shows the result of using FCFS.

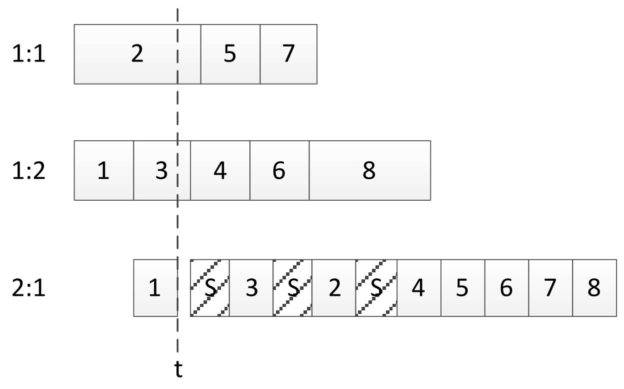


Figure 5.3 Gantt chart with setup times and FCFS.

The FCFS selects the jobs in the order they arrive in, which creates three sequence dependent setups. When the direct schedule is used, the result is different, see Figure 5.4.

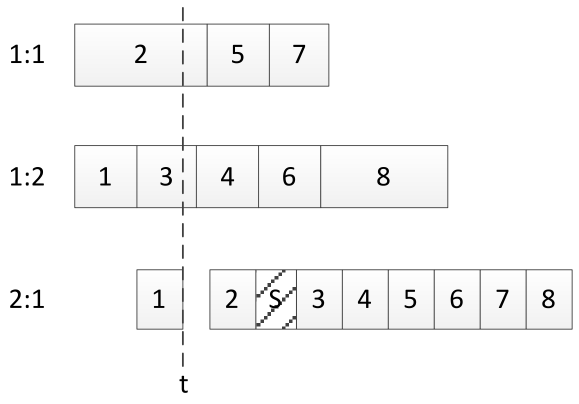


Figure 5.4 Gantt chart with setup times and direct schedule.

The direct schedule selects the jobs in the predetermined order, which creates only one sequence-dependent setup. Hence, in this example the direct approach is the better choice, compared to the previous example where the FCFS was the best one due to slightly better makespan. While the direct approach is able to avoid local sub-optimisations, the dispatching rules may utilise the machines more efficiently, and which method to use where is difficult to tell.

5.1.4 Schedule representation for predictive and realised schedules

The direct and indirect predictive schedules can be used to dispatch jobs on the shop floor. The following method, depicted in Table 5.1, can be used to explain the possible combinations of different schedule representations, when both the predictive- and realised schedules are considered.

Table 5.1 Predictive and realised schedules.

	Direct Schedule	**Indirect Schedule**			**Hybrid Schedule**	
Predictive	Sequences	PDRs			Sequences and PDRs	
Realised	Sequences	PDRs	PDRs and Sequences	Sequences	Sequences and PDRs	Sequences

The predictive refers to the representation used to create the predictive schedule and the realised refers to all the possible representations at the actual dispatching of jobs, resulting in the realised schedule. The different ways of creating schedules may also affect the representation used at the dispatching. Using a direct representation to create the predictive schedule will result in a direct schedule at the actual dispatching (direct-direct). When an indirect representation is used at the creation of the predictive schedule, the schedule can be represented by an indirect (indirect-indirect), hybrid (indirect-hybrid), or direct schedule (indirect-direct). In using a hybrid schedule to represent a schedule in order to create the predictive schedule means that the dispatching method may be a hybrid (hybrid-hybrid) or direct schedule (hybrid-direct). In a deterministic situation, the realised schedule will be the same, as long as the representation of the scheduling problem is valid. PDRs are adaptive to disturbances, since the actual schedule is created on-line, whilst the strict sequences may require a rescheduling in order to cope with those disturbances. There are, however, many different variants in order to cope with uncertainty, of which some are presented in Chapter 3.

5.1.5 Permutation versus non-permutation schedules

A permutation or non-permutation schedule is only relevant in the context of a multistage, production scheduling problem using a direct schedule at all production stages. A multistage scheduling problem is when multiple processing steps are needed to produce the products. Each stage can have its own sequence of jobs, separate from the other production stages. However, if the production stage sequence is the same for all the stages, it is called a permutation schedule. An example of a permutation schedule is displayed in Table 5.2.

Table 5.2 Example of a permutation schedule.

	Stage 1	Stage 2	Stage 3
First	1	1	1
Second	2	2	2
Third	3	3	3
Fourth	4	4	4
Fifth	5	5	5

However, if the sequence is changed on one or more stages, it is called a non-permutation schedule, e.g., as illustrated in Table 5.3. The reason for having non-permutation schedules may simply be that some jobs have a longer lead time through the production system or belong to a certain product family that may cause long sequence-dependent setups if it is intermixed with other product families. If an indirect schedule is used at any stage, the sequence cannot be guaranteed and it is therefore not meaningful to refer to that schedule as a permutation or non-permutation schedule.

Table 5.3 Example of a non-permutation schedule.

	Stage 1	Stage 2	Stage 3
First	1	2	2
Second	2	1	1
Third	3	4	3
Fourth	4	3	5
Fifth	5	5	4

5.2 Optimisation algorithm representation

The optimisation algorithm, parts of which originally presented by the author in Andersson et al., (2008), can be used on several real-world, complex hybrid flow shop scheduling problems, since its design is generic and deploying it is just a matter of configuration. The reason that the optimisation algorithm is able to be generic is that the scheduling problem itself is decoupled from the algorithm and handled by simulation models customised for the specific scheduling problems. In order to have an optimisation algorithm that manages to optimise a range of real-world, complex hybrid flow shop scheduling problems, data prerequisites are necessary. Furthermore, to be able to fully utilise the algorithm's potential, the following information is needed:

- The number of processing stages of the scheduling problem.
- The number of workstations available mapped to their processing stages.
- The number of product variants available and their production stages.
- Machine constraints: on/off, product variant constraints.
- Product variants' setup groups (product families).
- List of jobs to be scheduled.

The next sub-section describes the hybrid genetic representation, which is realised by the scheduling approach described thereafter.

5.2.1 Hybrid genetic representation

The optimisation algorithm handles various sizes of scheduling problems, since it divides each production stage into one unit that needs to be scheduled, see Figure 5.5.

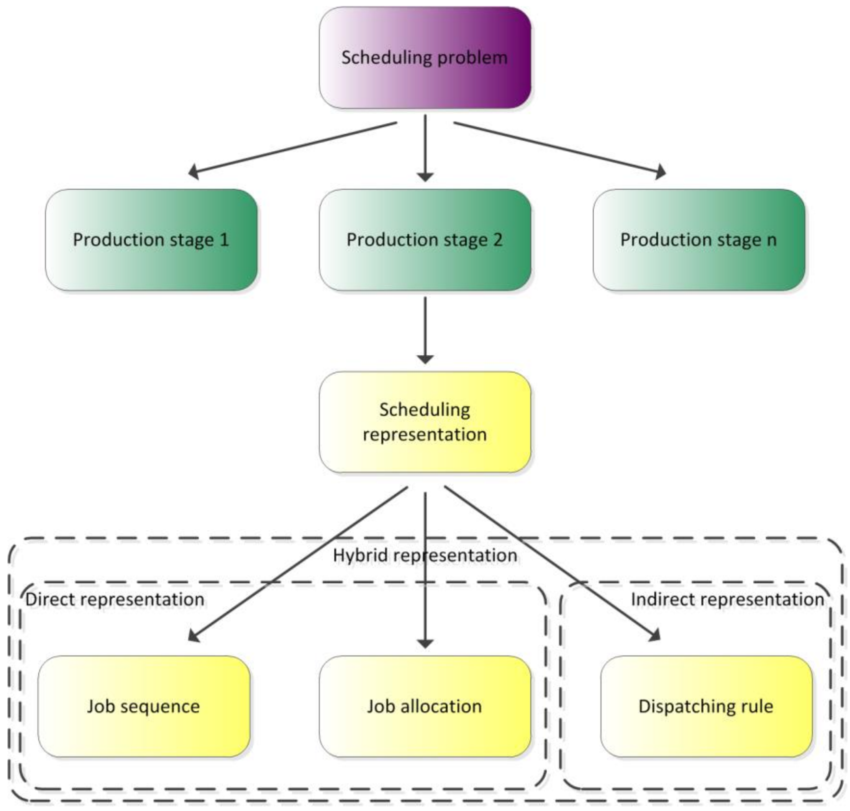


Figure 5.5 The hybrid genetic representation.

Each production stage has one of three scheduling representations: (1) Direct-, indirect-, or hybrid representation. Depending on the scheduling representation, a different representation of the scheduling at that stage will be made. There are three different ways that the algorithm represents the scheduling problem:

1. Job sequence representation that handles the sequencing at the production stage only.
2. Job allocation handling the selection of workstation/s at which the job will be produced.
3. Dispatching rules that handle both the job sequencing and job allocation dynamically.

The job sequence and job allocation are used in a direct representation and dispatching rules are used in an indirect representation. However, there are also hybrid dispatching rules that utilise all three representations, i.e., hybrid representation. Hybrid dispatching rules may have logic that makes dynamic decisions based on the job sequence, job allocation, and current situation on the production floor, e.g., Roundy et al., (1991).

5.2.2 Scheduling approaches in the GA

The optimisation algorithm handles different job sizes. For example, one job may be one part, one carrier of parts, several parts, or several carriers depending on the scheduling problem at hand. The GA encodes possible solutions as genomes and each genome instance represents a single solution to the problem – in this case, an operation schedule. In many applications, the efficiency of GAs is determined mainly on the problem structure and how the domain problem is encoded in the genome (Kiran, 1998; Talbi, 2009). The representation has therefore been carefully considered.

One of the contribution aims of this study is to propose a combination of the direct and indirect scheduling representations, so-called hybrid scheduling approaches. Simply put, a hybrid approach is when some stages use direct schedules and other stages use PDRs. Hybrid dispatching rules, where a dispatching rule uses some information about the direct schedule, i.e., hybrid representation, have been proposed by several authors (e.g., Matsuura et al., 1993; Roundy et al., 1991; Barua et al., 2005), but the use of a hybrid approach has not been found in the literature review. The schedule can be represented both directly and indirectly in the different production stages of this problem previously described. If several production stages are considered, the definition is either a direct-, indirect, or hybrid approach, as shown in Figure 5.6.

	Production stage 1	Production stage 2	Production stage n
Direct approach	Direct	Direct	Direct
Indirect approach	PDR	PDR	PDR
Hybrid approach	Direct	Direct	PDR

Figure 5.6 Direct, indirect, and hybrid approach.

The direct approach uses a direct representation of the schedules in all production stages and the indirect approach uses priority dispatching rules (PDR) in all stages. The hybrid approach has some kind of mix between direct and indirect representation in the different production stages. The indirect and hybrid approach may be classified as hyper-heuristics (Burke at al., 2003), since it is a meta-heuristics (GA) that selects other heuristics (PDRs).

5.2.2.1 Direct approach

The genome for this problem is designed to represent the schedule of the direct approach presented in Figure 5.7. The genome is implemented as a matrix in which each row corresponds to a specific job and each column represents a workstation. Each job is scheduled over parallel workstations in those production stages belonging to the process flow of the product type.

	Production stage 1				Production stage 2	Production stage n		
J1	1		▨	▨	1	1		
J2	▨	▨	1	1	▨		2	
J3	1		▨	▨	1			1
J4	▨	▨	1	2	▨	1	1	1

Figure 5.7 Direct representation of an operation schedule.

The example in Figure 5.7 shows job one to job four scheduled over three production stages using the direct approach. The jobs are to be allocated only in the white cells of the matrix and the striped areas are prohibited. The figure also illustrates that for this particular example the lowest job size is 1, but jobs are also allowed to be larger sets of 1 to create larger batches. Jobs one and three are processed in all production stages, whilst jobs two and four are not processed in production stage two. The task of the GA is to change the job sequence, i.e., the order of the jobs, and to fill in the solid cells of the matrix while considering some of the constraints, such as batch size. However, most of the difficult constraints, such as the required production stages of a product variant and a workstation’s constraints, are pre-processed in the initialisation of the optimisation run, in order to efficiently create valid solutions during the optimisation run.

It is possible to choose a permutation schedule or a non-permutation schedule for a multistage scheduling problem, when the direct approach is used. When a non-permutation schedule is used, the algorithm can also handle only scheduling some of all the production stages and letting the “non-scheduled” inherit the sequence from earlier stages, i.e., part permutation.

5.2.2.2 Indirect approach

No single PDR is likely to have high performance on a range of complex scheduling problems (Pierreval and Mebarki, 1997). Furthermore, the indirect approach is similar to that proposed by Barman (1997), but uses a combined GA with PDRs, which has been demonstrated to perform better than simple PDRs (Tanev et al., 2004; Ochoa et al., 2009). For this problem, the genome is designed to represent the schedule of the indirect approach presented in Figure 5.8. The genome is simple for the indirect approach where each column represents a production stage to which a dispatching rule needs to be assigned. The schedule, exact sequence of jobs, is created dynamically in the simulation. Each one of the workstations uses the dispatching rules and the task of the GA is to allocate the best dispatching rules to the production stages. It is not necessary to handle constraints in the optimisation algorithm, since the actual sequencing is carried out in the simulation.

Production stage 1	Production stage 2	Production stage n
SPT	EDD	FCFS

Figure 5.8 Indirect representation of an operation schedule.

5.2.2.3 Hybrid approach

It is also possible to use a hybrid that combines the direct and indirect approaches in a novel way. With the hybrid approach, at least one production stage uses a hybrid representation or at least one production stage uses the direct representation and at least one production stage uses the indirect. The example in Figure 5.9 shows Job 1 (J1) to Job 4 (J4) scheduled over three production stages using a hybrid approach. In production stages 1 and 2, the optimisation algorithm needs to allocate jobs to the workstations and optimise the job sequence. In production stage *n*, the optimisation algorithm optimises the selection of priority dispatching rules.

	Production stage 1				Production stage 2	Production stage n
J1	1		▨	▨	1	EDD
J2	▨	▨	1	1	▨	
J3	1		▨	▨	1	
J4	▨	▨	1	2	▨	

Figure 5.9 A hybrid of direct and indirect representation.

5.3 Optimisation algorithm steps

The optimisation algorithm is similar to a conventional genetic algorithm. However, it is not strictly population-based, but uses a steady state GA methodology which has been shown to be a more efficient way of utilizing distributed computational resources (Rogers and Prugel-Bennet, 1999). The difference is that a steady state GA uses non-stop optimisation and new solutions are generated continuously, compared to a conventional GA where a new population is generated after the completion of the previous, possibly causing a delay for distributed computing resources. The first two steps of the algorithm are to generate initial solutions and evaluate all of them. Thereafter, the actual optimisation steps take place where the population is continuously updated with new better solutions deleting the worst ones. See Figure 5.10 describing the algorithm steps.

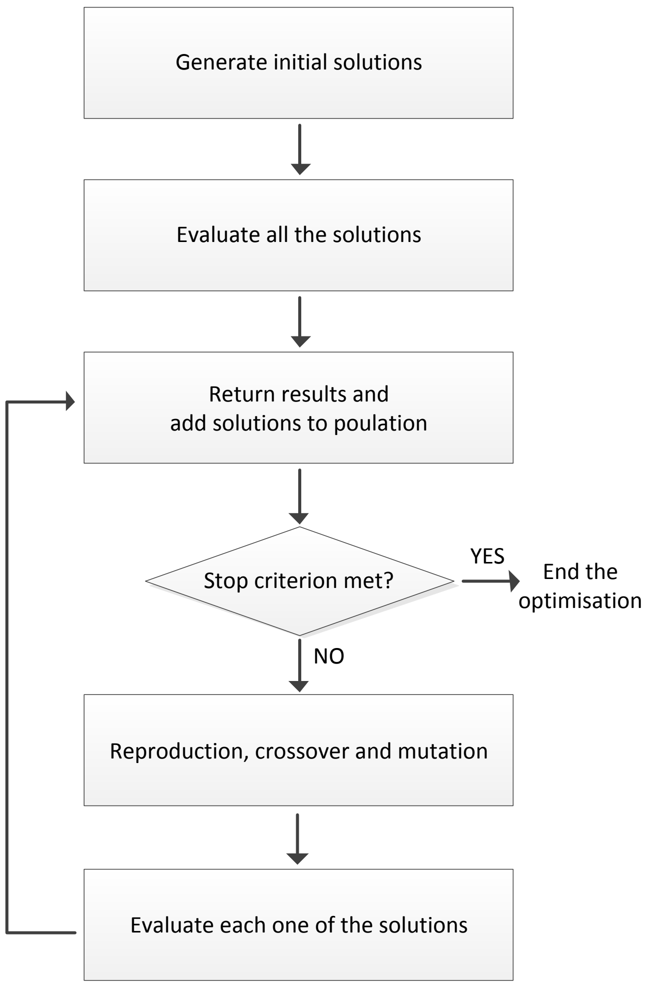

Generate initial solutions
Evaluate all the solutions
Return results and add solutions to poulation
Stop criterion met?
YES
End the optimisation
NO
Reproduction, crossover and mutation
Evaluate each one of the solutions

Figure 5.10 Flow chart of the optimisation algorithm steps.

The first step is to generate solutions until population size is met. After this step, all the initial solutions are evaluated in the simulation model and form the first population. The best solution is continuously updated and returned to the user in this step. Thereafter, a control is made whether the stop criterion has been met. If not, continue or stop the optimisation.

From the population of solutions, two pairs of solutions (four solutions) are randomly selected, see Figure 5.11.

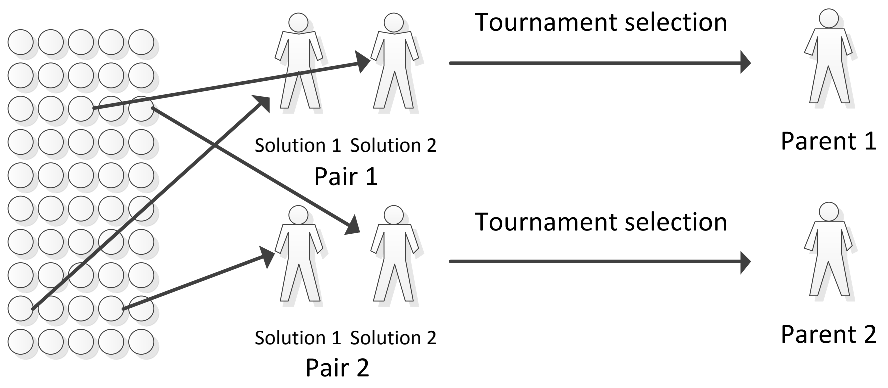


Figure 5.11 Reproduction in the optimisation algorithm.

A tournament selection for each pair is used to determine which ones will be used for reproduction, i.e., taking the better of two randomly chosen solutions. Thus, solutions with higher fitness values are more likely to be selected for mating; solutions with lower fitness values have some probability of being selected as well, in order to keep a large diversity of the population and to avoid premature convergence (Talbi, 2009). By using this method, only the weakest solution in the population will have no chance of being mated. The two remaining solutions will be the parents of one new solution (child). The next step is the crossover and mutation, see Figure 5.12.

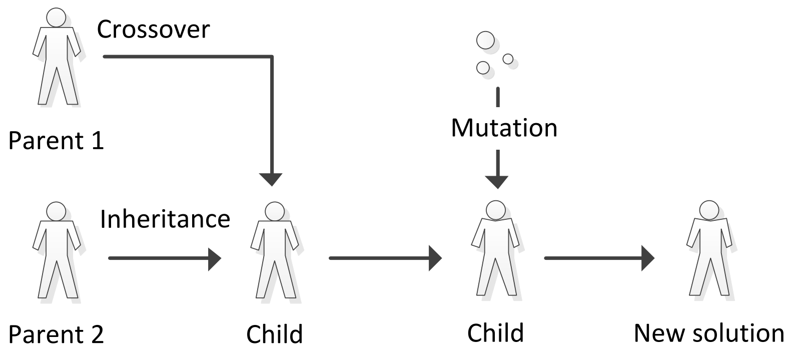


Figure 5.12 Mutation into new solution.

Parent two’s genetic material will be copied into this new child, onto which changes will be made. There is then the chance of a crossover from parent one. The crossover probability is 0.8 (p=0. 8). If a crossover is being made, there are three different ones, namely: sequence crossover (p=0.5), job allocation crossover (p=0.5), and dispatching rule crossover (p=1). The dispatching rule crossover will not compete against the stages that use direct approach and vice versa. Hence, there is a 100% chance of being selected for those stages. Only one child will be created from the two parents.

To maintain the genetic diversity from one generation to the next, the offspring solutions are mutated. The number of mutations is determined using a geometric distribution, where at least one mutation is always made. There are five different mutation types, namely: batch allocation, dispatching rule, single batch, variant group, and setup group mutation. All of these have the same chance of being selected.

Once the new solution has been created, it is added to a job list of solutions that are to be evaluated. The solution is evaluated in the simulation model and sent to the population, with its result.

5.4 Optimisation algorithm operators

In this section, different genetic operators (e.g., crossover and mutation) are described in detail, in order to gain a more thorough understanding of how the genetic operators of the different approaches are implemented. Different genetic operators are reviewed in Section 2.4.3.

5.4.1 Initialisation

A first population, currently set to 50 candidate solutions, is generated satisfying all the constraints. Each solution generated goes through the following steps:

1. Creates a new individual with no genetic material.
2. Generates the job sequence of the individual according to the following order: Production stage position, deadline, product variant number, and job ID number.
3. Randomises the job allocation and/or dispatching rule allocation.

This job sequence order is inherited by all the production stages using the direct approach. Each production stage is handled as one unit which could either have a direct or indirect representation. Production stages using the direct representation have their jobs randomly allocated for the allowed operations, i.e., satisfying the constraints. However, production stages using the indirect representation obtain randomised dispatching rules from the set of dispatching rules.

5.4.2 Job sequence crossover

The job sequence crossover operator used is the linear order crossover (lox) operator, because it keeps the relative order of the jobs (Liaw, 2000). A range of the parent jobs are selected, their sequence positions are copied and inherited by the child replacing the current positions of these jobs on all production stages of the child, as described in Chapter 2.

5.4.3 Job allocation crossover

A uniform crossover is used for the job allocation crossover. For each job, there is a probability of inheriting the job allocation from the parent.

5.4.4 Dispatching rule crossover

A uniform crossover is used for the dispatching rule crossover. For each job, there is a probability of inheriting the dispatching rule from the parent.

5.4.5 Job allocation mutation

The batch allocation mutation operator is only applied to work on machines that have the direct representation. In the mutation procedure, a job is first selected for allocation mutation within a production stage. Then one unit of this job is re-allocated to another parallel machine, provided that some other available machines exist within the chosen operation group.

5.4.6 Dispatching rule mutation

Dispatching rule mutation is carried out on those stages that have the indirect or hybrid representation. A random production stage is selected and its dispatching rule is exchanged for another randomised dispatching rule.

5.4.7 Sequence mutations

In order to prevent previously good solutions being cast into unfeasible regions of the search space, a domain-specific, directed mutation operator that follows the rules of the constraints can be used. A mutation operator using *iterative directed swap of adjacent groups* is applied, in which dynamically sized groups of jobs are selected. It is a type of *adjacent elements swap* (Bäck et al., 1997a) for groups of subsequent jobs in one direction, in an iterative manner. The groups could be one job, referred to as *single mutation operator*, a group of jobs of the same product variant, referred to as *variant group mutation operator*, or a group of jobs of the same product family, referred to as *setup group mutation operator*. In order to obtain valid schedules, a common stopping rule is also applied to the mutation operators, in which the two first mutation operators stop when they reach a job of the same product variant and the latter stops when it reaches a job of the same product family. Furthermore, permutation mutations for all succeeding production stages are carried out, when a non-permutation schedule is used to create valid schedules. It is not used in the permutation schedule, since the sequence order is automatically inherited by the other production stages. The mutation operators used generate valid solutions, can reach every solution of the search space, and make small changes, which Talbi (2009) identified as important issues when designing and using mutation operators. Furthermore, the mutation operators are believed to be able to accomplish smart mutations that do not become stuck in local optimas, due to an iterative process and the grouping of jobs. Quantitative results of the three different sequence mutations are compared in Chapter 7.

5.4.7.1 Single job sequence mutation

A single job is randomly selected and moved a number of steps in one direction. The single job sequence mutation is a variant of the variant group sequence mutation and, consequently, has the same implementation as that one except only one batch is moved.

5.4.7.2 Variant group sequence mutation

Grouping jobs of the same variant type is commonly used in industry, in order to reduce setup times and to make them easier to handle. The size of these groups may not be optimal and it may be a predetermined size that is not connected to the actual demand and the production line status. The variant group mutation has been made, in order to naturally group product variants of the same type, without determining the size of the batches beforehand. Thereby, these group sizes will be adapted to the situation at hand and may reduce the sequence dependent setup time. The single batch mutation can do the same thing, but probably not as efficiently as the variant group mutation. There is no risk that product variants are put into groups that are too large, since the groups can also be split to form smaller ones by the same mutation. Consequently, schedules at the same level as the single batch mutation can be created. The advantage of grouping jobs is that the mutation operator can do more in one mutation step and thereby avoid some of the really poor solutions. It may be better to move more than one job at a time, as moving only one job could result in more setups. Furthermore, it will make the schedules easier for the workers to follow, since it will indirectly generate schedules with fewer different shifts at the work stations. The variant group sequence mutation is only applied to the production stages using direct or hybrid representation. Figure 5.13 illustrates the different steps in grouping product variants and moving them.

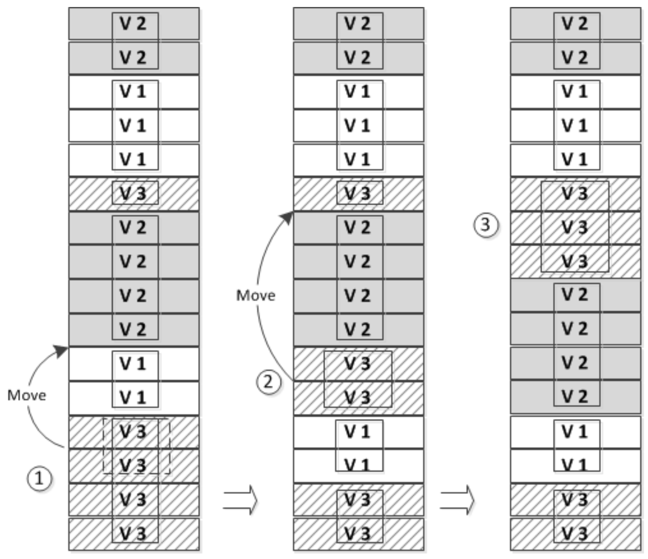


Figure 5.13 Variant group sequence mutation.

The procedure shown in Figure 5.13 is as follows:

1. First, one variant group is chosen randomly - in this case a group of four batches of V3 (Variant 3). There is a probability 0.8 (p=0.8) that the whole variant group will be moved. In addition, the example shows there is a possibility (p=0.2) of group splitting. A random job within the group is selected, after which a direction is randomised. In this example, the direction is up and, therefore, all jobs within the group from and above the selected job will form a group that is moved upwards.
2. From the group's newly obtained position, there is a chance of moving it even one more step, by a probability of 0.8 (p=0.8). In this example, another move is made.
3. All succeeding steps have a probability of 0.8 (p=0.8), unless the group reaches a group of the same variant type, in which case the moving procedure will stop and the two groups will be merged into one, as shown in this example.

Performing the same kind of move in the same mutation execution with the single job sequence mutation would be unlikely.

5.4.7.3 Setup group sequence mutation

Setup group sequence mutation is very similar to variant group sequence mutation, but larger groups are handled. The reason for using setup groups is to avoid those long sequence setup times that may occur in many production systems. The variant group sequence mutation can do the same thing, but not as efficiently as the setup group mutation. However, the setup group mutation cannot replace the variant group sequence mutation, since it does not work at that low level. Usually, a product variant belongs to a product family, e.g., material, that has the same characteristics, such as different variants of four cylindrical petrol engine blocks. A product family or a subset of a product family may form a setup group which does not have long sequence setup times internally, but longer sequence setup times externally outside that group of products in some or all of the production stages. By creating these groups, the algorithm may sort those products into larger batches, enabling more efficient and clever sequence reordering. To clarify the example of the procedure, there are three setup groups: (1) V1, V3, V5, (2) V2 and (3) V4, in Figure 5.14.

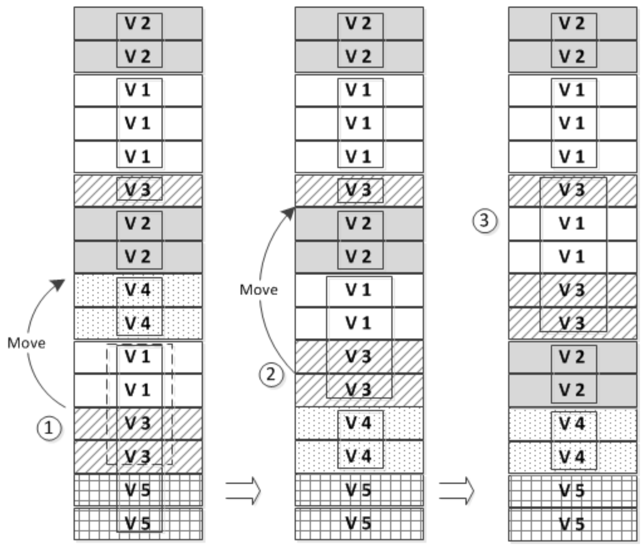


Figure 5.14 Setup group sequence mutation.

The procedure shown in Figure 5.14 is as follows:

- First, one setup group is chosen randomly - in this case, a set of three variant groups (V1, V3, V5) and a total of six jobs. This example shows a probability of 0.8 (p=0.8) that the whole setup group will be moved and a 0.2 (p=0.2) probability of group splitting. With regard to splitting, a random variant group within the setup group is selected, after which a direction is randomised. In this example, the direction is up and therefore all jobs within the group from and above the selected variant group will form a setup group that is moved upwards.
- From the newly obtained position of the group, there is a chance of moving it even one more step by a probability of 0.8 (p=0.8). In this example, another move is made.
- All succeeding steps have a probability of 0.8 (p=0.8), unless the setup group reaches a setup group of the same type, in which case the moving procedure will stop and the two groups will be merged into one, as shown in this example.

The setup group sequence mutation is a way of additionally making sequence changes, in order to avoid sequence dependent setup times. Compared to the variant group sequence mutation that is actually able to replace the single batch sequence mutation, the setup group sequence mutation cannot replace another sequence mutation, since it works at a higher level. The setup group sequence mutation may split setup groups, but never splits variant groups.

5.4.7.4 Generating non-permutation schedules efficiently

A permutation or a non-permutation schedule is only relevant in the context of a scheduling problem with multistage production using direct representation on all production stages. When the algorithm uses a non-permutation optimisation method, the scheduling problem becomes even greater, since every production stage may have its own sequence order. However, a crossover or mutation may or may not inherit changes to other production stages. Non-permutation schedules are particularly important, if some products do not need to be processed in all production stages or if sequence dependent setup times are considerable in some of the production stages. A setting will determine the probability of using a non-permutation, permutation mutation, or crossover.

5.4.7.4.1 Non-permutation sequence mutation and crossover

When a non-permutation sequence crossover or mutation is made on a production stage, the change is only applied to that stage. For example, if a specific product variant has a shorter deadline than the other product variants, because it is not being produced in production stages two, three and four, the non-permutation mutation may have a hard time prioritising jobs of that variant type efficiently. If the jobs of that product variant need to be prioritised on the last production stages of the production line, several mutations would need to be carried out, similar to the single job sequence mutation.

5.4.7.4.2 Permutation sequence mutation and crossover

When using a permutation sequence mutation or crossover, the change that is made at one production stage will be inherited by the succeeding stages. In the crossover, the position of the range of jobs is simply inherited by the other subsequent production stages. In the sequence mutation, the relative change (number of steps) is inherited by the other subsequent production stages. When using the permutation mutation and crossover, the scheduling problem is not constrained and all kind of schedules can still be created. The difference is that changes that will probably result in better schedules are made more efficiently.

5.5 Optimisation methods

Before running an optimisation, an optimisation method must be chosen. The method has predefined boundaries of how to run the optimisation with regard to the different scheduling approaches. Different optimisation methods are shown in Figure 5.15. However, the scheduling approaches are not limited to these methods.

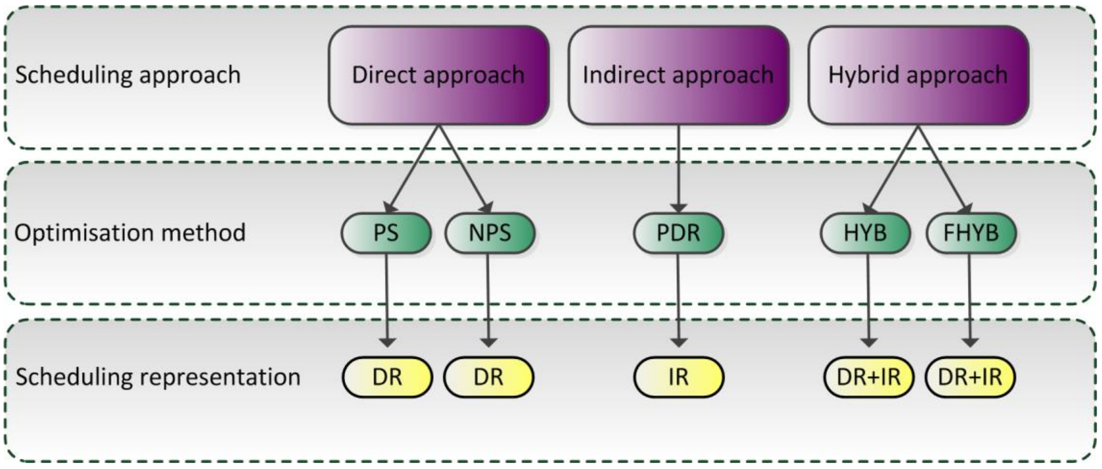


Figure 5.15 Optimisation methods.

The direct approach uses a direct representation (DR) of the schedules in all production stages and it is possible to use either a permutation optimisation (PS) method or a non-permutation optimisation (NPS) method. The indirect approach uses an indirect representation (IR), i.e., priority dispatching rules (PDR) in all stages, and the task of the optimisation is to select where the different rules are to be used. The hybrid approach has some kind of mix between direct and indirect representation in the different production stages. There are currently two optimisation methods within the hybrid approaches. The first, “HYB”, is the manual hybrid method originally presented by the author in Andersson et al., (2008), wherein the user selects which production stages will use direct, indirect or hybrid representation. The second method is called

“FHYB” (Andersson, 2011), which stands for “free hybrid”, since the optimisation itself selects which of the stages will use the one or the other, similar to the PDR method. These optimisation methods are evaluated in Chapter 7.

5.6 Concluding remarks

A dispatching rule-based approach (PDR) can be a very useful scheduling method, as it is time-efficient, but for highly complex scheduling problems or scenarios, a direct approach may be advantageous when searching for good solutions. However, in principle, a hybrid representation of these two approaches may offer their combined strengths. This study has proposed a generic, novel hybrid genetic representation for real-world, complex hybrid flow shop scheduling problems. Nevertheless, since hybrid flow shop is the focus of this study, whether the hybrid genetic representation can address other types of flows, such as job shops, is not considered. As reviewed in Chapter 2, to our best knowledge, such a hybrid GA representation is unique, as it has the capability to cope with direct, indirect- and hybrid approaches flexibly. In what way the optimisation is part of the generic scheduling system architecture is described in Chapter 4. Furthermore, the full-scale industrial implementation and qualitative evaluation of the optimisation system is presented in Chapter 6. Quantitative results from applying the hybrid representation to the full-scale industrial case study are given in Chapter 7.

Chapter 6

6 A Full-Scale Industrial Case Study

As mentioned in Chapter 1, one of the objectives in this study is to design a system framework with real-time and reactive support and then evaluate this framework qualitatively using a real-world industrial case study. A new and novel real-time industrial scheduling system using simulation-based optimisation technologies is proposed. To prove the system architecture, thus, testing the optimisation methods and techniques proposed in this thesis in a real-world industrial environment is crucial. In order to do this, a full-scale industrial case study was carried out in this work. Full-scale means the system and methods proposed were implemented in an industrial environment so that real data from the shop floor could be gathered and the optimisation methods could be tested and evaluated using the real production planning scenarios of the factory. Before the optimisation results and analyses are presented in Chapter 7, this chapter serves to address: 1) introduction of the targeted production line in the case study; 2) detailed information of how OSS was implemented for the production line, and 3) how the implemented OSS system was deployed and applied in a real-world setting.

6.1 The industrial case study

According to Yin (2003), one reason why a case-based research approach is appropriate is when contextual conditions have to be covered because it is believed that they are relevant to the phenomenon of study. The literature review has revealed that too many restrictive assumptions have been made (Kiran, 1998) and real-world problems are important future research (Jahangirian et al., 2010). When it comes to the case study as a research method, it is possible to use one or several case studies (Yin, 2007). However, in this study a single, full-scale industrial case study was more suitable, rather than testing different parts of such a system in several case studies. At the same time, it would be difficult replicating such a system implementation in several real-world case studies, due to the substantial work required for implementation (Pinedo, 2005), without making those unrealistic assumptions usually made in scheduling research. Wiers (1996) also points out that the implementation of proposed scheduling systems architecture would provide important information that could help scheduling techniques succeed in practice. This case study was selected to represent similar, real-world complex scheduling problems, and at the same time advance the research of manufacturing scheduling systems. Parts of this case study were originally presented by the author in Frantzén et al., (2011). The presentation of the case study is divided into three parts in order to describe different aspects of the industrial case study, namely: (1) the scheduling problem, (2) the production system, and (3) the simulation model.

6.1.1 The scheduling problem

The real-world problem considered in this full-scale industrial case study is a machining line at an automotive manufacturer in Sweden. Currently, the work of the production scheduler consists of making decisions about batching and the preferred start order of the carriers that transport the products. The dispatching is actually carried out by the production personnel on the shop floor: operators and shift leaders. Even though different jobs are prioritised in a specific order, the operators usually reschedule them to minimise the number of setups. The consequence of these manual decisions in the machining line is that while some machines might be locally optimised, the overall performance of the line is not. The scheduling problem could be categorised as a hybrid flow shop (see Section 2.3.1), described in Ribas, et al., (2010):

$$HF7\left(R5^1,0^2,0^3,R3^4,R3^5,R7^6,R3^7\right)\left|Multiobjective\right|Multiproperties \quad (6.1)$$

where, HF stands for a hybrid flow shop, 7 stands for seven production stages, R stands for unrelated parallel workstations and the corresponding number is the number of parallel workstations at each production stage. The 0 (zero) stands for a production stage with a single workstation and the term $Multiobjective$ means that the scheduling problem has multiple objectives. The case study represents a typical, real-world, complex multi-stage flow shop scheduling problem with the following features: machine eligibility, unavailability periods (failures, see Section 3.1.1), blocking (e.g., buffer limitations, see Section 2.3.2), machine dependent setup times, sequence dependent setup times, transportation times, bypass, lot splitting (see Section 2.3.3), lot sizing, and rework. The scheduling problem deals with the simultaneous lot sizing and scheduling problem, since the production is carried out with a make-to-stock policy. There are currently, in total, eleven production stages and more than ten product variants. The first production stage is not to be scheduled because it is a non-bottleneck production stage that manufactures product family components from raw materials, according to a PDR and inventory order point. In addition, the last three production stages are not to be scheduled because these are handled by the FCFS-PDR in the real-world system, due to practical circumstances. Consequently, there are seven production stages to be scheduled, although the last three need to be partially handled using FCFS. An assumption made for this scheduling problem is not to handle the operators at the production line directly, due to the substantial complexity of their work. The operators and their working hours are handled indirectly in the calendar client, which is part of the input data client. Furthermore, processing times are assumed to be constant and raw material is assumed to be available.

6.1.2 The real-world production system

The real-world production system is a machining line at an automotive manufacturer in Sweden. The production line could be defined as a mass production system (Groover, 2000), since about 8000-12000 parts are manufactured every week during a two-shift production schedule (7.30a.m.-3.48p.m. and 3.48 p.m.-12.00 midnight). The machining line has flow shop production with a product layout, i.e., the machines are arranged in sequence in order to carry out different parts of the processing/machining required. All product variants require processing through most of the ten different production stages, but some product variants bypass some of the production stages. This type of production is commonly found in automotive components manufacturing industries, such as Volvo Powertrain, Volvo Cars, Volkswagen, Vici Industri or Arkivator. The machining line is semi-automated with robots that feed machines inside the cells, but the loading, unloading and deciding when or where to process different types of parts is done by operators at each work area. The machine equipment is quite flexible, i.e., it can handle several different product variants, by changing tools. However, changing tools may result in long sequence-dependent setup times. Typical operations carried out in these production lines are milling, drilling, lathing, balancing, grinding, washing, and quality control. The product manufactured in this production system is a component found in a car or a truck engine. Typical products of this type are camshafts, crankshafts, or connecting rods.

The production line uses a make to stock policy (MTS) and has target levels that need to be reached each week. At the same time, there are security levels of the different product variants in the finished goods inventory (FGI). Based on a daily demand of the different product variants, there will be a lack of product variants, according to the security levels (about 1-2 days coverage time) or the target levels, sometime in the next few days. The task of the production planner is to make sure that the security levels are maintained over time and that the different target levels are reached each week. Consequently, the scheduling horizon is one week, but a two weeks horizon has been used in this study, as explained in Section 6.7. The different product variants can be sorted into their main groups (A or B), setup groups (used by the optimisation algorithm, see Section 5.4.7.3), or target level groups (used by the optimisation objective function, see Section 7.1.1).

Table 6.1 Product variants groups

Variant	**Group**	**Setup group**	**Target level group**
Variant 1	A	1	TL2
Variant 2	A	2	TL1
Variant 3	A	2	TL1
Variant 4	A	2	TL1
Variant 5	A	2	TL1
Variant 6	A	2	TL1
Variant 7	A	3	TL1
Variant 8	A	3	TL1
Variant 9	A	4	TL2
Variant 10	A	4	TL2
Variant 11	A	4	TL2
Variant 12	B	5	TL3
Variant 13	B	6	TL4
Variant 14	B	6	TL5
Variant 15	B	6	TL6
Variant 16	B	6	TL7

The product variants are transported on carriers in the production line, which will limit the total number of parts in the production line to a maximum of 6.000 parts. However, the target level of work in process (WIP) is set to 4350 parts (see Appendix G), so the number of parts in the production line is usually around that number. Each carrier may, at most, carry 50 parts of the same product variant. There are three different loops of carriers, i.e., between production stages 1-5, production stages 6-7 and production stages 8-10. The change of carriers is done at the end of each carrier loop, i.e., a free carrier of the next loop has to be available. The number of parts in different sections of the line is also constrained by the different buffer sizes. The lead times of the products are typically 24-35 hours, depending on product variant, current WIP, current product mix, failure of machines, and so forth. The actual total processing time through the line (value adding time) is about ten minutes. It is possible to prioritise a job, one carrier of parts, but since there are usually 50 parts on one carrier, it will still take about 8 hours to be produced. Different jobs (carriers with parts) of the same product variant or setup group will be grouped together over different production stages in order to reduce the sequence-dependent setup time (see Sections 5.4.7 and 7.2). The type of data usually changed through the input data client is described in Section 6.3.1 and the data used in the simulation model (based on real-world data) is presented in detail in Appendix G. The next Section describes the simulation model of this production system.

6.1.3 The simulation model

Due to the complexity of the problem, a detailed simulation model was implemented to model the existing production line. The discrete simulation model is currently built in C# (see Appendix H) to enable fast simulation runs (1-2 seconds on a 2.5 GHz Pentium core for two weeks simulation horizon), rather than commercial software that may take longer to execute. At the start of the simulation, the model is updated with the transient status, in order to obtain a start condition that matches the real production line. In the user interface for inputting data, the user has to define the specific scheduling scenario, since each production day/week is different. Hence, the simulation model has the flexibility to cope with changes in the production line. The simulation model and the optimisation can be set up to use different scheduling approaches: direct, indirect, and hybrid, as defined in Chapter 5. When using a direct approach, the input to the simulation model will be the sequences for the different machines and the output will be a feasible schedule. By using the indirect approach, the input to the simulation model will be different PDRs and the output to the shop floor will be a set of selected PDRs, a schedule, or a combination of both, depending on the specific scheduling scenario. In the hybrid approach, the input to the simulation model will both be sequences and PDRs and the output to the production floor is a schedule or a combination of schedule and the selected PDRs. The data used in the simulation model (based on real-world data) is presented in detail in Appendix G.

6.1.3.1 Structure of the simulation model

The simulation model is built with several production stages and each stage is similar to the next. The differences between each stage are the number of parallel workstations, machine constraints, and so forth. The constraint *blocking* is naturally used in the simulation model, due to the size of buffers and the number of carriers transporting the parts. Uncertainty is primarily modelled for failures, but external (indirect) variables, such as demand forecasts and scrapped parts, also generates uncertainty. The workstations have their individual settings and are classified as *unrelated parallel machines*. Furthermore, the workstation equipment is a combination of old and new equipment and changes to the product mix result in tool re-allocation. The simulation model easily adapts to the production variants of the different production stages needed. As illustrated in the variant production stages in Figure 6.1, there are currently five groups of variants when considering the necessary production stages. Not all variants go through all operations and thereby use the scheduling constraint or characteristic called *bypass*.

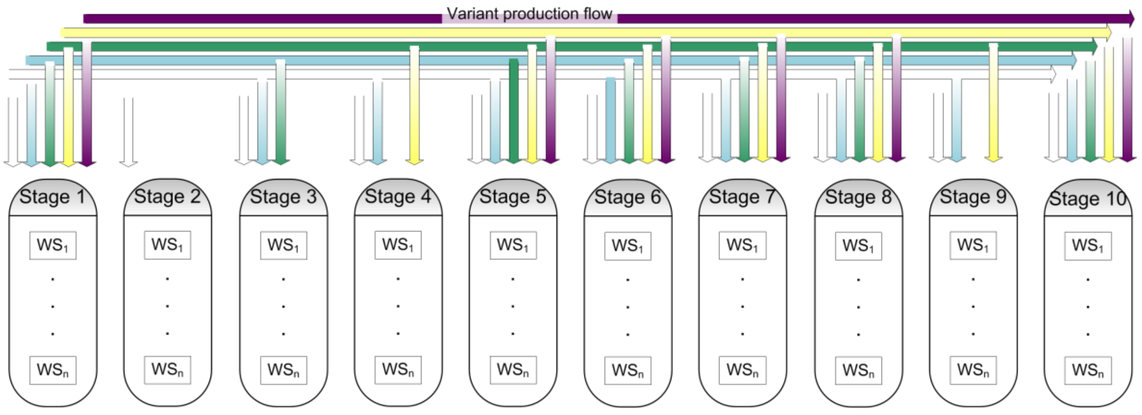


Figure 6.1 Production stages.

In Figure 6.1, each colour represents one group of product variants. Only one of the groups goes through all of the production stages. Furthermore, in each stage, there may be several parallel workstations. These groups are not equivalent to product families and may change as the production line changes, i.e., a product variant may change its necessary production stages if some processing operations are moved or if the line is re-balanced. Each stage consists of a production stage buffer, input workstation buffer, output workstation buffer, and the workstations themselves, see Figure 6.2.

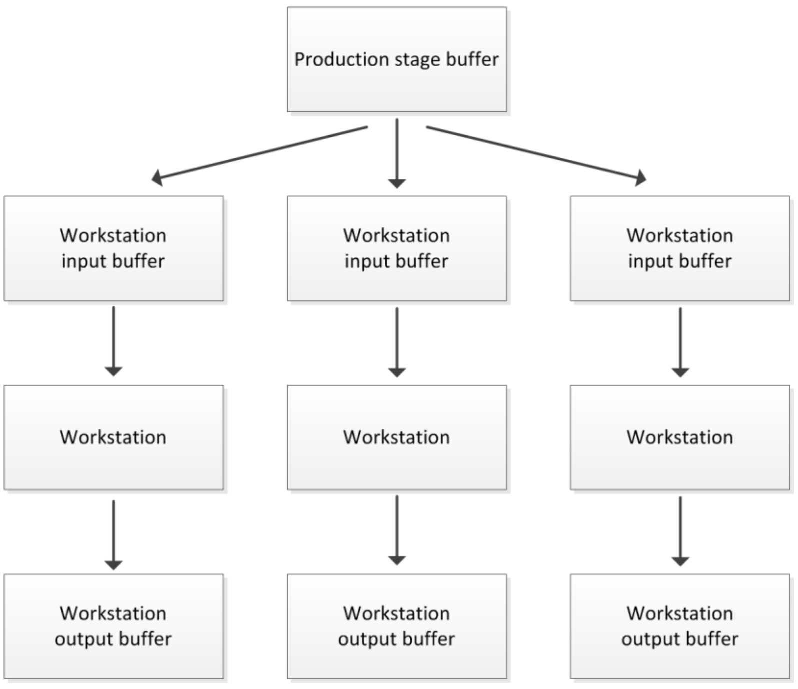


Figure 6.2 Structure of a production stage.

Each one of the buffers has its own settings for capacity and whether the buffer is used or not. For example, some stages may not have a production stage buffer, which can be used for intermediate stores or when several workstations flexibly share the input queue. Each workstation has its own settings such as sequence dependent setup times or machine dependent setup times. However, all these settings are extracted from the simulation model to the input data client (Excel file). The data used in the simulation model (based on real-world data) is presented in detail in Appendix G.

6.2 The OSS implementation and integration overview

The functions, described in Section 4.4.1, of the implemented OSS are described in this sub-section and the whole system is illustrated in Figure 6.3.

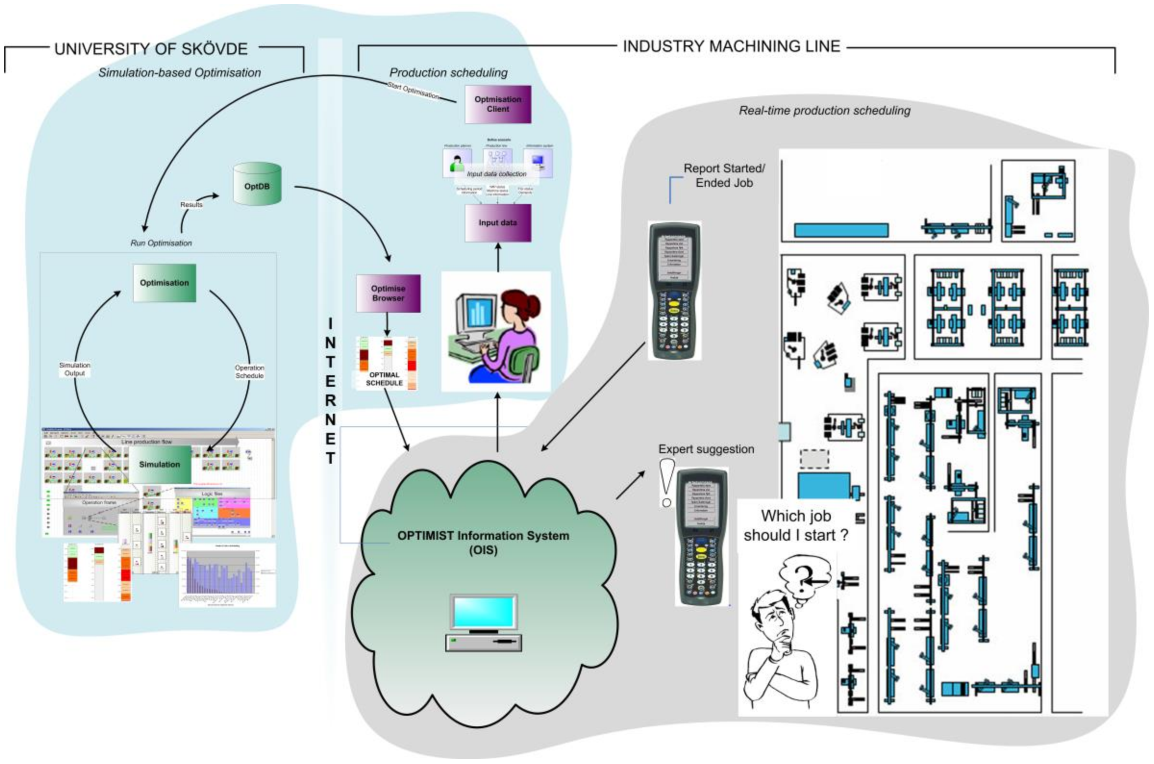


Figure 6.3 OSS implementation.

The OSS implemented in this case study is based on “SME without simulation resources”, as described for the OPTIMISE platform in Chapter 4. The production scheduler takes care of the scheduling activities using the different program clients and the actual simulation-based optimisation is carried out with the computer cluster at the university, see Figure 6.3. In addition, the production operators use Personal Digital Assistants (PDAs) on the shop floor to execute the schedule. The different parts of the complete system are located on various servers at the University or the company. In

Figure 6.4, the different parts of the OSS architecture are coloured to display their location. The web-based simulation optimisation (OPTIMISE platform) is implemented at university servers. However, the real-time dispatching has been installed on a company web-server, so that the company can access the OIS from several different internal networks. At the same time, it allows external consultant companies and the university to reach the server through virtual private network (vpn) for easy access. The response time of this server is almost immediate, since it is typically used by the company for shop floor systems. The user-interface applications are mainly used at the company by the production scheduler, but can also be run directly from the server. Furthermore, the user-interface applications can be installed on several computers and communication with the OIS server is achieved using TCP/IP and Message Queuing services. Communication with the university is accomplished using web-services. The integration sub-modules are mainly implemented on the company web-server, but changes have also been made in other systems, e.g., automatic counting points that count parts entering the Finished Goods Inventory (FGI).

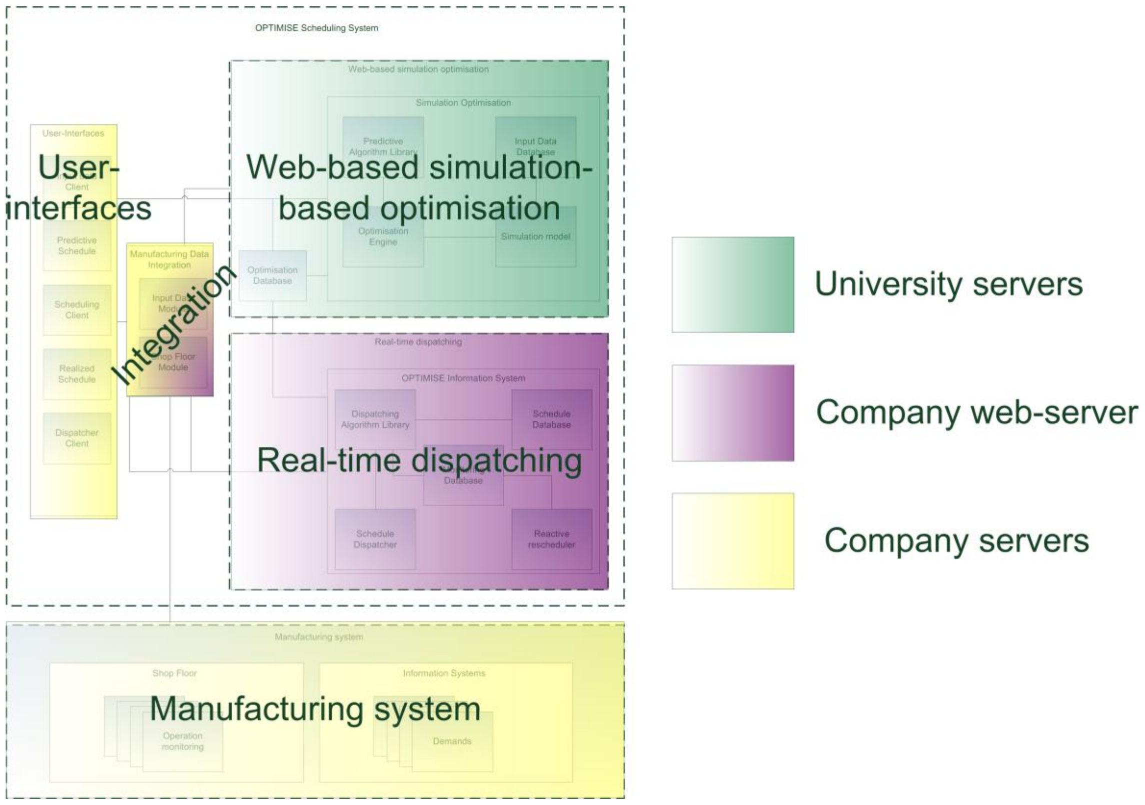


Figure 6.4 OSS distributed on different servers.

6.2.1 Input data module

The input data module imports data from several sources, when a rescheduling is taking place. It imports WIP information from the monitoring database in OIS (see Section 4.4.3) at the preparation of the scheduling data. Furthermore, information about the execution status of the schedule is automatically imported, when the optimisation is started in order to run a forecast simulation. However, for this particular implementation of the system, other information from the information systems is not automatically integrated with the input data module. Information such as demand forecasts, targets, and calendar data is handled through the input data client.

6.2.2 Shop floor module

The shop floor module handles the communication to and from the shop floor and, in fact, comprises several programs that receive, possibly restructure, and pass messages to the right destination. The dispatcher client used on the shop floor is the main program that employs the shop floor module. Dispatching requests are sent to the OIS through the shop floor module and the answer is sent back. Work-in-process (WIP) messages are sent from the shop floor and the dispatcher client specifically, or from automatic counting points. At the end of the production line, before the products enter the Finished Goods Inventory (FGI), the automatic counting of parts that identifies the correct product variants takes place. However, for this particular implementation of the system, machine failures and current machine status are not automatically integrated with the shop floor module, due to complications executing the integration.

6.2.3 Predictive- and dispatching algorithm library

The different scheduling rules could be direct, by using a direct representation; indirect, by using an indirect representation, as well as hybrid, by using a hybrid representation (see Chapter 5). The scheduling rules added to the predictive- and the dispatching algorithm libraries are the same rules. The scheduling rules in both the predictive algorithm library and the dispatching algorithm library comprise the following:

- Direct scheduling rules:

    - Sequence/Schedule
- Indirect scheduling rules:
    - FCFS stands for “first come first served”.
    - EDD stands for “earliest due date” first.
    - SPT stands for “shortest processing time” first.
    - LPT stands for “longest processing time” first.
    - VSD stands for “variant setup due date” first. What it does is select the same product variant as previously produced, if possible, otherwise it sorts the jobs according to setup time followed by due date.
- Hybrid scheduling rules:
    - HNBS stands for “hybrid non-blocking sequence”, which is a hybrid representation of schedules. The scheduling rule selects jobs according to sequence, but if those jobs are not available then it selects jobs according to FCFS. This rule is adopted from the SB-DIS rule proposed by Barua et al., (2005), with the difference that a GA is used to generate the global schedule.
    - HSNBS stands for “hybrid setup non-blocking sequence”, which is a hybrid representation of schedules similar to the latter rule. The scheduling rule selects jobs according to sequence, as long as the buffer is not full. If the input buffer is full and the next job in sequence is not available, the PDR VSD will be used to select jobs. This is due to disturbances that may set the production in a stage where the buffers become full with the wrong product variants, due to the earlier stages’ dispatching rules. Consequently, in order to avoid a deadlock caused by the dispatching rules when there are constrained buffer sizes, the strict sequence needs to be relaxed.

6.3 Scheduling user-interfaces

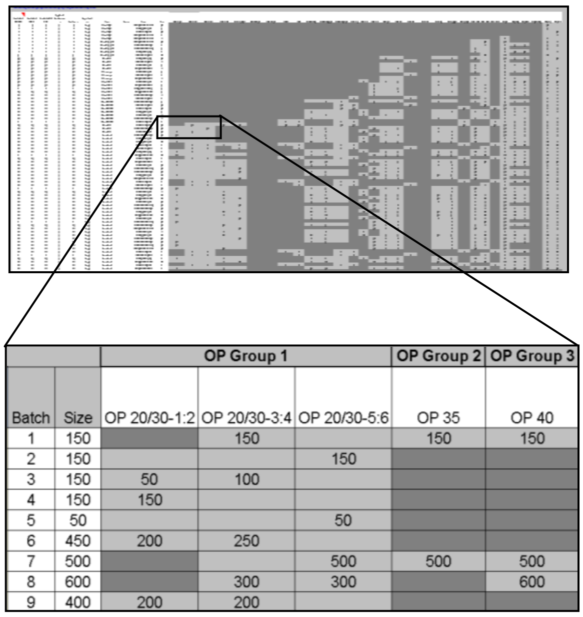
There are quite a lot of user interfaces and they are an important area of research (McKay et al., 2002). This statement is further strengthened in the thesis because it has been necessary to use many different user-interfaces in the study. The user-interfaces are in Swedish, but some of them have been translated into English for the presentation in this chapter.

6.3.1 Input data client

The users of the simulation model are not expected to be simulation experts and therefore they will work from a custom made graphical user-interface (GUI). Built in Microsoft Excel, the GUI is important because it enables the users to flexibly test many different options in the settings and some attributes of the simulation model, without the need to open the model. The GUI includes production input data such as:

- Machine processing times (13-200 seconds per part).
- Availability: mean time between failures (50-170 minutes), mean time to repair (5-25 minutes).
- Setup settings: Sequence-dependent setup times (0-2 hours), machine-dependent setup times (0-10 minutes).
- Buffers capacity (500-1500 parts) and FGI capacity (about 20.000 parts).
- Number of carriers (120).
- WIP (about 4000 parts) and FGI-contents (about 10.000 parts).
- Machine settings: on/off, product variant restrictions.
- Product demands (about 10.000 parts per week).
- FGI (about 9000 parts) and WIP (4350 parts) target levels.
- Schedule and scheduling settings.
- Product variants’ production stages.
- Product variant names mapped to product groups.
- Transportation times (120 seconds between production stages).

Consequently, it is also possible to configure and run production related tests. GUI also includes different settings that are connected to the scheduling of the line, such as start conditions and schedules for different parts of the line, see Figure 6.5.



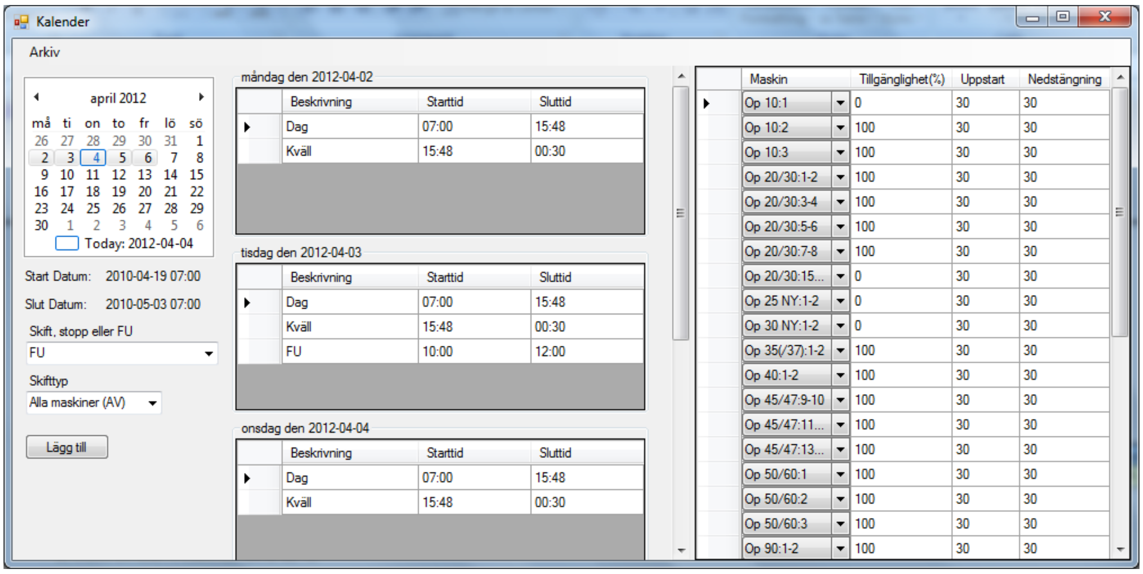
		OP Group 1			OP Group 2	OP Group 3
Batch	Size	OP 20/30-1:2	OP 20/30-3:4	OP 20/30-5:6	OP 35	OP 40
1	150	▨	150		150	150
2	150			150	▨	▨
3	150	50	100		▨	▨
4	150	150			▨	▨
5	50			50	▨	▨
6	450	200	250		▨	▨
7	500	▨		500	500	500
8	600	▨	300	300	▨	600
9	400	200	200		▨	▨

Figure 6.5 Screenshot of a schedule in Excel.

The GUI allows and helps the user fill in a schedule of the line manually and has the capability to check whether the user has made a valid plan. The input data client is created in Microsoft Excel, because it is a very flexible and frequently used tool in industry. However, the users of the Excel input data file do not need to fill in all the information through Excel, since some external programs are used to export data to the Excel file, such as the calendar function. The calendar client allows the user to create weekly workforce schedules, as described in the next section. Production engineers need to keep the simulation model up to date and are supported by user input data client to do so. The data used in the simulation model (based on real-world data) is presented in detail in Appendix G.

6.3.1.1 The calendar client

The calendar client is opened from the Excel file, but is in fact an external C# application, see Figure 6.6. The reason for having an external application is due to the usability requirements, which was a major issue with the first implementation in Excel. The calendar client allows the user to create weekly schedules by adding different shifts, planned maintenance, and so on, to each day.



må	ti	on	to	fr	lö	sö
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Today: 2012-04-04

Start Datum: 2010-04-19 07:00

Slut Datum: 2010-05-03 07:00

Skift, stopp eller FU: FU

Skifttyp: Alla maskiner (AV)

måndag den 2012-04-02

Beskrivning	Starttid	Sluttid
Dag	07:00	15:48
Kväll	15:48	00:30

tisdag den 2012-04-03

Beskrivning	Starttid	Sluttid
Dag	07:00	15:48
Kväll	15:48	00:30
FU	10:00	12:00

onsdag den 2012-04-04

Beskrivning	Starttid	Sluttid
Dag	07:00	15:48
Kväll	15:48	00:30

Maskin	Tillgänglighet(%)	Uppstart	Nedstängning
Op 10:1	0	30	30
Op 10:2	100	30	30
Op 10:3	100	30	30
Op 20/30:1-2	100	30	30
Op 20/30:3-4	100	30	30
Op 20/30:5-6	100	30	30
Op 20/30:7-8	100	30	30
Op 20/30:15...	0	30	30
Op 25 NY:1-2	0	30	30
Op 30 NY:1-2	0	30	30
Op 35(/37):1-2	100	30	30
Op 40:1-2	100	30	30
Op 45/47:9-10	100	30	30
Op 45/47:11...	100	30	30
Op 45/47:13...	100	30	30
Op 50/60:1	100	30	30
Op 50/60:2	100	30	30
Op 50/60:3	100	30	30
Op 90:1-2	100	30	30

Figure 6.6 Calendar client.

The different shifts are prepared on predetermined shift forms used by the production staff, according to system load and available personnel. The calendar function allows the assignment of types of shift for one or several days. After selecting the days and the type of event (day, evening, night, stop, and maintenance), a shift type is chosen. All shifts and maintenance activities assigned to one day can be seen in the middle field of the program. When selecting one of the events, e.g., shift, its settings will appear on the right side of the program listing the specific settings of the machines associated to that event. When the schedule has been created, the production scheduler saves the file and presses “export data” in the program menu, which then exports the data to the Excel file. Consequently, the input data file in Excel will hold all the scheduling scenario data.

6.3.2 Scheduling client user-interface

The scheduling client is the client that the production scheduler will use to generate a new schedule. The production scheduler and the shift leaders use the scheduling client, which, in Figure 6.7, contains only the information needed to reschedule.

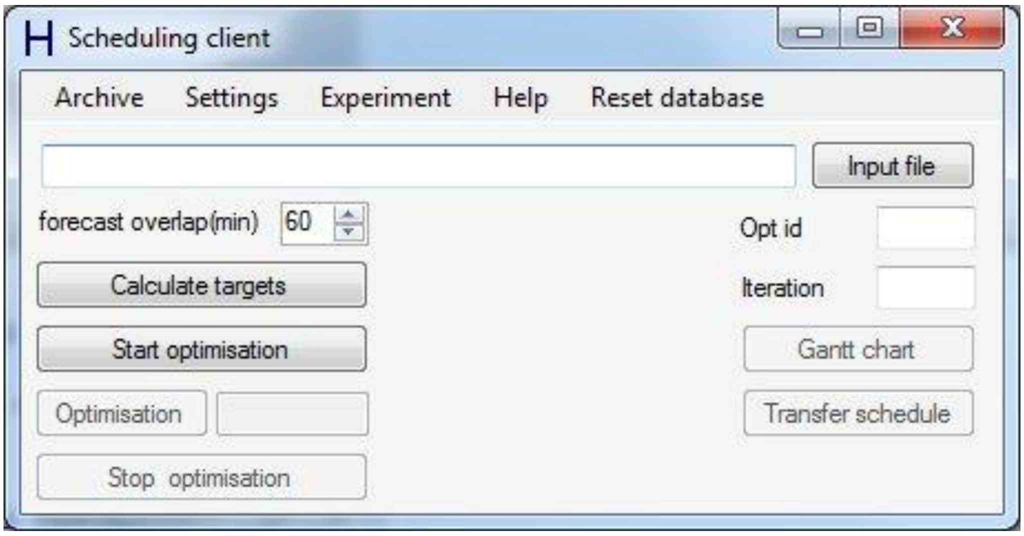
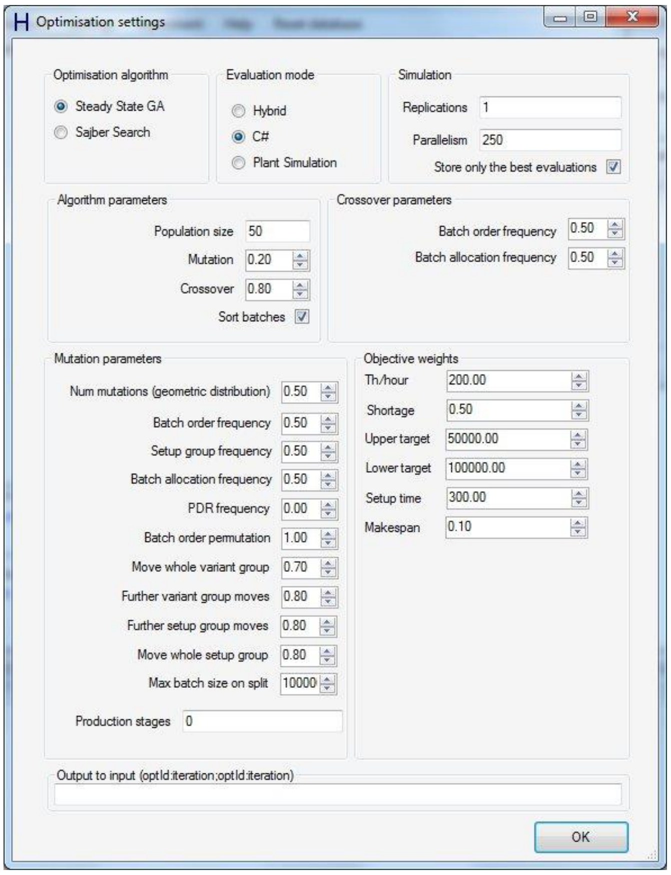


Figure 6.7 The scheduling client.

The main rescheduling tasks carried out by the production scheduler are done through the scheduling client. The input file is selected, which is the Excel file containing all the machine data, products, calendar information, and so forth. “Calculate targets” is an inter-stage for calculating proposed target levels according to demand changes. The production scheduler will have the file updated, in order to illustrate the consequences of demand changes. Thereafter, the production scheduler may decide whether the planned production volume needs to be increased or not. The “forecast overlap” is the time in which the old schedule overlaps the new schedule, in order to obtain a valid new schedule and a smooth transition to it. The “Start optimisation” option will first start a forecast simulation of updated state, using the current schedule, and then uses this as input for the optimisation that is started thereafter. The user is able to watch the optimisation and stop it once it is finished. The Gantt chart of the new forecasted schedule may be reviewed and then the new schedule is exported to the scheduling system by the option “Transfer schedule”, i.e., send and launch schedule in OIS. The production scheduler may also need to print carrier flags, which is launched through the menu and used to create information sheets attached to the carriers in the production line. This sub-program is coupled to the database that monitors what has been printed and started in the line, see Appendix A.

6.3.2.1 Optimisation algorithm settings

Through the scheduling client menu, it is possible to reach the settings of the optimisation algorithm, such as operators and population size. The following fields of settings are available through the scheduling client, see Figure 6.8.


Optimisation settings

Optimisation algorithm
[✓ Steady State GA] Sajber Search

Evaluation mode
Hybrid [✓ C#] Plant Simulation

Simulation
Replications: 1
Parallelism: 250
Store only the best evaluations [✓]

Algorithm parameters
Population size: 50
Mutation: 0.20
Crossover: 0.80
Sort batches [✓]

Crossover parameters
Batch order frequency: 0.50
Batch allocation frequency: 0.50

Mutation parameters
Num mutations (geometric distribution): 0.50
Batch order frequency: 0.50
Setup group frequency: 0.50
Batch allocation frequency: 0.50
PDR frequency: 0.00
Batch order permutation: 1.00
Move whole variant group: 0.70
Further variant group moves: 0.80
Further setup group moves: 0.80
Move whole setup group: 0.80
Max batch size on split: 10000
Production stages: 0

Objective weights
Th/hour: 200.00
Shortage: 0.50
Upper target: 50000.00
Lower target: 100000.00
Setup time: 300.00
Makespan: 0.10

Output to input (optId:iteration;optId:iteration)

Figure 6.8 Optimisation algorithm settings.

- Optimisation algorithm: The current choice of algorithms may allow the user to choose among a list of available algorithms. In the current study, the steady state GA is well suited to this problem, since it utilises parallel computers (simulation evaluations) in an efficient way.
- Evaluation mode: Selection of simulation model to evaluate the different schedules is chosen here.
- Simulation: The number of simulation replications and maximum number of parallel evaluations is selected here. In order to reduce data transfer, only the better solutions are transferred from the agents and stored in the database.

- Algorithm parameters: The basic settings of the GA are set here, namely, population size, mutation rate and crossover rate. Sort batches are used to keep the internal order of jobs of the same variant type in an ascending order based on job ID.
- Crossover parameters: The crossovers batch allocation and batch order frequency can be set here.
- Mutation parameters: The mutation parameters are many because of the importance of product variant groups. Besides the common setting of these mutations there are also non-permutation settings. “Production stages” represent the stages that are to be scheduled, i.e., zero (0) means permutation schedule since all succeeding stages will inherit the job sequence order. Hence, the permutation grade can be set here from a complete permutation schedule to a complete non-permutation schedule.
- Objective weights: Here the user may tune the objective weights of the different objectives.
- Output to input: Two-stage experiments for transforming outputs to inputs, used when going from indirect to direct representation of schedules.

6.3.2.2 Advanced mode of the scheduling client

The scheduling client also has an advanced mode used for other experiments or research related questions. Compared with the usual interface, the different functions used in the process of rescheduling are extracted here and more settings are available, see Figure 6.9.

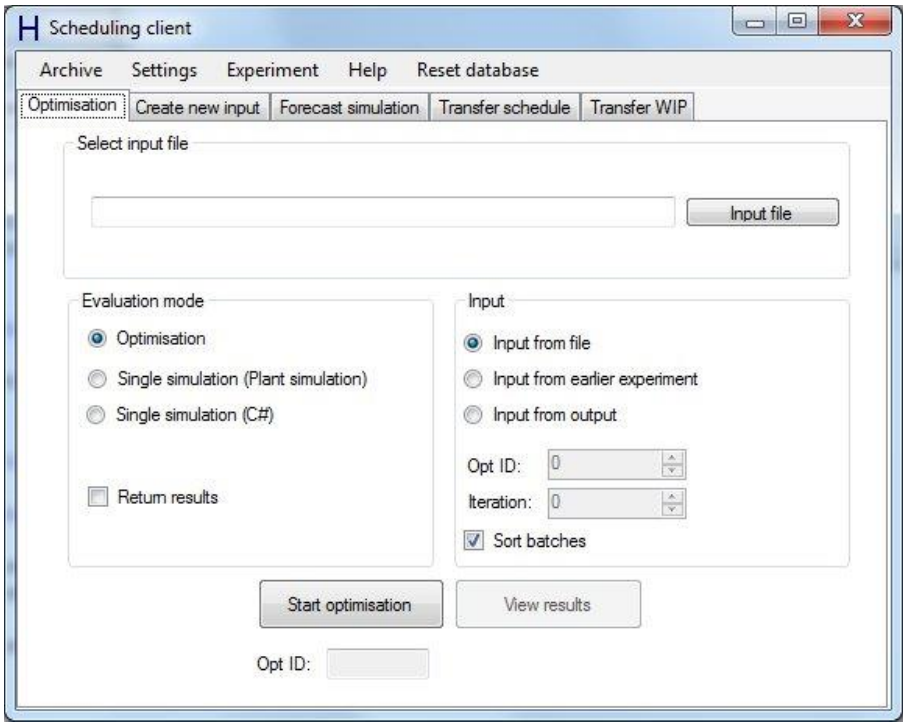


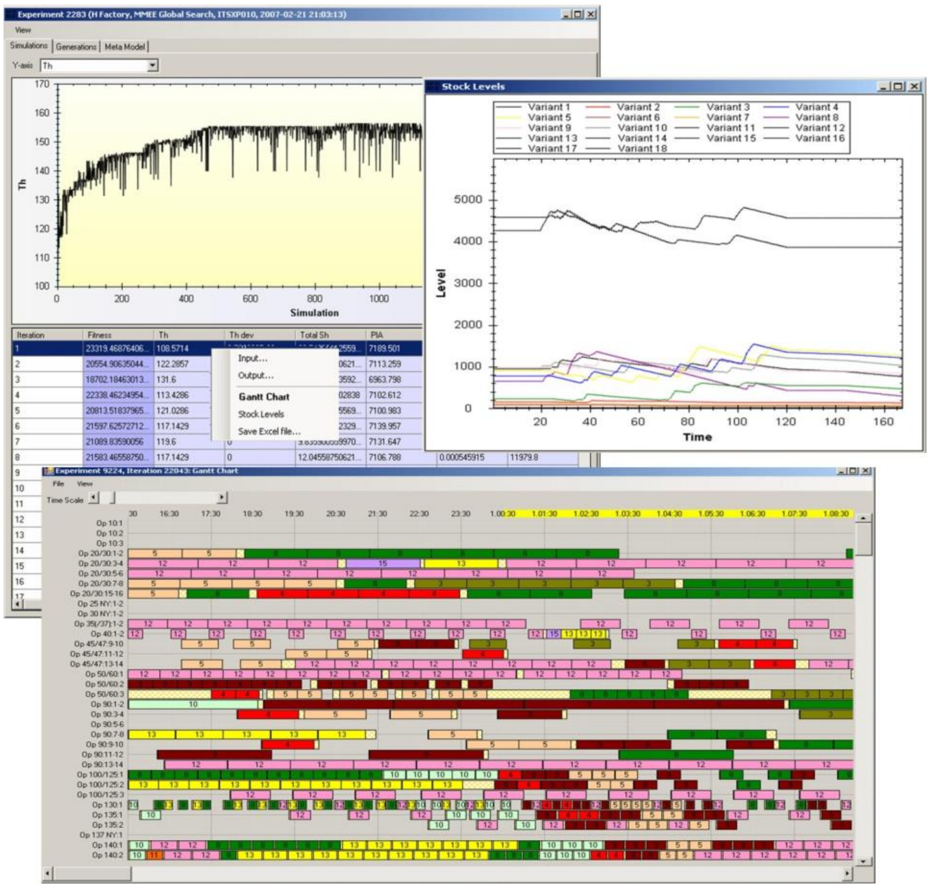
Figure 6.9 Advanced optimisation client.

Many functions are available in the advanced mode, because the different functions described for the simple client are decomposed here and may be combined in different ways. The advanced mode is mainly used for analysis, such as forecast simulation, or for optimisations not connected to the scheduling system. The forecast simulation (see Section 6.7) in the advanced mode is particularly interesting, because it provides the scheduler with the capability to predict the consequences of the current schedule, by performing a forecast simulation. By starting a simulation evaluation based on the current status of the system directly from OIS, the user obtains a new predictive schedule and objective values. The forecast simulation will indicate how the current schedule will perform if it continues with the same settings without performing a rescheduling. Thereafter, the user is provided with the decision-making support to decide whether a rescheduling is necessary or not.

6.4 Predictive schedule user-interface

With OPTIMISE Browser, it is possible to see different data plots, study the optimisation progress, and generate Gantt charts for each individual evaluation. Figure

6.10 provides the screen shots of the information that can be viewed using OPTIMISE Browser. When the optimisation is finished, the best operation schedule can be used for the real production line. Not only is it important that the industrial users view the output data related to the fitness function, but even other production data is very useful in their daily work, e.g. in decision making. Large amounts of data are stored in the optimisation database, OptDB, which is graphically displayed in the OPTIMISE browser. Production personnel might, for example, use forecasted inactive periods for planned maintenance using the Gantt chart. Being able to view the machine utilisation and stock levels over time makes it easier for the scheduler to identify which machines are the possible bottlenecks for the period and which product variants have critical stock levels that need to be followed up during the day.



Iteration	Fitness	Th	Th dev	Total Sh	PIA		
1	23319.46876406...	108.5714	[illegible]	[illegible]2559...	7189.501		
2	20554.90635044...	122.2857	[illegible]	[illegible]0621...	7113.259		
3	18702.18463013...	131.6	[illegible]	[illegible]3592...	6963.798		
4	22338.46234954...	113.4286	[illegible]	[illegible]02838	7102.612		
5	20813.51837965...	121.0286	[illegible]	[illegible]5569...	7100.983		
6	21597.62572712...	117.1429	[illegible]	[illegible]2329...	7139.957		
7	21089.83590056	119.6	0	[illegible]9970...	7131.647		
8	21583.46558750...	117.1429	0	12.04558750621...	7106.788	0.000545915	11979.8
9							
10							
11							
12							
13							
14							
15							
16							
17							

Figure 6.10 Optimisation progress.

6.4.1 Optimisation progress

Some of the output data, stored in the database, is more important from a scheduling point of view for the experts of the machining line, and is therefore presented in graphs. It enables the users to be able to observe the progress of the SBO process. Mean values and standard deviation are presented in graphs of fitness, throughput, shortage, lead time, and WIP, but if necessary, it is possible to present more output data in graphs, as the data is stored in the database. The left upper corner graph in Figure 6.10 is a screenshot from the OPTIMISE browser displaying the optimisation progress of the throughput value of a scheduling scenario.

6.4.2 Gantt chart

For each evaluation, a Gantt chart can be generated in order to study the status of machines (processing, setup, failure, planned stops, etc.) and jobs allocated to them in the scheduling period. Here the user can see how a schedule affects each machine over time. This will give the operators at the machining line additional, previously unknown information. The scheduler can see how a schedule affects the line; shift leaders and production engineers can plan the number of operators needed, plan extra work such as testing new variants, as well as plan machine maintenance and other events such as meetings. The Gantt chart at the bottom of Figure 6.10 shows the schedule from one evaluation in the simulation model. In order to make the Gantt chart useful for experts of different domains, it is possible to configure how it should be displayed. The zoom function allows the user to watch a period of interest. Furthermore, the user may select which activities (processing, setup, planned stops) should be displayed or not, in order to customise the Gantt chart to individual needs. Information is attached to jobs showing the variant number and job ID number of a processing activity. The Gantt chart bars (jobs) can be coloured according to user specifications, where in default, the colours are based on product variant types and activity types. By selecting a job in the Gantt chart, all tasks of this particular job will be marked so that it can easily be followed throughout its operation steps.

6.4.3 Stock levels

Due to the importance of handling demand fluctuations and unexpected events, such as machine break downs and product defects, it is important to monitor over time the security stock levels in the FGI of different product variants. Therefore, the FGI levels are logged over time during each evaluation, see Figure 6.10. This enables the production personnel to see a forecast of the variants that may have some critical stock levels which need to be followed up during the day.

6.4.4 Export input and output data

If necessary, it is easy to export input and output data from the OptDB through OPTIMISE browser to Excel, which enables the user to analyse and plot customised graphs on a spread sheet (e.g., using Excel). If the user wants to visualise a schedule other than just by using the Gantt chart, it is easy to export the schedule to the input data GUI and launch a simulation run manually. It is also possible to make modifications to the input data directly in the input data file if needed, e.g., change the job sequence order or machine availability. Exporting the results of the output data is done automatically by copying the matrix of the optimisation output data into Excel.

6.5 Realised schedule user-interface

Three user interfaces are employed at the actual execution of the schedule: the schedule execution status, line status, and the program used in the PDAs. Operators mainly use these PDAs to monitor WIP, but they will also be used to transfer and display scheduling information. The schedule execution status and line status are used by both the production personnel and the production scheduler to monitor the schedule execution and line status.

6.5.1 Schedule execution status

The schedule execution is presented in Figure 6.11, i.e., the Gantt chart of the realised schedule updated continuously. Consequently, this Gantt chart shows the history of the jobs that have been started at the workstations. A job currently in process at a workstation will be given an estimated end time which will be updated to the actual end time when the job is finished.

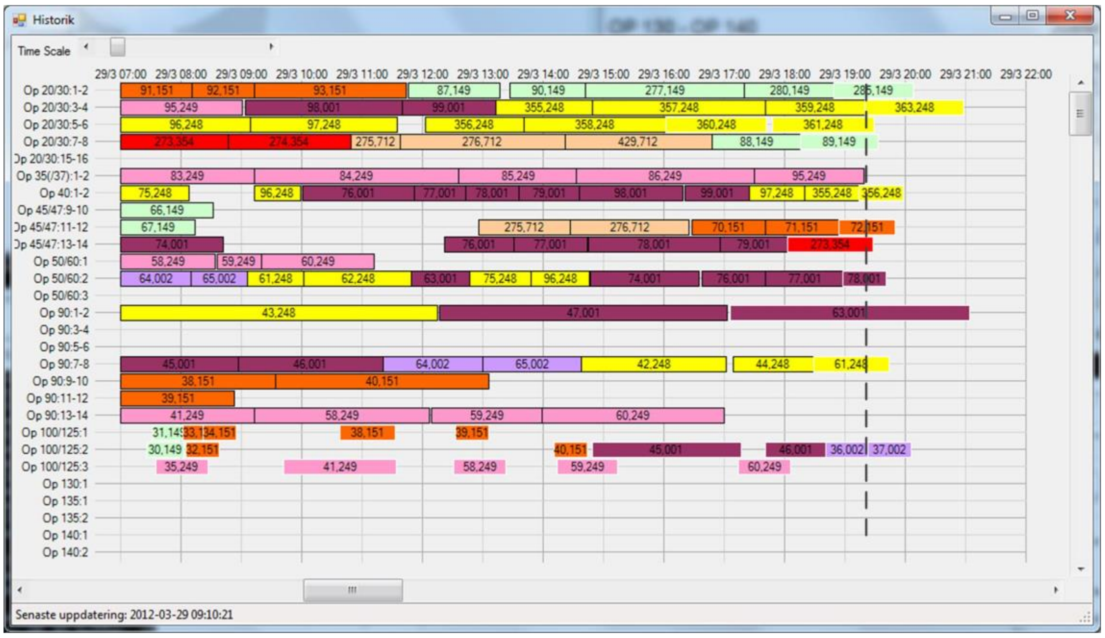
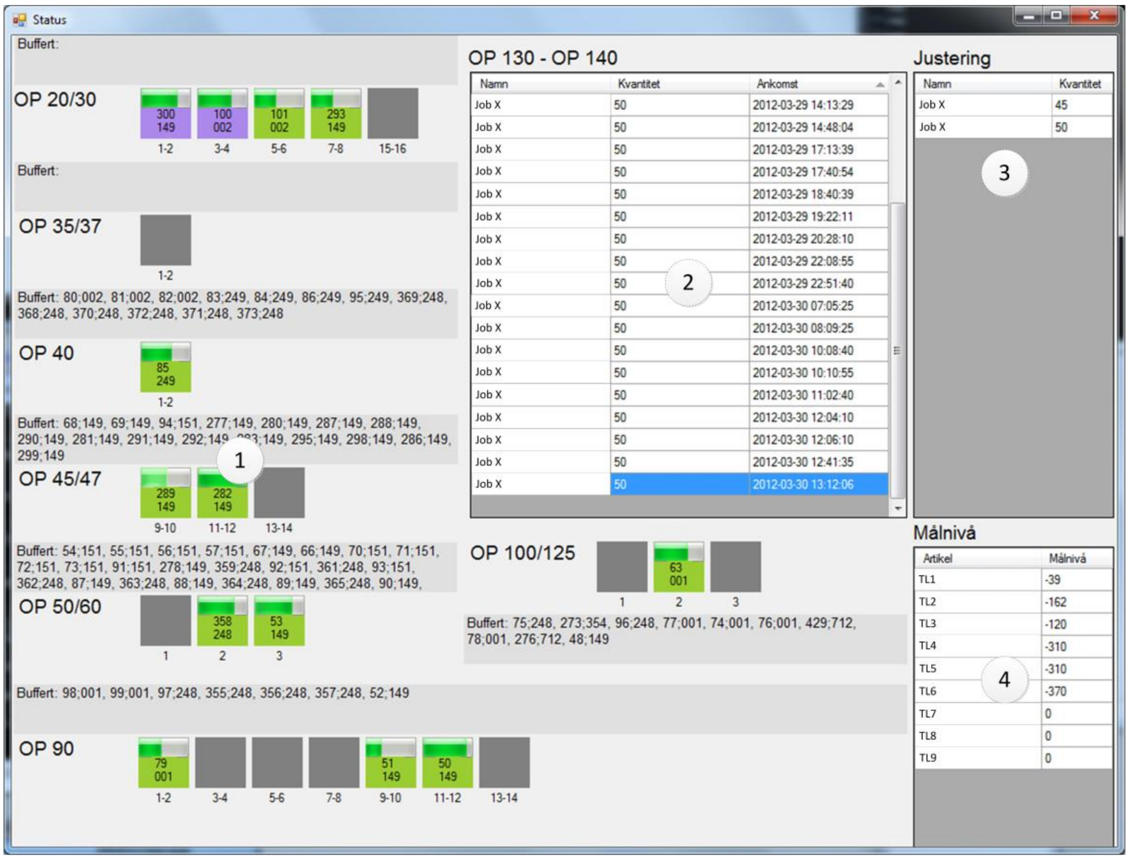


Figure 6.11 Schedule execution status.

The time line inserted in Figure 6.11 shows the time when the snapshot was taken. The jobs on and after this line show those currently being started, i.e., predicting the end time of the jobs. Also, the history and current state of machines give the scheduler a quick status overview of the system, because it is easy to see whether a machine has been idle for some time. If it has been idle, it might imply that there is some material deficit or workstation failure. Since data is communicated only from the dispatcher client and not from the actual workstations, the reason for the idle periods cannot be clearly presented. The dispatcher client, described in section 6.6, mainly sends data about jobs being started or finalised at the workstations.

6.5.2 Line status

The line status program provides WIP data and associated information. In the line status program, as displayed in Figure 6.12, the user can see the current WIP and machine status (indirectly). Furthermore, there is information about the target levels of the product variants for the week.



Status

Buffert:

OP 20/30

300 149	100 002	101 002	293 149	
1-2	3-4	5-6	7-8	15-16

Buffert:

OP 35/37

1-2

Buffert: 80;002, 81;002, 82;002, 83;249, 84;249, 86;249, 95;249, 369;248, 368;248, 370;248, 372;248, 371;248, 373;248

OP 40

85 249
1-2

Buffert: 68;149, 69;149, 94;151, 277;149, 280;149, 287;149, 288;149, 290;149, 281;149, 291;149, 292;149, 283;149, 295;149, 298;149, 286;149, 299;149

OP 45/47

289 149	282 149	
9-10	11-12	13-14

Buffert: 54;151, 55;151, 56;151, 57;151, 67;149, 66;149, 70;151, 71;151, 72;151, 73;151, 91;151, 278;149, 359;248, 92;151, 361;248, 93;151, 362;248, 87;149, 363;248, 88;149, 364;248, 89;149, 365;248, 90;149,

OP 50/60

	358 248	53 149
1	2	3

Buffert: 98;001, 99;001, 97;248, 355;248, 356;248, 357;248, 52;149

OP 90

79 001				51 149	50 149	
1-2	3-4	5-6	7-8	9-10	11-12	13-14

OP 130 - OP 140

Namn	Kvantitet	Ankomst
Job X	50	2012-03-29 14:13:29
Job X	50	2012-03-29 14:48:04
Job X	50	2012-03-29 17:13:39
Job X	50	2012-03-29 17:40:54
Job X	50	2012-03-29 18:40:39
Job X	50	2012-03-29 19:22:11
Job X	50	2012-03-29 20:28:10
Job X	50	2012-03-29 22:08:55
Job X	50	2012-03-29 22:51:40
Job X	50	2012-03-30 07:05:25
Job X	50	2012-03-30 08:09:25
Job X	50	2012-03-30 10:08:40
Job X	50	2012-03-30 10:10:55
Job X	50	2012-03-30 11:02:40
Job X	50	2012-03-30 12:04:10
Job X	50	2012-03-30 12:06:10
Job X	50	2012-03-30 12:41:35
Job X	50	2012-03-30 13:12:06

OP 100/125

	63 001	
1	2	3

Buffert: 75;248, 273;354, 96;248, 77;001, 74;001, 76;001, 429;712, 78;001, 276;712, 48;149

Justering

Namn	Kvantitet
Job X	45
Job X	50

Målnivå

Artikel	Målnivå
TL1	-39
TL2	-162
TL3	-120
TL4	-310
TL5	-310
TL6	-370
TL7	0
TL8	0
TL9	0

Figure 6.12 Line status program.

6.5.2.1 Buffers and workstations

In the first area, marked with the number "1" in Figure 6.12, the WIP areas are divided into two different parts: the buffers and the workstations. The current WIP gives the user a quick overview of the line status, i.e., long or short queues of different product variants. The production stage buffers are those that store all jobs to be processed at that production stage. Each production stage may have several parallel workstations and each workstation has its own WIP area, as displayed by the machine boxes in dark grey, green, and purple. The dispatcher client mainly sends information about when a job has been started or finalised at the workstations. Each job is presented with its specific job ID number and product variant number. When a job is finalised, it will be moved to its next production stage, which may differ for a different type of product variant.

Consequently, the job will be moved to the production stage buffer and wait there until it is selected for a workstation. When a job is started at a workstation, it will be moved from the production stage buffer to the WIP area of the workstation. The workstation displays the current job and the box colour will be set to green, which stands for “processing” in the company. The status bar at the top of the workstation will display the estimated progress of the job, as a guide for the operators about when the job will be finished. The machine status is indirect information based on dispatcher client messages, which means the user can quickly see the status of the machine.

At some of the workstations which have input conveyors, two jobs can be started at the same time if they are of the same product variant type. The reason for this is to keep the workstation busy without any delays. However, only the first job will be displayed in the box, but the colour will be set to purple indicating that two or more jobs have been started at the workstation, see Figure 6.12. The status bar will be associated with the first job, but when it is finished, the status bar will be associated with the next job. More detailed information of the workstations and their jobs can be viewed by clicking on the workstation.

6.5.2.2 Buffer area without job ID numbers

The second area, marked with the number “2” in Figure 6.12, is a large buffer area at the end of the production line. Since this part of the line is less complex, it uses FCFS priority dispatching. Furthermore, due to practical reasons, the jobs lose their job ID number at this part of the line, since the carrier flags will be detached from the carrier (job). Additionally, the operators will not use the dispatcher clients at this part of the line. For this reason, the jobs located at the end of the line, which consists of three production stages, will be estimated by keeping the jobs on a FCFS priority list. Consequently, all jobs started at the last workstation in the first area (first part of the line) will be added to the WIP list of the second area. At the end of the line, when the jobs enter the FGI, they are automatically counted, which is communicated to the program, making it possible to maintain a correct WIP status.

6.5.2.3 Adjustments and quality control

The WIP of area three, marked with the number “3” in Figure 6.12, monitors the jobs that have been sent for adjustment or quality control. Quality control is a continuously on-going activity that must be carried out. Therefore, some of the parts of a carrier may be transferred to this area. Adjustments are sent here in the same way and some of them will be scrapped and others sent back to the production line.

6.5.2.4 Target levels

The fourth area, marked with the number “4” in Figure 6.12, shows the target fulfilment of the current week. There are different target levels that need to be reached for different product groups and the number of parts currently over or under the target level is displayed here. The target levels will help the production personnel reach targets, and the targets will be updated (if necessary) at each reschedule.

6.6 Dispatcher client user-interface

The dispatcher client is the device used by the operators to execute the schedule. In this case study, the dispatcher client device used is a PDA, see Figure 6.13. PDAs are mainly used by the operators today to monitor WIP. The main reason for using PDAs is the fairly low investment cost of working with existing equipment. The PDAs will therefore also be used to transfer and display scheduling information.

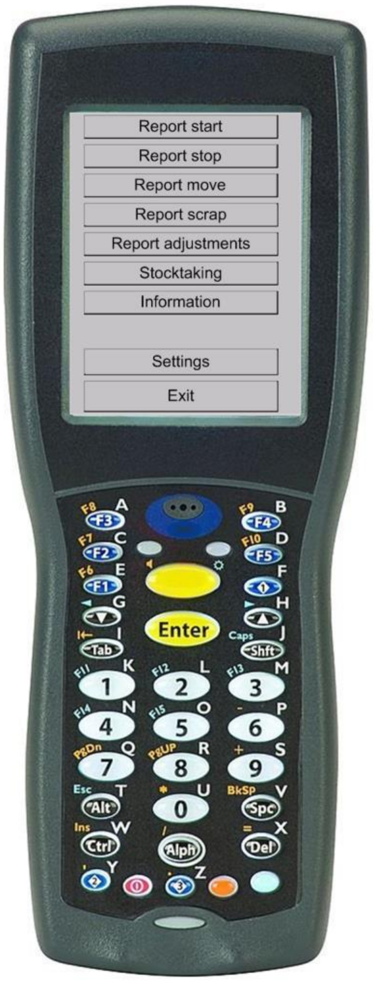


Figure 6.13 Personal digital assistant.

The PDAs are programmed to be able to report jobs being started or ended and retrieve information about the next job to start. The WIP level of a production stage denotes the number of jobs available for execution at the workstations belonging to that production stage. The different WIP messages are: *job started, job stopped, job on hold, parts scrapped, job continue, move job started/stopped, add job*. When an operator requests an expert suggestion in the PDA, a message is sent to OIS, where it uses a program called “Schedule Dispatcher” that determines the next job, depending on the current scheduling approach used, i.e., sequence or dispatching rule. Thereafter, the expert suggestion is sent back to the PDA, based on both the scheduling approach used and the available WIP. The PDAs have linear bar code readers, making the process of handling them more efficient.

Information signboards with different bar codes for the machines are placed on the production line and each carrier has the product variant bar code, in order to facilitate the work for the operator. Once the operator scans a “start bar code”, he/she will immediately receive an expert suggestion in the PDA, see Figure 6.14.

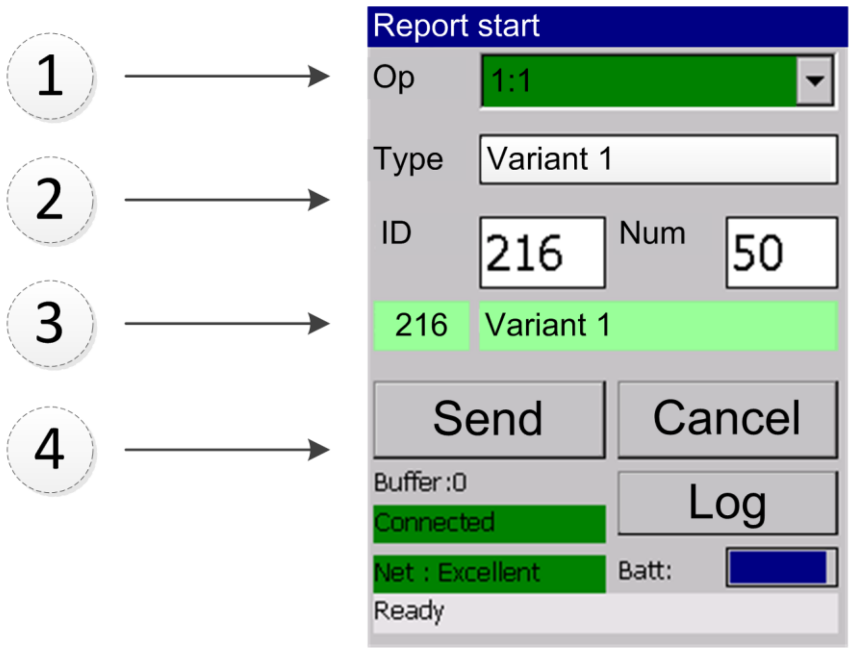


Figure 6.14 Expert suggestion in the PDA.

The expert suggestion screen, in Figure 6.14, works in the following way:

- The expert suggestion screen will appear when the start-bar code of a workstation is selected or scanned. Thereafter, a message is sent to OIS which will return the next job in line to be produced at the workstation in less than one second.
- The ID number and variant type of the job is returned. If the job is currently available, it will be coloured green, but if it is not available, it will be coloured red.
- The operator selects the correct job, in this case job ID number 216, by typing the ID number or scanning the barcode. Information is sent to OIS, which immediately returns information about the job’s variant type (Variant 1) and the number of parts associated with the job (50).
- The last step is to execute the start of a job by pressing send.

The PDA program consists of several screens, in order to handle different tasks, e.g., there is a special function for stocktaking; efficiently and quickly counting the jobs and noting their position in the production line. Almost all screens can be seen in Appendix B.

6.7 Scheduling scenarios

In order to run an optimisation, the user first defines the scheduling scenario, starts the optimisation, and then receives the results through the OPTIMISE browser. To define the scheduling scenario, certain information is needed. Although the scheduling period of interest might only be one day, the simulated scheduling horizon will be longer, in order to avoid any sub-optimal schedules (Wiers, 1996), see Figure 6.15.

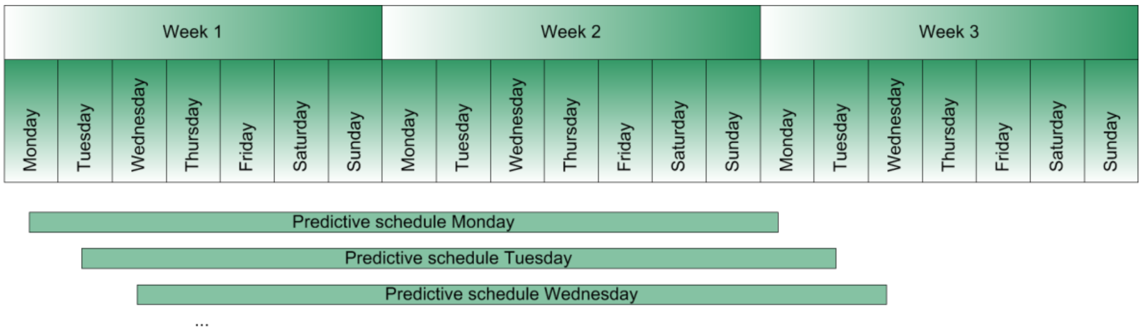

Week 1
Week 2
Week 3
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Predictive schedule Monday
Predictive schedule Tuesday
Predictive schedule Wednesday
...

Figure 6.15 Scheduling horizon and periodic rescheduling.

Figure 6.15 shows what a periodic rescheduling policy would look like for the industrial case study, when using a scheduling horizon of two weeks and a periodic rescheduling each production day. At each scheduling cycle, the user defines the scheduling scenario by collecting information from different systems. The scheduler needs to import or create the following information for the scheduling program: calendar information (shift forms, planned maintenance), status of machines and personnel, WIP status, FGI status, and demands. The calendar information is put into the calendar program manually by the scheduler once a week, unless something has changed. Furthermore, there are different predefined weekly shift forms that the user can select and add to the calendar. Providing this information can be done once a week, but also for several weeks ahead. Daily changes, such as status of machines and personnel, are communicated to the scheduler and the necessary alterations are manually fed into the program. When the user is finished, the data is stored in a SQL-lite database file and exported to the selected, input data Excel file. FGI status and demands are fed into the input data Excel file, i.e., data is copied manually from another system. The information gathered for the scheduling scenario is used to create a new schedule on which the optimisation is based. Detailed WIP status and machine setup are imported directly from OIS. A built-in macro in the input data file in Excel allocates WIP and the start of new jobs, which is based on demand information. The WIP status provides information about the location of different jobs, i.e., in front of or inside machines. If a job is being processed in a machine, then the estimated remaining processing time is imported as well. If the optimisation was to be finished immediately or in just a few seconds, the new schedule would not have to bother about the merging problems between the old and the new schedule, because nothing would have happened during this period of time. However, the optimisation takes time (currently10-20 minutes) and gathering input data also takes some time, which makes it impossible to guarantee that no jobs have been started during this period. Therefore, the schedule reconfiguration, i.e., the process of bringing the active schedule in line with the new schedule, is necessary. The forecast simulation helps to achieve a smooth transition to the new schedule. The process here is an almost seamless launch of the new schedule so that the operators will not even notice that a new schedule has been created. Consequently, the forecast simulation reduces the risk of creating a schedule that is not up-to-date when released and the schedule reconfiguration updates the new schedule (match-up scheduling) with the events that have actually occurred during the optimisation run.

There are six machines (M1-M6) in this example, on rows one to six, where M1-M3 are parallel machines in one processing step and M4-M6 are parallel machines in the subsequent processing step. The example in Figure 6.16 shows the realised schedule until 14:00, the rescheduling point at 14:00, and the forecast simulation for the next two hours.

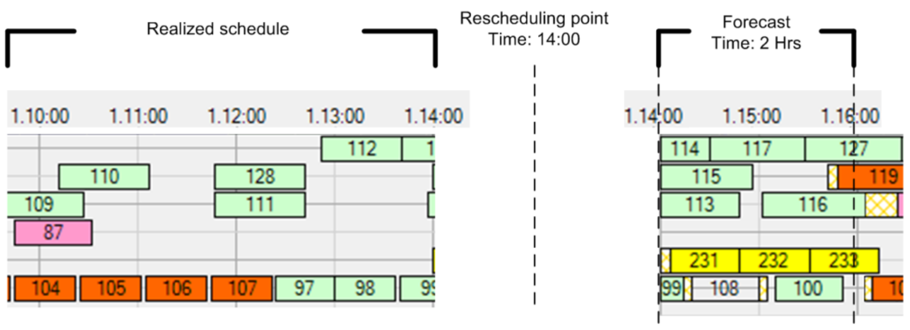


Figure 6.16 The rescheduling point.

The forecast of two hours marks those jobs that could be produced in the next two hours, if production was to continue with the current schedule. In this example, the jobs are: 114, 117 and 127 for M1; 115 and 119 for M2; 113 and 116 for M3, none for M4; 231, 232 and 233 for M5; 99, 108 and 100 for M6. These jobs are then used as input for the rescheduling optimisation, so that the new schedule always puts these jobs first, i.e., a frozen period, see Figure 6.17.

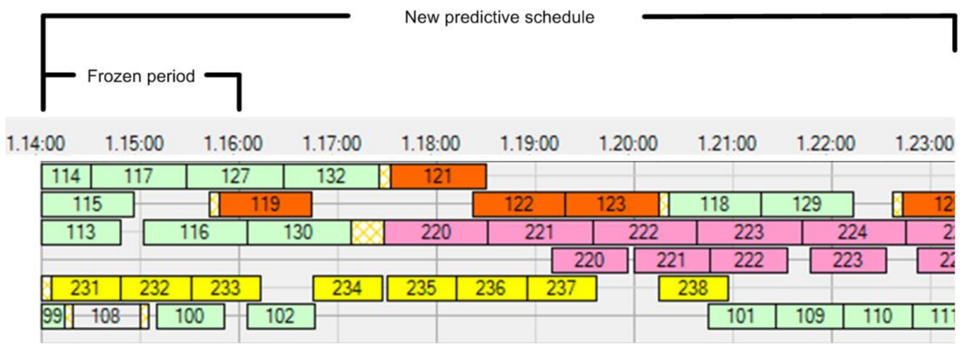


Figure 6.17 The new predictive schedule.

The scheduler uses the scheduling program to start an optimisation, which means that the input data file is sent to the optimisation engine at the server site via a web service interface. At the client site, the user can watch the progress of the optimisation using OPTIMISE Browser. When the optimisation is finished, the user can select the best schedule and then send it to OIS, where it will update the current schedule with the new one. The schedule reconfiguration program is executed and makes sure that the new schedule is up-to-date. The complete realised schedule from the example can be seen in Figure 6.18.

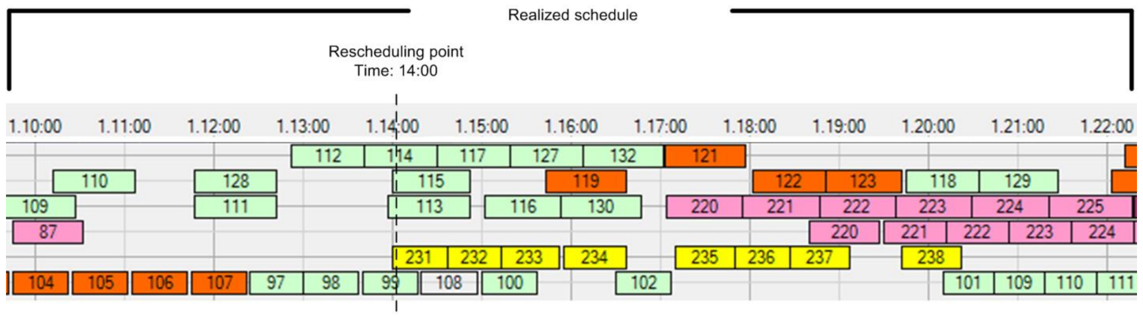


Figure 6.18 The realised schedule.

By using the forecast scheduling function, the number of jobs that deviate at schedule reconfiguration is diminished or zero. If the operators at the different machines follow the expert suggestion without any deviations, a schedule reconfiguration is not necessary, due to the use of the forecast function.

6.8 Real-world deployment stages

Wiers (1996) commented that most literature reports of scheduling systems only describe the architecture of a system and do not indicate whether a system has been implemented in the real world. He also stated that the implementation pitfalls need to be identified by researchers for other researchers, in order to succeed with scheduling techniques in practice. This section describes the steps toward a complete deployment of the OSS at the targeted automotive manufacturer and presents the outcomes. There are totally six deployment stages, as illustrated in Figure 6.19.

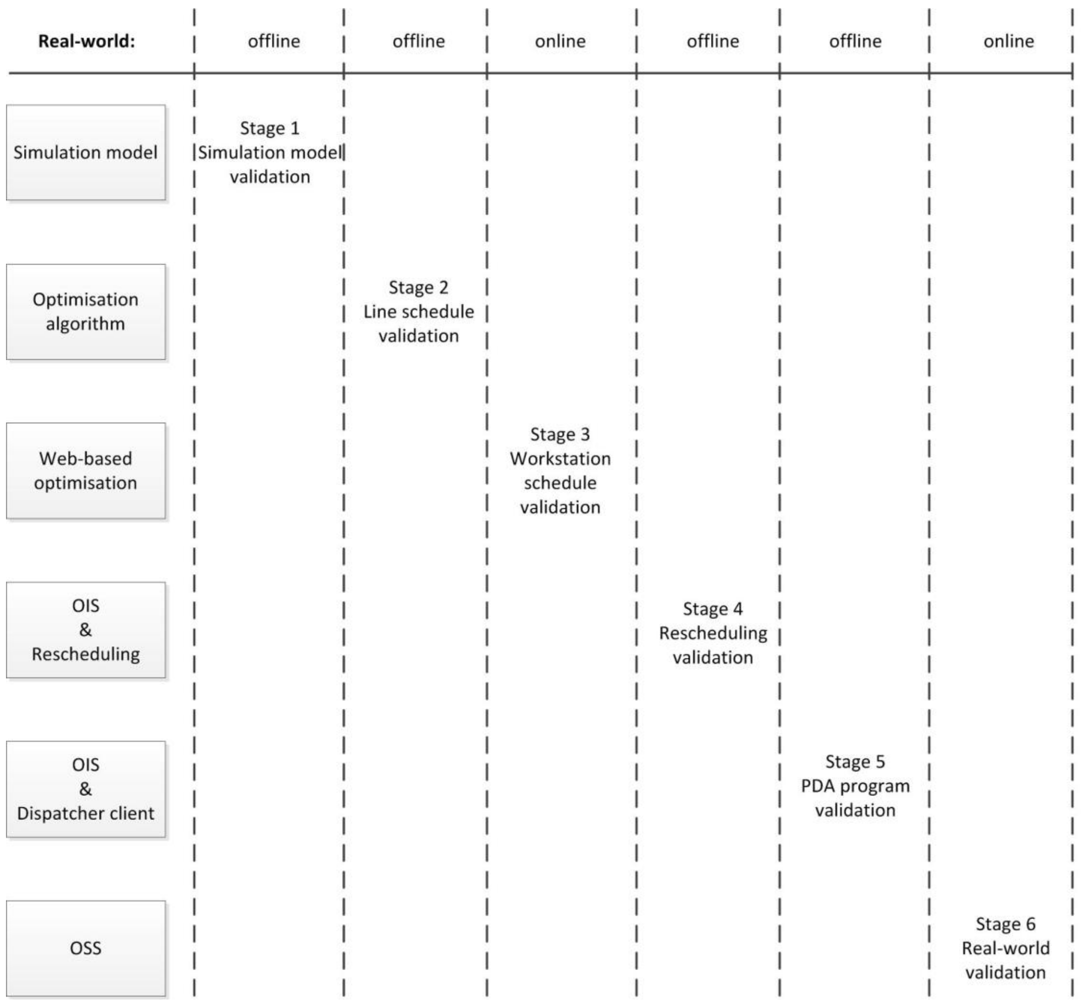

Real-world:
offline
offline
online
offline
offline
online
Simulation model
Optimisation algorithm
Web-based optimisation
OIS & Rescheduling
OIS & Dispatcher client
OSS
Stage 1 Simulation model validation
Stage 2 Line schedule validation
Stage 3 Workstation schedule validation
Stage 4 Rescheduling validation
Stage 5 PDA program validation
Stage 6 Real-world validation

Figure 6.19 Real-world deployment stages.

There are six stages in Figure 6.19 beginning with the simulation model and ending with the whole OSS system. At the top of the figure, it is also indicated whether the test experiments were performed online in the real-world production line, i.e., using the schedule obtained from the optimisation in the daily work.

6.9 Stage 1: simulation model validation

A validation plan was created to decide how to validate the simulation model and to serve as a basis for production follow up. The simulation model has been validated primarily by using two different techniques to enable correct and credible conclusions to be drawn. In the first step, a structured model walk through of the simulation model and the assumptions document was carried out on the simulation model together with the scheduler, operators and production engineers, in order to approve functionality and simplified assumptions in the simulation model. In the second step, detailed instructions were formulated on what data was needed for the production follow up and how this data could be collected. Excel templates were also created in order to make the production follow up and comparison of results easier. The output data from the simulation model closely resembled the output data of the actual system, and the simulation model was considered valid (Law and Kelton, 2000).

6.10 Stage 2: line schedule validation

In stage two, the schedules were evaluated using real-world data without actually disturbing the real-world production line, so called mock run. In the mock run, schedules based on real production data were created but not used on the production floor, simply in order to ascertain whether good schedules can be created in various real-world situations. In Figure 6.20, the process of one of the methods used in the mock runs is shown.

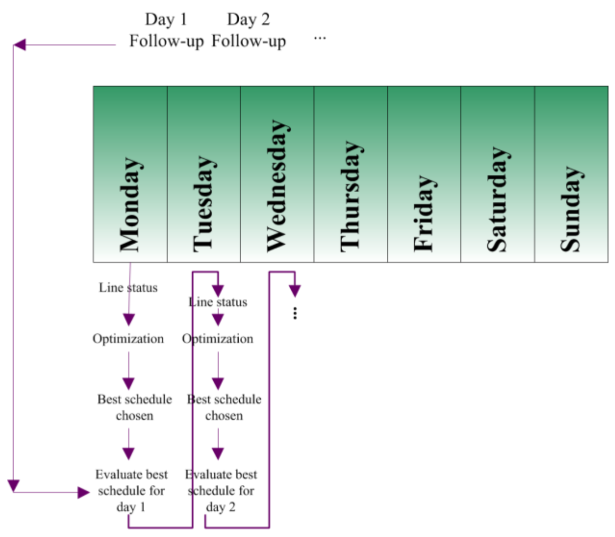


Figure 6.20 Validation of a mock run.

The scheduling period was set to one week to avoid sub-optimal schedules. According to Figure 6.20, data was collected each day of the scheduling period. On the first day, all this data came from the real production line. An optimisation was started and the best schedule was chosen. Then one day that included information from the production floor, e.g., failures, was simulated with the chosen schedule. The next day, the scheduling process continued except that parts of the data collected, i.e., WIP and production results came from the simulation. The result of the experiment, in this case the number of parts produced, can be seen in Figure 6.21.

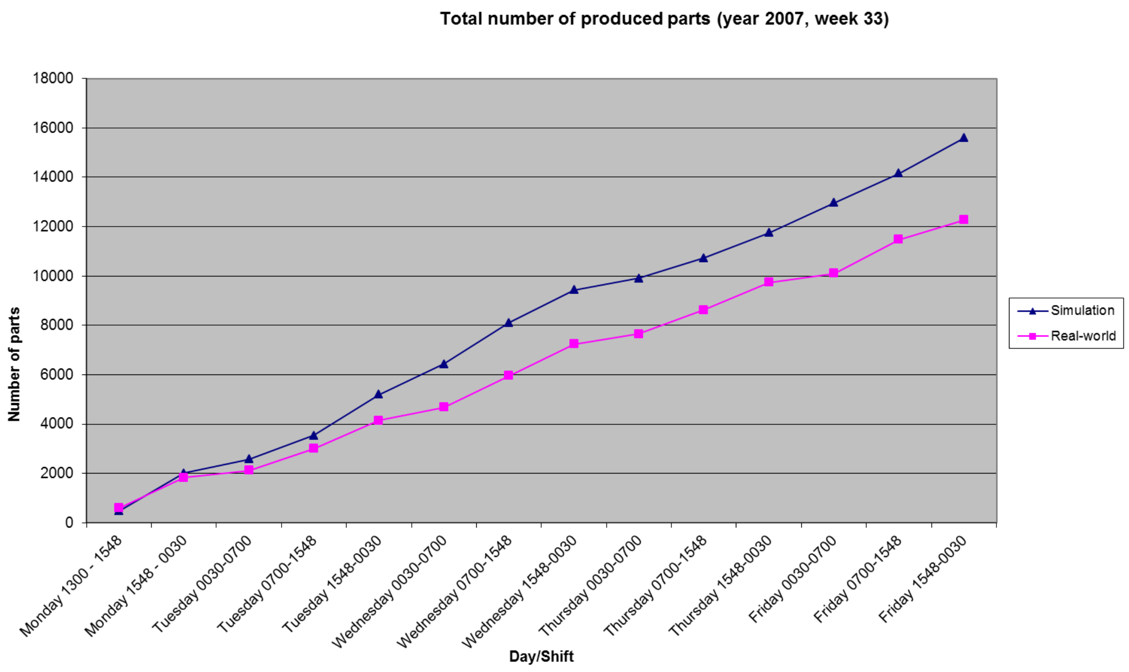


Figure 6.21 Comparison of results from OSS with actual output from the real line.

After one week of production, the results of the simulation-based scheduling were compared with the results of the real production system. The outcome indicates that the simulation-based scheduling can generate better schedules than the ones used from real production.

6.11 Stage 3: workstation schedule validation

When only the web-based simulation optimisation system is used, there is no real-time dispatching. The reason for testing this part of the system was to see whether this part of the system was sufficient for decision support and whether the predictive schedules are good enough when used in daily real-world production. Only the web-based simulation optimisation part was evaluated in this stage, which omits the OIS. Instead of using the OIS, the workstation schedules were printed and attached at the machines. Since this application introduced a new mode of operation to the production scheduler, operators and production engineers, real-world test weeks were needed to thoroughly examine its applicability. An example of a predictive workstation schedule used during the test is shown in Figure 6.22.

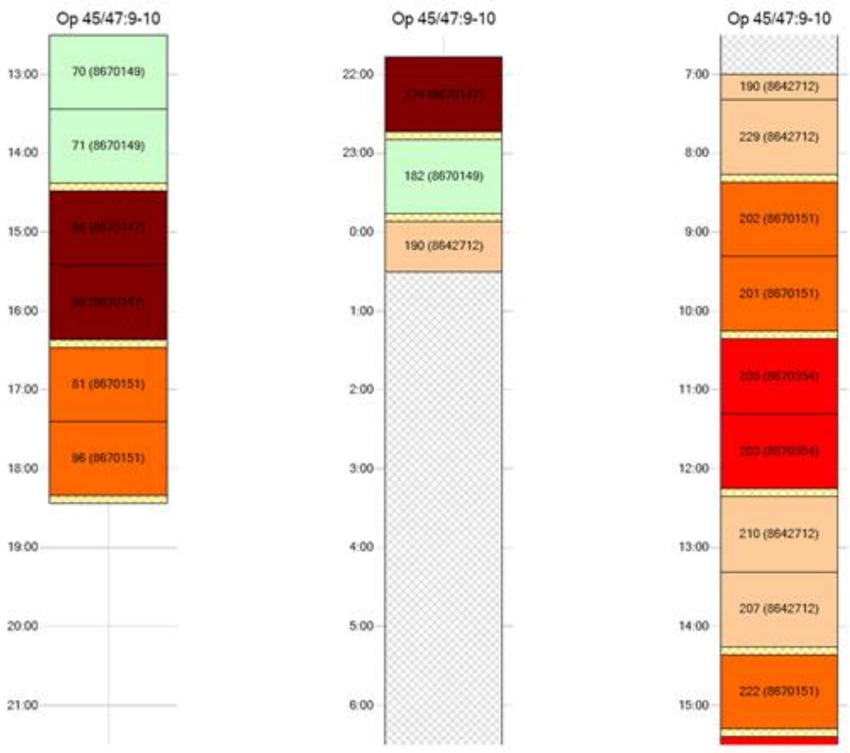


Figure 6.22 A workstation schedule.

The workstation schedule shows the predictive times of jobs being produced, setups to be made, and unplanned periods. Several scenarios and trial weeks were tested and the results were promising. Better schedules were found when it was compared to the scheduler's prediction of makespan. The importance of including many real-world constraints in the model representation of the scheduling problem was found to be significant. For example, the predictive schedule looked odd because a machine was unscheduled for the first two hours for some reason, even though the right jobs were available. The production personnel and scheduler thought that something was wrong with the schedule, but the number of carriers in the second carrier loop was limited and caused exactly the same behaviour in the real-world. However, the work procedure was found to be impracticable for three reasons: (1) the data collection was too time-consuming each time a rescheduling was needed; (2) distributing the machine schedules in the production line was time-consuming; (3) system nervousness due to system reconfiguration at the implementation of a new schedule. The operators also thought that the schedule generated had a too high mix of product variants. This matter was partly solved by having variant and setup grouping in the genetic algorithm, which has been studied further in Chapter 5. It has been demonstrated qualitatively that OSS has actually solved many of the problems identified here.

6.12 Stage 4: rescheduling validation

When a predictive schedule is created, it is the same schedule being realised (the realised schedule), as long as there is no deviation or disturbance. In reality, things do not go entirely as planned, creating a situation where there is not a scheduling problem, but rather a rescheduling one (Graves, 1981). From a scheduling system point of view, the optimisation of the predictive schedule is only one part of the solution. Being able to actually use the schedule when the real production line is subject to different disturbances, resulting in the realised schedule, is another important feature of the industrial application. The whole system needs to be tested both off-line and on-line in the real-world production. In order to test the system off-line, a virtual environment is needed that simulates the real-world production.

Kempf et al., (2000) present different ways of comparing schedules. The method used to find out whether good schedules have been found or not is the *relative comparison* to the real-world result. Furthermore, a *dynamic measurement* is used here where the predictive schedule is tested in the real environment subject to disturbances. The disturbances used in this test are historical data from five successive weeks, using a trace-driven simulation approach (Law and Kelton, 2000). The *schedule measurement*s are multiple objectives and the *state measurement* is handled through the implementation of simulation-optimisation target levels. Examples of a state measurement that has been included here are FGI target levels, also included in the optimisation objectives, and WIP target levels that are not included in the optimisation objective. The WIP target levels are still important, since leaving a production line with levels that are too low or with the wrong kind of WIP may lead to a problem later on.

Consequently, the overall production target will not be reached until the WIP target levels have been achieved.

The multiple objectives used to compare the simulated realised schedule to the real-world realised schedule are somewhat different from the objectives used at the optimisation, due to measurability issues in the real world. The virtual environment used here not only tests whether a good realised schedule has been attained, but also parts of the communication, such as the communication with OIS. Hence, the whole OSS system is tested as well. The experimental settings can be found in the next sub-section and all quantitative results of the experiment, including those from the reactive scheduling tests, are presented in Chapter 7.

6.12.1.1 Simulation environment and experimental procedure

In the simulation environment test, extensive and detailed data was gathered for five successive weeks from the real-world production line. This test was similar to a validation with historical data, i.e., an actual historical record is used to drive the simulation model (trace-driven simulation) and then the model output is compared with the system data. Following is a list of the important data collected through OSS:

- WIP status of the line at the start of the period. (See Appendix G, Table G.9)
- FGI status over time.
- Demand forecasts. The forecasted demand (e.g., Appendix G, Table G.14) is not the same as the actual historical record (see Section 3.1) and therefore the forecasted demand is updated once per day when used for the SBO.
- Unplanned disturbances, e.g., breakdowns/failures of machines (See Section 3.1). Failures (about 20 hours per week), demands, and processing time uncertainties were considered.
- Processing times. The processing times used for SBO is based on historical data whilst the data used for the simulation environment is actual historical records (see quality of data in Section 3.1).

- Working hours. Two shifts, 7.30 a.m.-3.48 p.m. and 3.48 p.m.-12. 00 midnight.
- Planned maintenance: Mondays 10.00 a.m.-10.30 a.m., Mondays 5.00 p.m.-5.30 p.m., Thursdays 10.00 a.m.-12.00 noon and Thursdays 5.00 p.m.-7.00 p.m.
- Real world results.

The transient status of the line, i.e., machine settings, calendar information, WIP status, and so forth, on the Monday morning of week number one was used as input to start an optimisation. The demand forecast obtained the same day was used to generate the necessary jobs to be started in order to satisfy the demand. Disturbances, such as failures, were estimated and based on earlier production follow-ups.

A specific input data file that takes into account actual current state and future events based on forecasts was created in Excel for the SBO. The simulation-based-optimisation was executed and the best schedule was selected to be used in the simulation environment. Another input data file that takes into account actual current state and future events based on actual real-world outcomes for a specific week was created in Excel for the simulation environment. The results of that week consist of actual demand, which is different from the forecasted demand, and those disturbances, i.e., short and long machine stops, which occurred during that week. However, the schedule from the optimisation was used for the schedule execution in the simulation environment. The procedure of the simulation environment test can be seen in Figure 6.23.

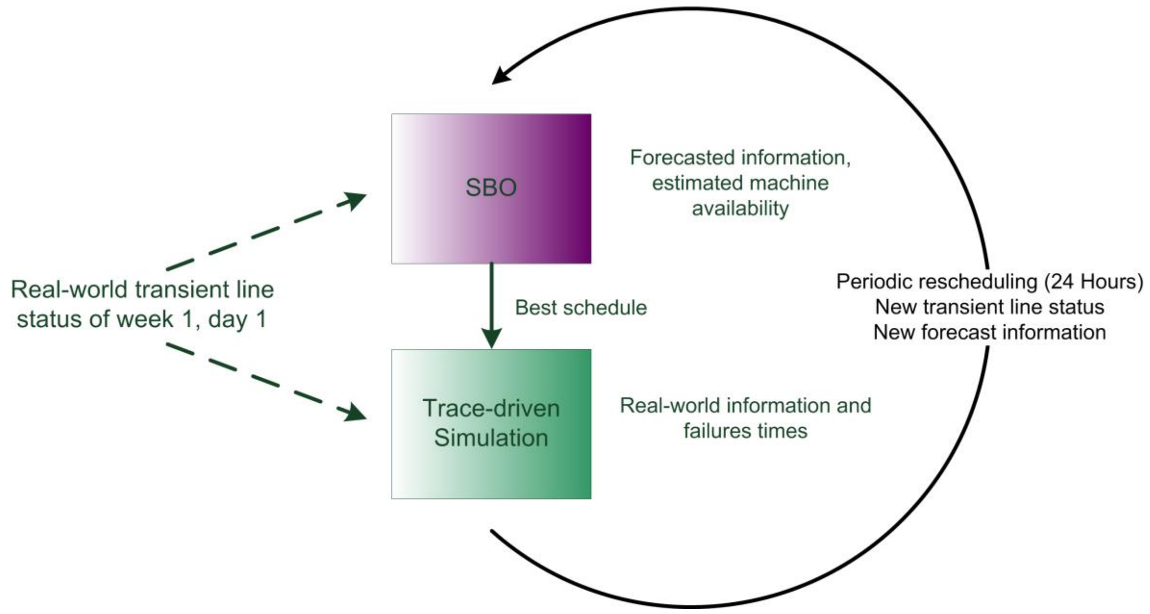

SBO
Forecasted information, estimated machine availability
Best schedule
Real-world transient line status of week 1, day 1
Periodic rescheduling (24 Hours)
New transient line status
New forecast information
Trace-driven Simulation
Real-world information and failures times

Figure 6.23 The procedure of the simulation environment rescheduling test.

The procedure of the rescheduling to test the realised schedule is as follows:

1. SBO uses the transient line status from the real world. The transient line status is the WIP information, current machine failures, possible changes, forecast information on demands, estimated machine availability, and so forth. This is done only when starting an experiment.
2. The best schedule is chosen and transferred to the simulation environment. The schedule reconfiguration program is used to make sure that possible deviations to schedule are taken care of.
3. The simulation in the simulation environment runs for one day (a periodic rescheduling is used and set to 24 hours). During this day, the simulation communicates with OIS just as if the actual PDAs were being used (expert suggestions, WIP messages, etc.). In the simulation environment, actual real-world data is used, which is in contrast to the optimisation where only forecasted and estimated data is used.
4. A forecast simulation is run using the current status of the simulation environment, but using forecasted machine availability instead.

5. Simulation-based optimisation is carried out using the transient line status from the simulation environment and "frozen jobs" from the forecast simulation. The forecast information is updated and the optimisation is started.
6. Repeat from step 2 until sufficient days have been tested.

Even though the system has the capability to carry out a rescheduling whenever needed, the real-world case study does not allow a full integration and automation of system data in the stages of proving the concept. Therefore, these tests were conducted on the assumption that a new schedule is created, according to the current procedure, once a day, i.e., every 24th hour. If rescheduling is done more frequently, or, even better, if it is triggered by events, then realised schedules can be generated more effectively. However, the periodic rescheduling each day will make no false assumptions that rescheduling may be carried out at any other times. While the SBO uses the ordinary simulation model to find good schedules, the trace-driven simulation is of another kind. The trace-driven simulation does not only use different data, it acts just as if it was the real-world system, regarding PDA communications. Each one of the machines in this simulation model will send a signal, whenever this is triggered, to the OIS, which will in turn send an expert suggestion back to the machine through the schedule dispatcher program. The reason for providing this functionality is to be able to validate the entire system in a single test run, when a rescheduling is carried out.

In order to clarify how the test works, an example replication of one of the schedule testing weeks in a simulation environment can be seen in Table 6.2.

Table 6.2 Simulation environment test.

Evaluation	**Shutdown time**	**Reached target levels**	**Target level precision (percentage)**
SBO Monday	Friday 2.30p.m.	7 out of 7	101%
SBO Tuesday	Friday 2.15p.m.	7 out of 7	100%
SBO Wednesday	Thursday 10.45p.m.	7 out of 7	100%
SBO Thursday	Thursday 10.45p.m.	7 out of 7	100%
SBO Friday	NA	NA	NA
Sim. environment	Thursday 9.30p.m.	7 out of 7	110%

Real-world result	Friday 11.30p.m.	5 out of 7	121%

The results from the first week of one of the replications show that the SBO application is able to find good solutions, even though the situation changes. As shown in the simulation environment, all production targets are reached (see Section 6.1.2), which in this case are also better than the real-world results. The actual demand was 26 % less for the whole week, compared to the forecasted demand on Monday, which is why there is such an over-production in both the simulation environment and the real line. The update on the Wednesday forecast seems to have the right volume and, therefore, the shutdown time can take place almost one full day earlier than forecasted. Each one of the optimisations generates a schedule for two weeks ahead, in order to avoid sub-optimal schedules. The simulation environment uses the SBO Monday schedule on Monday to Tuesday, the SBO Tuesday schedule on Tuesday to Wednesday, and so on. The simulation environment seems to have managed this week's schedule in a good way, even though long disturbances in excess of 20 hours from machine stops occurred and a direct approach was used which allowed no space for deviations from the sequence. Complete results from this evaluation can be found in Chapter 7.

6.13 Stage 5: PDA program validation

In the PDA evaluation, the whole system was used to generate schedules, distribute them through the PDAs, and evaluate the working procedure of the system. The test was executed off-line, i.e., not directly in the line, but in meeting rooms at the university or at the company. The organisation of the machines was setup in a *flow line production* which uses a *product layout* where production stages are arranged in sequence. Each production stage may consist of several parallel workstations. The parts, in this case bar code flags, were physically moved through the sequence of machines, of which each will process a small amount of the total work needed for the product to be completed, see Figure 6.24.

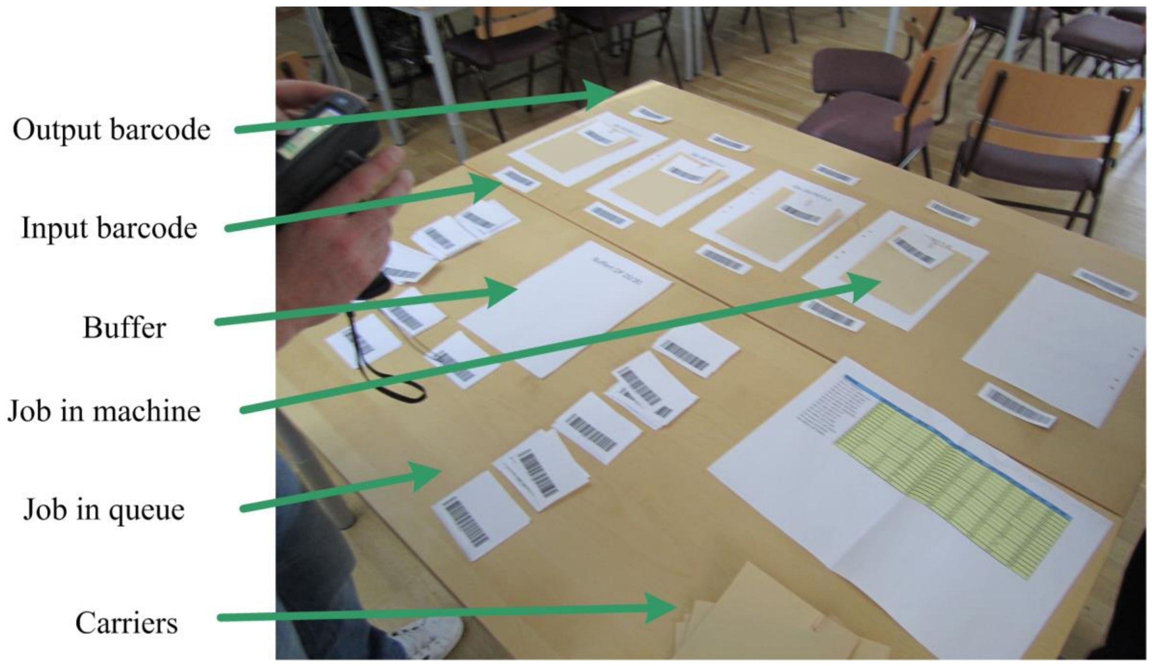


Figure 6.24 One production stage in a PDA-program validation test.

At each production stage, there is a buffer holding the parts that are available to be produced at each stage. In order to speed up the work procedure, bar codes of the workstations were used as well. Each workstation has one “input” bar code and one “output” bar code. Participants from different levels of the company, such as operator and IT consultants, and university staff were recruited to realise the tests. A clock was also implemented so that a time factor could be set to speed up time or pause it when necessary. The main target of such an evaluation was the working procedure with the PDA program. Overall, the result of the test was successful, even though some necessary changes were identified in the PDA program to make it more robust and intuitive. Real-time dispatching fault control of soft and hard constraints (Higgins, 1996) was identified as particularly important. Soft constraint violations, such as job selection deviation from expert suggestion, gave a warning sound and a popup window that needed to be confirmed. Attempting to violate the hard constraints, such as trying to start a job in an invalid machine, was successfully hindered by the PDA program. Furthermore, the users of the PDAs do not always do as they are told and handling those “disturbances” was identified as being important.

6.14 Stage 6: real-world validation

Realising the real-world validation was a huge project. About two years were needed to prepare just for this part, although the PDA implementation and validation was carried out in the meantime. The preparation was much about the IT-issues and implementation, since larger industries have comprehensive IT-regulations. Due to the fact that the OSS was a web-based system that uses both internal and external communication, the system had to be approved in several IT-gates, on the basis of architecture and integration reviews, in order to be authorized for on-line testing in the production line. When something in a production line is tested on-line, nothing can be left to chance, because causing production to stop could be very costly. Therefore, detailed fall-back plans were established that could be executed whenever needed, to ensure continued function in the system. Furthermore, several documents were created to support known “What-if scenarios”, i.e., instructions in how to handle certain situations. Additionally, responsibilities during the test were assigned to different personnel.

The test was finally executed in the real-world production line during a production week, and the results were mainly good. The operators had never had such good support from the PDAs and the production monitoring programs. How the PDAs handled the soft and hard constraints was positive and the operators and shift leaders were impressed by the line status program. Furthermore, the schedules generated were good and the operators could successfully use them. However, the fact that it was not possible to accomplish full system integration, due to technical constraints of the machine equipment, resulted in a delay of disturbance information. Rescheduling was therefore sometimes started too late, because the manual input of data made the predictive schedule inaccurate when a direct approach was used. Consequently, the main deficiency of this implementation was the integration with the shop floor system. Apart from the integration, the system as a whole worked well: the SBO quickly delivered good and valid schedules and the OIS was responsive and accurate.

There are alternatives to solving the issue of integration with the shop floor system. The most important information that is missing in the implementation of OSS is the failure of machines data. A full integration with the machines would be desirable, but additional information could be needed anyway. An operator should be able to inform the system about the estimated time of failure, in the event of a more serious failure, so that the system can take this into account when generating a new schedule. Therefore, a future simple solution to the problem could be to add this task to the PDA-program.

6.15 Concluding remarks

The deployment of the system architecture, as proposed in Chapter 4, has demonstrated the applicability of the architecture and given further insight into industrial real-world scheduling problems. For example, the stage to test machine schedules identified the importance of grouping jobs, which is one of the contributions described in the optimisation algorithm in Chapter 5 and furthermore tested in Chapter 7. The deployment of the system in a real-world environment also confirms the importance of supporting the real-time scheduling task, rather than only generating a schedule in off-line mode. Furthermore, various user-interfaces were identified as important in Chapter 3 and have been demonstrated necessary here to support the scheduling task. For example, shop floor nervousness has been avoided by distributing new schedules electronically, with consideration to the previous schedule. Additionally, the deployment stages of the scheduling system OSS were described, in order to explain how and what stages were executed to implement the OSS architecture. Furthermore, the reactive scheduling stage was identified as important, the results of which are presented in Chapter 7.

Chapter 7

7 Quantitative Results and Analyses

This chapter presents the experimental results of applying the hybrid genetic representation, described in Chapter 5, to the real-world, crankshaft machining line case study presented in Chapter 6. All of the results included in this chapter were obtained from the OSS implementation on the real machining line.

The chapter is divided into four main parts: (1) general optimisation problem formulation; (2) setup reduction experiments; (3) predictive scheduling results, and (4) reactive scheduling results. The first section addresses the general experimental settings applied to various scheduling scenarios. Thereafter, the results of the setup reduction experiments through the mutation operators embedded in the optimisation method are presented. In the predictive scheduling, different scheduling approaches for finding good schedules are analysed in detail. Lastly, the results of the reactive scheduling are analysed and compared to the real-world findings.

7.1 General settings

Different experiments have different purposes, but a description of the main shared settings follows. In addition, further settings specifically assigned to each experiment are described under each sub-section.

7.1.1 Weighted-sum objective function

Recall that the real-world scheduling case study, described in Chapter 6, is by nature a multi-objective and multi-properties scheduling problem of a hybrid flow shop with parallel machines and by-pass. Therefore, a weight-based objective function was used, because the production scheduler needed to obtain the result quite quickly and the user had no time to study separate sub-targets. According to Kempf et al., (2000), the scheduling objectives will and should vary from one organisation to another. However, the scheduling objectives presented here are variants of commonly used objectives, such as makespan, throughput rate, and tardiness. Fitness is the sum of all the sub-targets' fitness values.

$$F = f_{th} + f_{sh} + f_{tl} + f_{sa} + f_{st} \tag{7.1}$$

where, F is fitness, f_{th} is the fitness of Throughput, f_{sh} is the fitness of Shortage, f_{tl} is the fitness of the Target Levels, f_{sa} is the fitness of Stopped in Advance, and f_{st} is the fitness value of the Setup Time. Throughput is the number of products produced per hour measured for the whole simulation period. It is important to increase the overall throughput of the system. Apart from the mean throughput value, the fitness function needs to take its standard deviation into account, because a high variability could cause losses. Therefore, the objective function for throughput (f_{th}), Equation (7.2), was created according to the values of experts of the system.

$$f_{th} = w\left(k\sigma + \left(\tau - \mu\right)\right) \tag{7.2}$$

where, w is the objective weight for Throughput, μ is the mean value of Throughput, τ is the target level of Throughput, σ is the standard deviation, and k is the weight of the standard deviation. An essential objective for the machining line was maintaining the security levels of the variants in the finished goods inventory (FGI). If a variant's FGI level falls under the security level, it results in a momentary shortage in equation (7.3).

$$ms = \left(\frac{cl - sl}{sl} \times 100\right)^2 \tag{7.3}$$

where, ms is a momentary shortage, cl is the current stock level, and sl is the security level of one product variant. All momentary shortages are totalled and, at the end of the simulation, the variant shortage is calculated.

$$vs = \sqrt{\frac{\sum_{i=1}^{n} ms_i}{n}} \tag{7.4}$$

where, vs is variant shortage and n is the total number of measures carried out. A robust schedule is requested, so based on a quadratic loss function (Sanchez, 2000), the fitness of shortage in Equation (7.5) is a good approximation in the representation of this objective, as it penalises small values of deviation to target minimally, but penalises large ones considerably.

$$f_{sh} = \sum_{i=1}^{n} w_i \left(\sigma_i^2 + \mu_i^2 \right) \tag{7.5}$$

where, μ is the performance mean and σ is the standard deviation, w is the product variant's weight, and n is the total number of product variants. The fitness of target levels measures the ability to reach necessary targets for different product variant groups. Target levels are measured in FGI once a week and the best result is when a variant is equivalent to the target level. A measured level that is less than the target level is given a higher penalty than one that is over the target level.

$$f_{tl} = \sum_{i=1}^{n} w_i \left(\sigma_i^2 + \mu_i^2 \right) \tag{7.6}$$

where, μ is the performance mean, σ is the standard deviation, n is the number of target levels, and w is penalty weight, which depends on whether the measured value is above or below the actual target level. Stopped in advance is based on when the production is shut down each week, and is referred to as makespan, since it is the same thing. Machines can shut down when the corresponding shortage and target levels that the machine produces are reached for the week.

$$f_{sa} = -w\mu \tag{7.7}$$

where, w is the objective weight and μ is the performance mean of ´Stopped in Advance´ of all machines. The fitness of setup time is included, since it is important to reduce the total setup time because it will reduce the workers' time at the machine.

$$f_{st} = w\left(\frac{\mu_{st}}{\mu_{st} + \mu_w}\right) \tag{7.8}$$

where, w is the objective weight, μ_{st} is the mean value of the total setup time of all machines, and μ_w is the mean value of the total processing time of all machines.

By comparing the graph of the fitness value with the graphs of the different objectives, i.e., throughput, shortage, target levels, makespan, and setup, it is possible to see how the optimisation is carried out, see Figure 7.1.

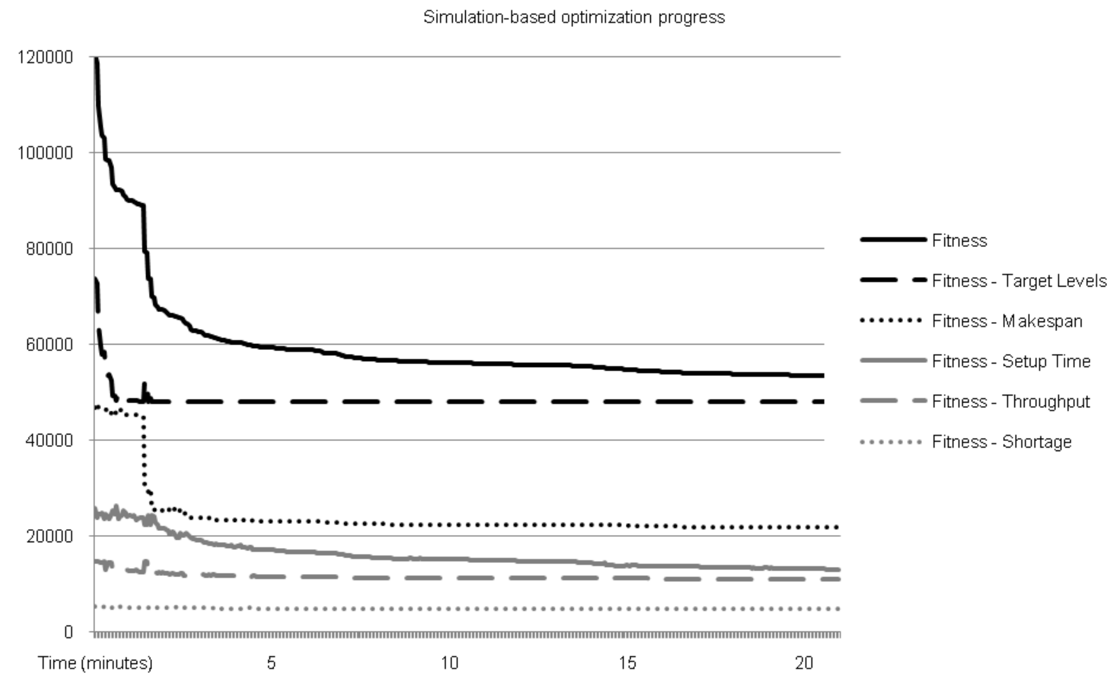


Figure 7.1 Simulation-based optimisation progress.

At first, the optimisation focus is on producing the right variants at the right time by decreasing the shortage, in order to reach target levels and at the same time minimise makespan, the focus thereafter is on increasing the throughput and minimising setup time. In this Scenario, the shortage is at very low levels, which makes the optimisation focus on the other objectives. The setup time objective is mostly in conflict with all other objectives, because it tries to produce as many product variants of the same type, consecutively. It is in conflict with the shortage and target levels, because these objectives usually need a schedule where the variants are in accordance to current demand, which could be a very mixed order. The productivity of a bottleneck stage may benefit from the setup reduction, but system throughput per hour usually needs a specific variant mix, in order to utilise the system in an efficient way, which would make throughput a conflicting objective to setup reduction as well.

7.1.2 Scheduling scenarios

Two scenarios based on real-world data were tested, in order to study how the optimisation methods perform in different circumstances. There are two main product groups, namely, product group A and product group B (see Section 6.1.2), that mostly do not share the same type of resources. The first Scenario is a typical, stable production state in which the product mix is fairly low. The theoretical system load in Scenario 1 is about 80% for the two production weeks. The system load is calculated on the number of parts divided by the capacity of the primary bottlenecks. The first week has a system load of 90% and the other a load of 70%. Only a few batches (6%) have a short deadline, i.e., deadlines for already started jobs (WIP) within one day and jobs not yet started within two days, and there are no rush orders, i.e., deadlines for jobs not yet started within one day. Products with short deadlines are common, due to sudden demand changes or quality problems. It is simply Work-In-Process (WIP) or material that needs to be somewhat prioritised, in order to be produced on time. Rush orders are rare, e.g., they can occur when large numbers of parts of the same product variant need to be scrapped or reworked, due to quality issues. Rush orders must be prioritised or there will be a deficit of the product in the Finished Goods Inventory (FGI). The theoretical system load calculated on the primary bottlenecks in Scenario 2 is approximately 82.5% for the two production weeks. The first week has a system load of 95% and the other has a load of 70%, which is normal and due to demand fluctuations. The second Scenario is a variant of the first one, but has more products with a short deadline (10%) and some products (4%) that are rush orders in the first week. This has been accomplished by increasing the demand of some variants in product group A, which also indirectly results in a higher mix of product variants over a short-term period. The second Scenario is common in production for several reasons, e.g., demand fluctuations actually increase demand, there are scrapped parts or blocked parts that need rework, and are therefore important and relevant for testing as an additional scheduling Scenario. The shorter deadlines of some product variants may cause additional sequence-dependent setups, since there may be no time to avoid setups by producing larger batches and at the same time reaching the target levels or avoiding shortage of the product variants. Both scenarios simulate two weeks of production, in order to avoid sub-optimal schedules for a short time period.

7.1.3 General optimisation settings

The optimisations of each optimisation method, described in Chapter 5, used ten replications in order to obtain reliable results. For hypothesis testing, the Mann-Witney test was chosen instead of the 2-sample t-test, due to the relatively small number of replications and since it is not known whether the results are normally distributed. The exact results of the hypothesis tests can be seen in Appendix F. Each replication used the maximum time limit of 20 minutes, which is a reasonable requirement with regard to the scheduling of these types of problems, e.g., when the operating time of a job ID (batch) in a processing step is from about 20 minutes. The requirement will make sure that not much has happened in the production line during the scheduling. Although the optimisation itself is stochastic, each evaluation in the simulation model is deterministic. The genetic algorithm settings are a mutation rate of 20%, crossover rate of 80%, and population size of 50 individuals. Each mutation type has the same chance of being chosen and there is a 50% chance of doing another mutation after a successful mutation.

7.2 Results with setup-reduction mutation

In these experiments, the results show the importance of good genetic algorithm operators when using a direct approach. The operators, variant-grouping and setup-grouping, which have been tested here, are described in detail in Chapter 5. The two different optimisation methods used are permutation schedule (PS) and non-permutation schedule (NPS), in order to be able to see how much they affect the fitness objectives. The experiment includes the following:

- Scheduling scenarios: Two different scheduling scenarios, as previously described.
- Optimisation methods: permutation schedule (PS) and non-permutation schedule (NPS).

- Mutation type: “single batch”, “variant groups” and “setup groups”. The single batch mutation is the most straightforward implementation, since it will try to move one batch at a time. The variant groups have been implemented, because batches of the same product variant can be grouped into larger batches to reduce setup times and to prevent frequent changes of the product variant occurring at the machines. The setup groups are similar to the variant groups, with the difference that larger numbers of batches belonging to the same product family are grouped together. The setups between different product families are typically longer than the setups between product variants within the same product family. See Chapter 5 for more information.

The results of both scenarios are presented in the following way: “mean value (standard deviation)”. The results of Scenario 1 can be seen in Table 7.1.

Table 7.1 Setup reduction for scenario 1.

Scenario 1	Single batch		Single batch, variant groups		Single batch, variant groups, setup groups	
	PS	NPS	PS	NPS	PS	NPS
Shortage	3(5)	3(3)	76(59)	50(47)	70(53)	52(37)
Target levels	0(0)	6(14)	0(0)	0(0)	0(0)	0(0)
Makespan	-27436(593)	-26471(410)	-30439(390)	-29228(468)	-30441(711)	-29720(875)
Setup time	11808(613)	12677(485)	9442(602)	10888(769)	9416(447)	10571(905)
Throughput	19867(524)	20697(315)	16821(1142)	18523(495)	16773(881)	18120(999)
Total fitness	4244(1392)	6914(928)	-4098(885)	233(960)	-4179(1188)	**-975(1387)**

Both the PS and NPS show they are able to reach the targets in Scenario 1, using the single batch mutation. When the variant group mutation is used together with the single batch mutation, a huge improvement can be observed. The makespan, setup time, and throughput are much improved. The last mutation type setup groups were tested together with variant groups and single batch mutations. This only shows a small improvement compared to the variant groups and single batch experiment. However, the setup group mutation is significantly better than variant group mutation, when using the NPS optimisation method, see Appendix F. The results of Scenario 2 can be seen in Table 7.2.

Table 7.2 Setup reduction for scenario 2.

Scenario 2	Single batch		Single batch, variant groups		Single batch, variant groups, setup groups	
	PS	NPS	PS	NPS	PS	NPS
Shortage	1149(760)	1679(1204)	537(192)	720(213)	500(217)	513(284)
Target levels	9952(5806)	18116(5813)	407(180)	4579(1604)	364(126)	911(986)
Makespan	-1446(87)	-646(557)	-1729(118)	-1205(366)	-1775(98)	-1562(184)
Setup time	23764(1664)	27381(1196)	12799(624)	19812(1298)	10414(969)	17506(1532)
Throughput	25306(462)	26218(316)	21117(464)	25032(719)	20140(813)	22980(1098)
Total fitness	58725(7195)	72747(6289)	33131(1001)	48939(3059)	**29644(1099)**	**40350(3430)**

Both the PS and NPS show they are not able to reach the targets in Scenario 2, using the single batch mutation. When the variant group mutation is used together with the single batch mutation, a huge improvement can be observed. All of the objectives have been improved, but the target levels and setup time show the greatest improvements. The last mutation type setup groups were tested together with variant groups and single batch mutations. The setup group mutation is significantly better than variant group mutation for both PS and NPS.

7.2.1 Analysis

The two types of mutation variant groups and setup groups show a significant improvement in the results. The mutations reveal a great improvement with regard to both Scenario 1 and Scenario 2. The results demonstrate that Scenario 1 has been improved from an already acceptable level. PS and NPS do not reach their targets when only the single batch mutation in Scenario 2 is used. Consequently, further complexity of the scheduling Scenario increases the need to use variant group mutation and setup group mutation. Overall, the setup group mutation is significantly better than the other two mutation operators, probably because the search space is efficiently reduced, without constraining it, which leads to reaching good solutions via a shorter path.

7.3 Results of the predictive scheduling

In the predictive scheduling, different scheduling approaches for finding good schedules are evaluated. However, an analysis of why a certain approach is better than another is also evaluated with the help of bottleneck analyses, performance graphs, and dispatching rule frequency analysis.

7.3.1 Bottleneck analysis

The bottleneck analysis is carried out to facilitate analysing the results of the different approaches and their attendant optimisation methods. The optimisation methods used in the bottleneck analysis were PDR and PS, in order to obtain a reasonable fair resource utilisation of parallel machines at the production stages. Ten replications were used for each of the scheduling scenarios and the optimisation methods. In addition, each scheduling Scenario was run for two weeks, which is a really short time period, when it comes to a bottleneck analysis. However, generating a short-term period schedule is the result and, therefore, the short-term period bottlenecks are of interest. The bottleneck analysis used here comprises two methods: (1) active duration and (2) shifting bottleneck analysis (Roser et al., 2002). The notation "production stage: parallel workstation number" is used, i.e., 1:1 means production stage 1 and parallel workstation number 1. In Scenario 1, the active duration shows that production stage 1 is probably the bottleneck stage, see Figure 7.2.

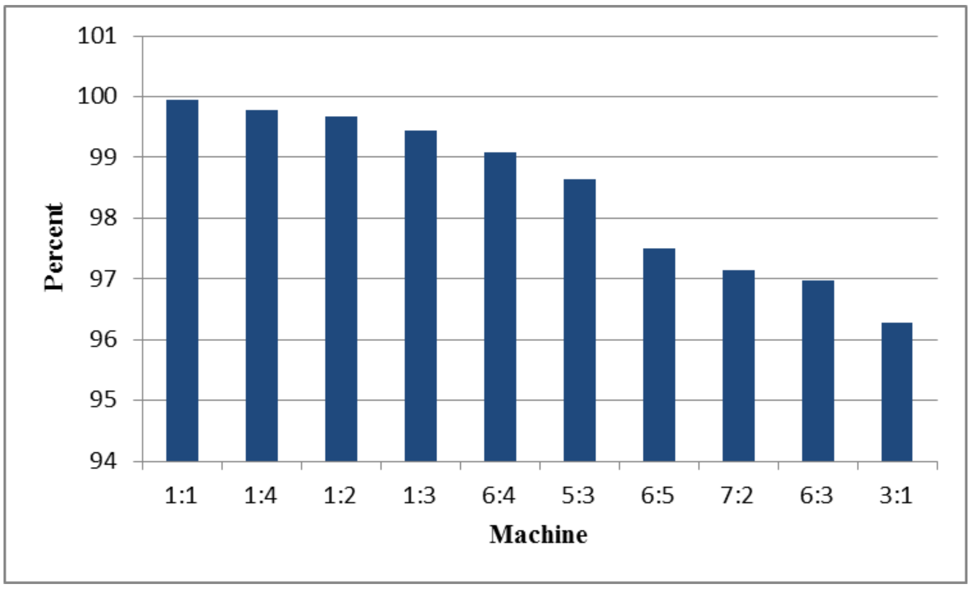


Figure 7.2 Active duration of scenario 1.

Furthermore, the shifting bottleneck analysis reveals that the bottlenecks are shifting bottlenecks most of the time and sole bottlenecks only a fraction of the time. It also confirms that production stage 1 is the primary bottleneck and, in particular, workstations 1 and 4, see Figure 7.3.

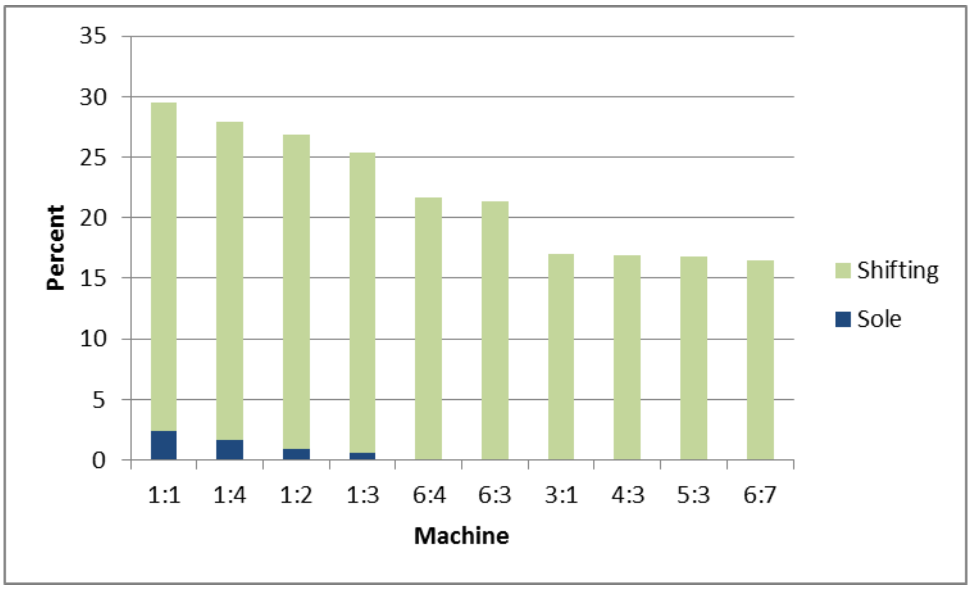


Figure 7.3 Shifting bottleneck analysis of scenario 1.

Both these types of analysis identify almost the same bottlenecks, although the order is different. The two main product variant groups, A and B, are of particular interest here. Two main product groups usually do not share resources, but this is an ever-changing process, since a demand change six months ahead could mean that more resources need to be shared. Therefore, the bottleneck analysis is important for a scheduling application, since balancing and setups may change the bottleneck situation completely. However, workstations 1:1 and 1:4 are configured to process product variants of group A and workstations 1:2 and 1:3 can only process product variants of group B. One could therefore conclude that the production of product variants in group A is a limiting factor. This can be further strengthened by the fact that the next four workstations on the active duration graph all produce group A products. Workstations 6:4 and 4:3 produce both group A and group B product variants, while workstations 5:3, 6:5, and 6:6 only produce group A product variants. Production stage six is identified as a possible secondary bottleneck stage. On the other hand, it has three parallel resources and, therefore, stage five may be a more sensitive point or, more specifically, workstation 5:3. In Scenario 1, production stage 1 is the primary bottleneck for both product group

A and product group B, mainly because the workstations have a high level of utilisation due to long processing times.

In Scenario 2, product group A demand is increased and some of the orders are rush ones. The result is similar, although the order of some of the workstations is different. Production stage 1 is no longer the primary bottleneck, as it is in Scenario 1, see Figure 7.4.

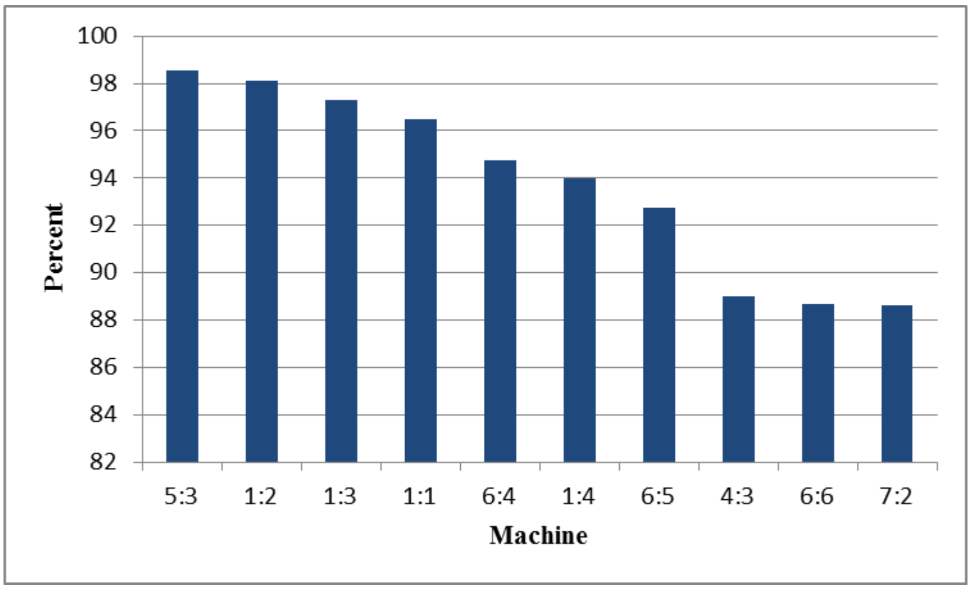


Figure 7.4 Active duration of scenario 2.

The active duration shows that 5:3 seems to be the primary bottleneck, followed by production stages one and six. The outcome of the shifting bottleneck analysis shows similar results in Figure 7.5.

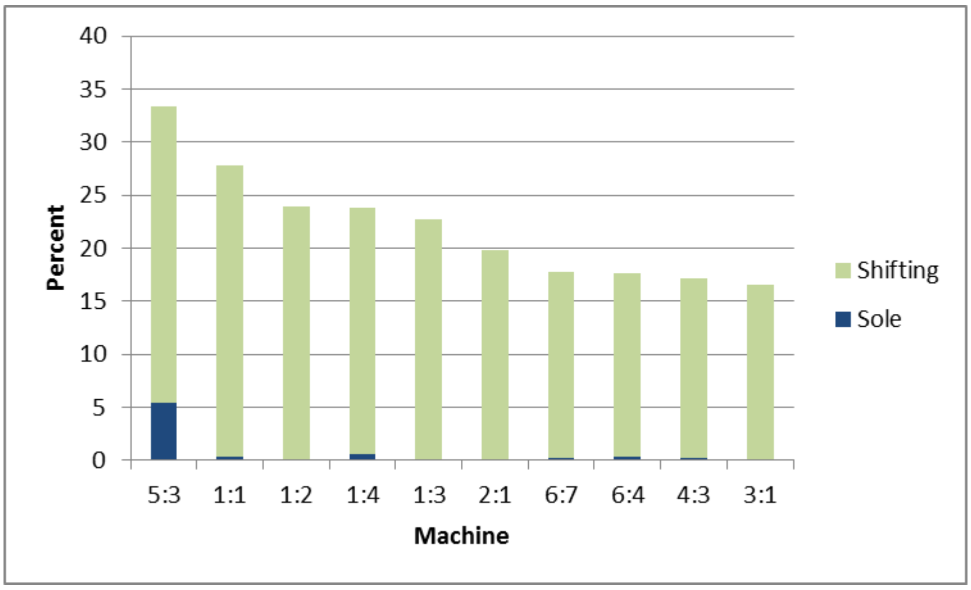


Figure 7.5 Shifting bottleneck analysis of scenario 2.

The shifting bottleneck analysis identifies workstation 5:3 as the primary bottleneck, followed by production stage one. Production stage one is the secondary bottleneck for the same reason as in Scenario 1, namely, long processing times. Production stage five, or more specifically workstation 5:3, is the primary bottleneck in Scenario 2, mostly for another reason. The workstation has long, sequence dependent setup times, partly because 10% of the orders have short deadlines, but mostly because 4% are rush orders. The bottleneck analyses of the two scenarios are important, when the experiments of the different optimisation methods are entered.

7.3.2 Different optimisation methods

Different optimisation methods are tested, in order to identify which one performs best with regard to two different scenarios. The optimisation methods for each approach are as follows:

- Direct approach:
    - Permutation schedule (PS)
    - Non-permutation schedule (NPS)
- Indirect approach:
    - Priority dispatching rules (PDR). The full set of results can also be found in Andersson (2011).
- Hybrid approach:

- Direct approach in the first stage and priority dispatching rules in the other stages. (HYB)
- Free hybrid, in which the direct approach or indirect approach is chosen by the optimisation algorithm for all stages (FHYB).

7.3.3 Results of different optimisation methods

The results of both scenarios are presented in the following way: "mean value (standard deviation)". All averages and standard deviations are rounded to an integer value. The results of Scenario 1 can be seen in Table 7.3.

Table 7.3 Results of scenario 1.

	PS	NPS	PDR	HYB	FHYB
Shortage	70(53)	52(37)	99(127)	106(99)	32(16)
Target levels	0(0)	0(0)	329(67)	137(64)	35(54)
Makespan	-30441(711)	-29720(875)	-29415(456)	-31528(93)	-30934(591)
Setup time	9416(447)	10571(905)	17937(845)	11027(1139)	9964(919)
Throughput	16773(881)	18120(999)	12281(272)	10764(676)	10696(544)
Total fitness	-4179(1188)	-975(1387)	1233(830)	-9493(1773)	-10206(1114)

The results of Scenario 1 show that FHYB is the best among the different optimisation methods, but not with a statistically significant difference compared to HYB. It performs well on all sub-objectives, but is surpassed by the permutation scheduling (PS) with regard to setup time and by HYB with regard to makespan. All optimisation methods demonstrate quite good results in Scenario 1, with regard to shortage, target levels, and makespan. The results of Scenario 2 can be seen in Table 7.4.

Table 7.4 Results of scenario 2.

	PS	NPS	PDR	HYB	FHYB
Shortage	500(217)	513(284)	67342(34514)	3252(2471)	607(248)
Target levels	364(126)	911(986)	31157(14855)	10235(1903)	779(242)
Makespan	-1775(98)	-1562(184)	-1612(155)	-1710(53)	-1754(91)
Setup time	10414(969)	17506(1532)	31035(5886)	21726(1482)	11654(1640)
Throughput	20140(813)	22980(1098)	21859(2321)	21280(288)	19129(246)
Total fitness	29644(1099)	40350(3430)	149783(33409)	54783(5337)	30417(1832)

The results of Scenario 2 show that PS is the best among the different scheduling methods, but not with a statistically significant difference compared to FHYB, see

Appendix F. However, FHYB is significantly better than PS, when these are compared to Scenario 1. FHYB is the most successful optimisation method overall, followed by the PS algorithm that performs well on the more difficult scheduling in Scenario 2.

7.3.3.1 Direct scheduling approaches

When comparing the two methods of the direct scheduling approaches, it is obvious which method is the best one. Permutation (PS) scheduling outperforms non-permutation scheduling (NPS) on total fitness. In Scenario 1, PS performs better because it is superior regarding makespan, setup time, and throughput. In Scenario 2, the difference between PS and NPS is even greater. Whilst both of them succeed in avoiding a large shortage and reach the target levels, PS outperforms NPS on the other sub-objectives. The reason why PS is better than NPS is that NPS has a much larger search space, since the order of batches needs to be changed at all production stages. PS only needs to change the order at the first production stage, which makes it beneficial for a complex scheduling problem such as this, due to the time constraints.

7.3.3.2 Indirect scheduling approach

The indirect scheduling approach optimisation method of priority dispatching rules (PDR) does not perform very well in either of the two scenarios. In Scenario 1 it obtains good results on all sub-objectives except setup time. This is quite natural, since dispatching rules usually do not have the same ability to group product variants into larger batches and wait for the right product variants to arrive, as a direct approach. The PDRs utilise the workstations quite extensively and obtain a really good throughput. When it comes to Scenario 2, which is a slightly tougher Scenario, the PDRs really fail in generating a good schedule. For the same reason, they are not able to reduce setup time in Scenario 1, they also fail in prioritising rush orders, which finally also makes them fail in shortage and reaching target levels. Dispatching rules are local at the workstations, making them functional for utilising workstations, but they therefore also suffer from not being able to act in a more global perspective.

7.3.3.3 Hybrid approaches

The HYB optimisation method of the hybrid approach uses a direct schedule at the first production stage and dispatching rules at the following stages, which is interesting since it combines the strengths of PS and PDR. In Scenario 1, HYB is the second best scheduling method after FHYB, although there is no significant difference between the two methods. HYB seems to utilise the workstations and produces a high throughput, but it is worse than PS and NPS with regard to setup time, although HYB maintains it at an acceptable level. With regard to Scenario 2, HYB shows its inability to prioritise batches in the line, similar to the PDR approach. Furthermore, it does not reach the target levels, obtains high shortage, as well as long setup times.

In the FHYB optimisation method of the hybrid approach, the optimisation was able to freely select suitable representation (direct or indirect) at each production stage. FHYB generated the best schedules overall, compared to all the other methods. The reason seems to be that the optimisation algorithm may decide for itself where to make a detailed schedule of the direct representation and where to make an indirect schedule with dispatching rules. Therefore, it is able to focus its powers where necessary, e.g., at a bottleneck stage.

7.3.4 Dispatching rule frequency

In this section, there is an analysis of the type of dispatching rules selected in the different optimisation replications. The frequency tables show the results of both Scenarios divided into the different production stages. This analysis was only made for PDR, HYB and FHYB, since only these approaches end up with a set of rules including sequence (SEQ) as one of those rules. The dispatching rules are sorted in descending order starting with the most frequently used on the left side. The number of times a dispatching rule is used is displayed within the parenthesis. The different rules used for these experiments are sequence (SEQ) for HYB and FHYB, as well as all the different dispatching rules used by all three methods, namely: “First Come First Served” (FCFS), “Earliest Due Date” (EDD), “Shortest Processing Time” (SPT), “Longest Processing Time” (LPT), and “same Variant lowest Setup time earliest Due date” (VSD).

7.3.4.1 PDR

The PDR method selected FCFS in the first production stage for Scenario 1, see Table 7.5. The reason is probably that all jobs are sorted by deadline and product variant, as the initial solution. Consequently, FCFS uses this exact order. Production stages 2 and 4 have many different types of rules and the reason is that those stages are not as important as the other ones. Production stages 3, 5, 6, and 7 seem to be dominated by the VSD rule which prioritises those jobs of the same type or low setup time and is secondary on deadline. The VSD will reduce the setup time, but it will also be able to finish on time, since Scenario 1 is fairly easy.

Table 7.5 Dispatching rules frequency PDR scenario 1.

Stage 1	FCFS(10)			
Stage 2	FCFS(3)	EDD(3)	LPT(3)	VSD(1)
Stage 3	VSD(7)	LPT(2)	EDD(1)	
Stage 4	FCFS(3)	SPT(3)	EDD(2)	VSD(2)
Stage 5	VSD(10)			
Stage 6	VSD(10)			
Stage 7	VSD(9)	LPT(1)		

Scenario 2 is harder to schedule, since there are rush jobs that need to be finalised as soon as possible. FCFS was selected for production stage 1, for the same reason as in Scenario 1, see Table 7.6. Production stages 2, 3, and 4 are not that important, although setups seem to be avoided in stage 3. The length of the setup times in production stage 3 is not a big issue; however, by grouping the batches, setups may be avoided at other production stages, e.g., production stage 6 which could be affected by the order from stage 3. Production stage 5, which is the bottleneck stage in this Scenario according to the bottleneck analysis, seems to struggle with conflicting objectives. In 6/10, it selects EDD in order to prioritise the rush orders and in 4/10 it tries to avoid long, sequence dependent setup times by using the VSD rule.

Table 7.6 Dispatching rules frequency PDR scenario 2.

Stage 1	FCFS(10)				
Stage 2	FCFS(4)	LPT(2)	SPT(2)	EDD(1)	VSD(1)
Stage 3	VSD(6)	EDD(2)	LPT(1)	SPT(1)	
Stage 4	LPT(6)	VSD(3)	FCFS(1)		
Stage 5	EDD(6)	VSD(4)			
Stage 6	LPT(6)	VSD(3)	FCFS(1)		
Stage 7	LPT(5)	SPT(4)	FCFS(1)		

7.3.4.2 HYB

The HYB method is the same as the PDR approach, with the exception that SEQ was selected for the first production stage. In Scenario 1 (Table 7.7), production stages 2 and 4 are not as important as the other stages. Production stage 3 seems to benefit from reducing the number of setups or keeping batches together, by using the VSD rule. Production stage 5 has VSD and EDD as the most frequently used rules. Whilst VSD reduces setups, EDD may help in prioritising the right kind of jobs, in order to obtain a shorter makespan. Production stages 6 and 7 are dominated by the VSD rule.

Table 7.7 Dispatching rules frequency HYB scenario 1.

Stage 1	SEQ(10)		
Stage 2	EDD(6)	SPT(3)	LPT(1)
Stage 3	VSD(10)		
Stage 4	SPT(8)	FCFS(2)	
Stage 5	VSD(5)	EDD(4)	FCFS(1)
Stage 6	VSD(10)		
Stage 7	VSD(10)		

In Scenario 2 (Table 7.8), the sequence (SEQ) in production stage 1 determines the order of jobs to be started. In production stages 2 and 4, the algorithm has a hard time determining the rules. The variety of rules selected for production stage 3 in Scenario 2 is rather hard to explain, but is an effect that can be observed for all optimisation methods. Production stage 3 is not a bottleneck stage, but the rules were probably selected to support other production stages. Production stage 5 is a bottleneck stage in Scenario 2 and EDD was selected in order to ensure the right jobs are chosen when they queue up in front of the workstations.

Table 7.8 Dispatching rules frequency HYB scenario 2.

Stage 1	SEQ(10)			
Stage 2	FCFS(5)	EDD(2)	LPT(2)	SPT(1)
Stage 3	LPT(7)	VSD(3)		
Stage 4	LPT(4)	VSD(4)	FCFS(1)	SPT(1)
Stage 5	EDD(10)			
Stage 6	LPT(7)	VSD(3)		
Stage 7	VSD(9)	FCFS(1)		

7.3.4.3 FHYB

The FHYB method has the choice of freely selecting between all dispatching rules including the SEQ. In Scenario 1 (Table 7.9), the bottlenecks in production stage 1 have selected SEQ as the dispatching rule, since the sequence is able to control what jobs start on the line in a good way. Production stage 3 also seems to benefit from the sequence, as it was selected in 7/10. Production stages 5, 6 and 7 benefit from selecting VSD as the primary rule, since it will reduce the total setup time at those stages.

Table 7.9 Dispatching rules frequency FHYB scenario 1.

Stage 1	SEQ(10)				
Stage 2	VSD(3)	EDD(2)	FCFS(2)	SPT(2)	LPT(1)
Stage 3	SEQ(7)	VSD(3)			
Stage 4	LPT(5)	FCFS(2)	EDD(1)	SPT(1)	VSD(1)
Stage 5	VSD(7)	SEQ(3)			
Stage 6	VSD(9)	FCFS(1)			
Stage 7	VSD(10)				

In Scenario 2 (Table 7.10), SEQ was selected for production stage 1 for the same reason as before. However, what is most remarkable is that SEQ was selected for production stage 5 in 9/10 replications. In the case where VSD was selected for stage 5, the other production stages upstream compensated that by having SEQ selected. SEQ is able to reduce the setup whilst making sure that the right product variants, i.e., rush orders, are prioritised.

Table 7.10 Dispatching rules frequency FHYB scenario 2.

Stage 1	SEQ(10)			
Stage 2	FCFS(4)	VSD(3)	SPT(2)	SEQ(1)
Stage 3	SEQ(5)	LPT(4)	VSD(1)	
Stage 4	LPT(8)	SEQ(1)	SPT(1)	
Stage 5	SEQ(9)	VSD(1)		
Stage 6	LPT(9)	VSD(1)		
Stage 7	VSD(5)	LPT(4)	FCFS(1)	

7.3.5 Performance analysis

When it comes to a real-world scheduling problem, there is limited time in which a schedule must be found. The various optimisation methods converge towards their best solutions at a different pace. Figure 7.6 shows the optimisation progress of the different optimisation methods in Scenario 1.

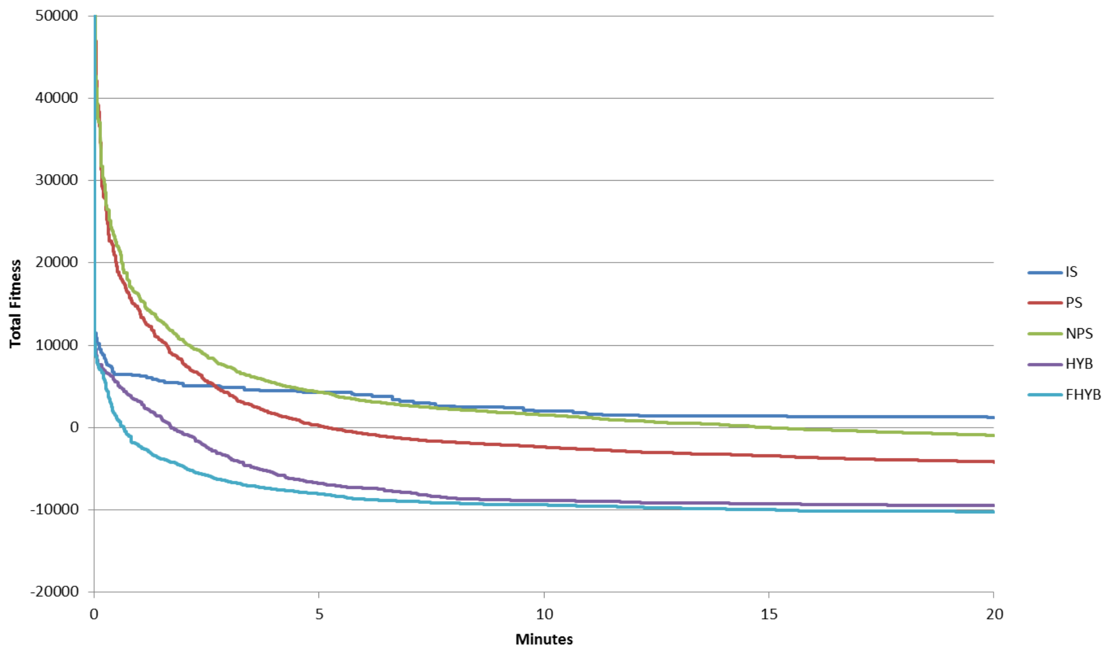


Figure 7.6 Algorithm performance in scenario 1.

There are two clear groups of methods: (1) those that only use the direct approach and (2) those that use the indirect approach in some or all of the production stages. The latter group obtains better initial solutions, due to the use of dispatching rules that dramatically decrease the search space. When it comes to the first group, the permutation schedule (PS) converges faster towards good solutions compared to the non-permutation schedule (NPS). Both direct approaches generate a better schedule compared to the priority dispatching rules approach (PDR) which seems to cease its improvement after the first rapid improvement down to a total fitness of 7000. The two remaining approaches, the hybrid (HYB) and the free hybrid (FHYB), are both fast at the beginning and continue to converge towards better solutions. With regard to Scenario 2, the optimisation methods perform differently, see Figure 7.7.

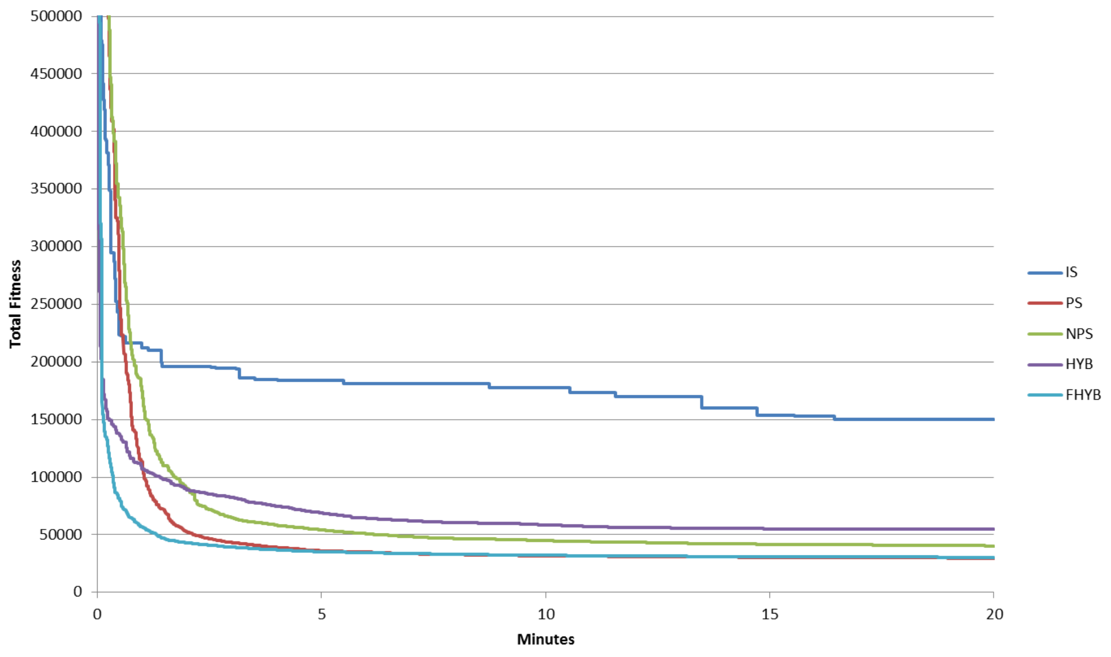


Figure 7.7 Algorithm performance in scenario 2.

Among the approaches, the direct approach is still the worst one at the beginning of the optimisation run, but PS and NPS improve quite rapidly, compared to most of the other methods. PDR is clearly the worst method, probably due to its inability to prioritise rush jobs and reduce setup times. HYB converges towards better solutions, but do not reach satisfactory levels of the different optimisation goals. NPS obtains good solutions for all the objectives except setup time, which is quite long compared to PS and FHYB. The PS and FHYB optimisation methods are the two which obtain the best results after 20 minutes. In the first five minutes, FHYB is faster because it is able to use dispatching rules and therefore concentrates its resources at the bottlenecks.

It seems that combining the direct and the indirect approaches, i.e., to form a hybrid approach, is successful. Particularly the FHYB method is good, since it seems to be able to handle different scenarios in different ways and use the detailed scheduling (direct representation) where it is needed. However, the direct approaches still obtain good results in both scenarios and, in particular, the PS method, since it seems to reduce the search space into what is feasible with a limited time period. The direct approaches seem to be good when some jobs need to be prioritised, e.g., rush orders.

Consequently, FHYB is fast in converging towards good solutions and efficiently focusing on the bottleneck stages. FHYB's flexibly can handle that bottlenecks shift over time.

7.4 Results with reactive scheduling

How to measure a schedule is based on Kempf et al., (2000) where several issues are discussed in order to obtain a good schedule. Here, a *dynamic measurement* is used where the predictive schedule is tested in the real environment with regard to disturbances. The result of the dynamic test is the realised schedule. The disturbances used in this test are historical data from five successive weeks, applying a trace-driven simulation approach as described in Chapter 6. The *schedule measurement*s are multiple objectives and the *state measurement* is handled through the implementation of simulation-optimisation goals (target levels). The multiple objectives used to compare the simulated realised schedule with the real-world realised schedule are somewhat different from the objectives used at the optimisation, due to measurability issues in the real-world.

7.4.1 Experimental settings

Detailed historical data from five successive weeks in 2010 was used as input and output data. One rescheduling Scenario comprised five successive weeks and a total of 23 possible working days, which means 23 periodic rescheduling points (optimisations). Ten replications were carried out for each rescheduling Scenario, in order to obtain reliable results. Output data from the rescheduling Scenario is compared with the real world result from the same period. More details regarding how the experiment, i.e., rescheduling validation, was carried out are included in Chapter 6.

7.4.1.1 Output measures

It was not possible to collect all the data from the real world and, therefore, three factors are compared quantitatively: (1) makespan, (2) missed target levels, and (3) the number of shortage hours. The fourth factor that is measured, but not compared with the real-world result due to measurability issues in the real-world, is the setups at workstations. The data format is not the same, due to a lower level of detail in the data measured. All results are presented on the basis of their average value per week. A more detailed description of the outputs follows:

- *Missed TL* stands for number of target level groups that did not reach their target levels. See Section 6.1.2 for more information.
- *Makespan rel* stands for the makespan difference (hrs) compared to the real-world result. The reason for not using the actual makespan value is that the only interesting value is the remaining production time that could be saved (or lost) when using the simulation-based optimisation. The production time remaining is not simply the remaining time, but the active production time after subtracting possible planned maintenance and other activities from the remaining time.
- *Shortage hours* stand for the total number of hours that jobs were below the different FGI security levels. For example, if three parts are below the security levels for two hours and one part is below the security level for six hours, it is added up to a total of twelve (3x2+1x6) hours. See Section 6.1.2 for more information.
- *Setup time* stands for the total setup time (hrs). The setup time is measured for all workstations in the production line, but cannot be compared to the real-world result, since the production line was not able to collect that data.

7.4.1.2 Scheduling approaches

Unfortunately, each replication takes several days to execute and, furthermore, the manual processing of output data is needed, which takes additional time. Consequently, the number of scheduling approaches tested was reduced to only a few.
When comparing the results of the optimisation methods of the different approaches for the predictive scheduling, it can be concluded that the FHYB method was the best approach, followed by the PS method, and both obtained really good results for the two scenarios. The PDR optimisation method was the worst one, but still the only indirect approach tested. The three methods mentioned were selected to see how they would perform when the actual schedules were realised, i.e., with regard to variability (dynamic measurement). The PS method optimises the predictive schedule using a direct schedule (direct approach using sequences) and the same schedule as input for the reactive scheduling. Hence, each rescheduling cycle will update the schedule used in OIS for the reactive scheduling. The PDR method uses the same rules as those for the predictive scheduling experiment, namely, FCFS, SPT, LPT, EDD, and VSD. In the same manner as the PS method, the PDR approach optimises the predictive schedule using an indirect approach and the same schedule as input for the reactive scheduling.

The FHYB method uses a combination of direct and indirect representation on the production stages decided by the optimisation methods NPS and PDR. The hybrid approach is nearly the same as the one presented as “Free Hybrid” (FHYB) in the previous sub-section, which obtained the best results overall. It is the same method when the optimisation is run to generate the predictive schedule, however, when the schedule is to be realised in the reactive scheduling, there is a difference. The difference is that the hybrid scheduling rule “hybrid setup non-blocking sequence” (HSNBS) is used. The scheduling rule selects jobs according to sequence, as long as the buffer is not full. If the input buffer is full and the next job in sequence is not available, the PDR VSD will be used to select jobs. The rule is mainly used to avoid a deadlock caused by disturbances.

7.4.2 Indirect approach

The results, i.e., the simulated realised schedule, of the indirect approach (PDR) are compared to the real-world findings, as shown in Table 7.11.

Table 7.11 Results of real-world compared to PDR.

	Real-world	PDR
Missed TL	1.0	0.6 (0.2)
Makespan rel	0.0	-12 (2)
Shortage hours	0.0	7480.4 (12971.2)
Setup time	NA	25.2 (2.1)

The real-world production did not reach the minimum target levels of 14.3% average each week, i.e., one of totally seven target levels. The PDR method obtained a better result which revealed that only 8.6% of the target levels were below the minimum level each week. The makespan, when all the products were finalised in one week, of the real-world result was much higher compared to the average result of the PDR method. The PDR method was able to finalise 12 hours before the real-world production, which could result in huge savings, since about eight to fifteen operators work simultaneously on the production line. However, the number of shortage hours is the worst aspect about the results of the PDR method, revealing an average of 7480 hours per week, which is the same as declaring that 10% of the products are each six hours late. The setup time cannot be compared to the real-world result, since it was not collected from the production line.

7.4.3 Direct approach

The results, i.e., the simulated realised schedule, of the direct approach (PS) are compared to the real-world findings, as shown in Table 7.12.

Table 7.12 Results of real-world compared to PS.

	Real-world	PS
Missed TL	1.0	0,4 (0,1)
Makespan rel	0.0	-6,5 (1,1)
Shortage hours	0.0	0 (0)
Setup time	NA	23,8 (1,6)

The result of the PS method is better overall compared to the real-world findings. The simulation obtained a much better result which revealed that only 5.7% (0.4/7) of the target levels were below the minimum level each week. The makespan, when all the products were finalised in one week, of the real-world result was much higher compared to the average result of the direct approach. The PS method was able to finalise 6.5 hours before the real-world production. Neither the real-world nor the PS method was below the security stock levels. The setup time cannot be compared to the real-world result, since it was not collected from the production line.

7.4.4 Hybrid approach

The results, i.e., the simulated realised schedule, of the hybrid approach (FHYB) are compared to the real-world findings. This FHYB method used either sequence or dispatching rules on the different production stages. The results can be seen in Table 7.13.

Table 7.13 Results of real-world compared to FHYB.

	Real-world	FHYB
Missed TL	1.0	0.4 (0)
Makespan rel	0.0	-7.8 (1.2)
Shortage hours	0.0	0 (0)
Setup time	NA	27.5 (1.2)

The result of the FHYB method is better overall compared to the real-world findings. The simulation obtained a much better result which revealed that only 5.7% of the target levels were below the minimum level each week. The FHYB method was able to finalise 7.8 hours before the real-world production. Neither the real-world nor the FHYB method was below the security stock levels. The setup time cannot be compared to the real-world result, since it was not collected from the production line.

7.4.5 Comparison of the optimisation methods

All three optimisation methods have a shorter makespan and missed less target levels compared to the real-world result, see Table 7.14.

Table 7.14 Comparison of different optimisation methods.

	Real-world	PDR	PS	FHYB
Missed TL	1.0	0,6 (0.2)	0.4 (0.1)	0.4 (0)
Makespan rel	0.0	-12 (2)	-6.5 (1.1)	**-7.8 (1.2)**
Shortage hours	0.0	7480.4 (12971.2)	0 (0)	0 (0)
Setup time	NA	25.2 (2.1)	**23.8 (1.6)**	27.5 (1.2)

The PDR method obtained the best result for makespan, but is ruled out due to the shortage which means that products are late. The PDR method seems to be able to utilise the resources in a good way with regard to disturbances, but fails in prioritising the right product variants at the right time. Both the PS and the FHYB methods obtained better results overall compared to the real-world findings and both approaches avoided any shortage, i.e., late parts.

The best result in each sub-objective is compared to the second best value using the Mann-Witney test, see Appendix F, and if it is statistically significant with a 95% confidence level, it is marked in bold. The PS method was better than the hybrid approach in setup time, because a completely strict schedule is better at controlling this aspect, which is similar to the results of the predictive scheduling. The setup time of the PS method is marked in bold, because the difference is significant compared to the FHYB method. The makespan of the FHYB method is marked in bold, because the difference is significant compared to the PS method. When the PS method is compared to the FHYB approach, the latter is the slightly better one due to the importance of makespan. The FHYB method has a makespan that is approximately 1.3 hours better than the PS method and 7.8 hours better than the real-world finding, which could mean huge savings, since about eight to fifteen operators work simultaneously on the production line.

7.5 Comparison of predictive and reactive scheduling results

When comparing the results of the different approaches and their optimisation methods for the predictive scheduling, it can be concluded that the FHYB approach was the best one, followed by the PS method, and both obtained really good results for the two scenarios. The PDR method was the worst one. Both FHYB and PS performed well on makespan, target levels, setups, and shortage, but a deeper analysis revealed that FHYB was better at minimising makespan and PS was better at reducing setup time. In contrast, the PDR method was not able to produce on target and at the same setup times. Additionally, makespan and shortage was high. The three methods mentioned were selected to see how they would perform when the actual schedules were realised, i.e., with regard to variability (dynamic measurement). The variability was based on real-world data gathered during a five week period from the targeted production line, in order to obtain reliable results.

What can be concluded here is that the PS method performs well on all objectives for both the predictive and reactive scheduling. The FHYB method is the best one for both the predictive and reactive results. The PDR method actually performs better in the reactive scheduling compared to the results of the predictive scheduling, since it is able to adapt to those disturbances that arise. However, with the PDR method, parts were late (shortage) in both the predictive and reactive scheduling experiments. FHYB is a combination of PS and PDR, which is also reflected in the results since it has lower makespan than PS, but is worse on setup.

The results of the predictive experiments show clear similarities to the results of the reactive experiments. The PDR rule and FHYB method show that there is a potential in reaching even better solutions when adapting to those changes that arise, e.g., machine failures.

7.6 Discussions

The main focus of this system is to support the scheduling task in real-world production lines, in which the scheduling operation is one part of the system. The proposed scheduling system, including the proposed hybrid genetic algorithm, has been demonstrated effective when implemented in a real-world production system (Chapters 6 and 7). Important functions to include in a scheduling system architecture are summarized in Section 3.4, and the procedure of the scheduling method, based on the system architecture, is described in Sections 4.4.3 and 4.4.4. Furthermore, identified steps and implementing lessons learned, e.g., the importance of integration, and a web-based simulation-based scheduling system, are described in Sections 6.8-6.14. The scheduling system architecture of OSS and the hybrid genetic algorithm is intended to be used for hybrid flow shop (see Section 1.4 and Section 2.3) production lines, such as machining lines similar to the automotive components production line described in Section 6.1.2. The hybrid genetic algorithm and the system architecture are not limited to automotive production lines, because similar problems may be found in various areas, such as in food or semi-conductor production. However, there may be specific constraints and features that need to be considered, e.g., constraints to be included in a discrete event simulation model. For example, job shop production is outside the scope of this study because the hybrid genetic algorithm is designed for hybrid flow shops, even though the system architecture is capable of handling other types of production flows.

7.7 Concluding remarks

The experimental results of the scheduling of a real-world, complex hybrid flow shop scheduling problem are presented in this chapter. The setup reduction experiments prove that the approach of handling groups of batches, proposed in Chapter 5, is very successful in generating better schedules. The novel hybrid genetic representation for real-world, complex hybrid flow shop scheduling problems, described in Chapter 5, was used to test different scheduling approaches. This way of generating schedules, the hybrid approach and more specifically the FHYB optimisation method, has demonstrated to be the best among the methods for both the predictive scheduling and reactive scheduling results. Furthermore, the analysis of the predictive scheduling results shows that FHYB is flexible in different scheduling scenarios and efficiently uses the computing resources where needed, e.g., at the bottleneck operation.

Chapter 8

8 Conclusions and Future Work

This Chapter presents the overall conclusions of the thesis (Section 8.1), a summary of the contributions to knowledge (Section 8.2), and identifies possible areas of future research (Section 8.3).

8.1 Conclusions

The overall thesis of this work can be stated as follows:
"*In order to advance the research and development of manufacturing scheduling systems which can more effectively address the problems found in a complex, real-world manufacturing flow shop, an approach which is a combination of systems engineering and algorithmic design with verifications from empirical experiments using real-world scenarios and data is paramount.*"

The hypothesis stated in Chapter 1 that existing scheduling approaches and algorithms are inadequate to effectively address the problems found in a complex, real-world manufacturing flow shop, has found its support based on the comprehensive literature review carried out in this study. Particularly, the review pinpoints the fact that a scheduling system with real-time and reactive support is badly needed in order to effectively address the uncertainty in real-world problems. Following are the detailed conclusions summarised based on the literature review on production scheduling conducted in this study:

- In order to diminish the gap between theory and practice, more complex scheduling problems need to be considered. Furthermore, in order not to make the same unrealistic assumptions (see Section 2.4.1) that seem to have been a natural part of theoretical scheduling research, real-world problems must be studied. See Section 2.5.
- Discrete-event simulation models have the capability to represent complex real-world systems in detail, as well as cope with several constraints and multiple

objectives. Furthermore, many real-world optimisation problems can only be treated by simulation models. See Section 2.4.5.

- Simulation is not an optimisation tool and GA is appropriately combined with simulation, because quite a few evaluations are needed, in order to search a large area of the solution space (see Section 2.4.5). Combining GA with dispatching rules has been identified by several researchers as a successful approach (see Section 2.4.5), and a hybrid method proposed in this thesis efficiently enhances the simulation-based optimisation in both the predictive and reactive scheduling.
- Solving the sequencing problems is not adequate, since the real-world schedulers are faced with day-to-day challenges of uncertainty (see Section 3.2). In order to handle the scheduling task including uncertainty, an efficient scheduling system is needed (see Section 3.3), not only an efficient algorithm.

From a system engineering perspective, a new, generic, simulation-based, optimisation system framework for solving real-world complex scheduling problems has been proposed (see Section 4.4) and implemented in a real-world industrial environment (See Chapter 6). The main focus of this system is to support the scheduling task in real-world production lines, in which the scheduling operation is one part of the system. The users of such an application are mainly the production scheduler and operators on the shop floor, but also the production engineers to some extent. The following points summarise the support offered to the scheduler:

- The system generates valid schedules (see Section 2.4.5 and Section 6.14) for the short-term scheduling horizon, which reduces the risk that the schedules are adjusted manually.
- The system supports the scheduler in creating schedules based on partial data (see Section 3.2.1), and the system does not require the availability of complete data input, i.e., match-up scheduling and right-shift scheduling may be carried out in reaction events (see Section 4.4.2.5).
- The system supports the scheduler in monitoring schedule execution (see Section 6.5) and performs necessary changes (see Section 4.4.4) when needed,

in order to fulfil scheduling targets. Hence, the scheduler is in direct control to answer questions and give directions.

- The system supports the scheduler to be proactive using what-if scenarios (see Section 3.4) and forecast simulation (see Section 4.4.4), by offering easy access to experimentation using the OPTIMISE platform (see Section 4.3).
- The system supports the scheduler by being accessible for use anytime, anywhere, through Web Services technology (see Section 4.3).

The transparency offered by the system, e.g., through the monitoring capability (see Section 6.5), is also important for the operators at the production line (see Section 3.3.1). The system supports the operators in maintaining control by offering schedule execution monitoring (see Section 6.5.1) through user interface applications, because it is the operators that execute the schedules in a semi-automatic or manual system. The system is reactive, since it deals with the dispatching decisions (see Section 4.4.2) in soft real-time based on the current status of the system. Furthermore, the dispatching client supports both flexibility and fault control (see Section 6.13), because it allows the breaking of soft constraints, but not the breaking of hard constraints. Production engineers need to keep the simulation model up to date and are supported by user interfaces to do so (see Section 6.3.1). Additionally, production engineers may use the scheduling system to test future physical configurations and product changes, i.e. running what-if scenarios.

In terms of contributions in algorithmic design (see Chapter 5 and Section 5.2) for complex production scheduling problems, a novel hybrid approach (see Section 5.2.2) that may be able to handle a wider range of problems and lead to more general systems has been proposed in this thesis. The hybrid approach is believed to be unique compared to other hybrid representations, because it allows using different representation (direct, indirect or hybrid) at different production stages. Other approaches usually apply a hybrid representation to all production stages. An optimisation method using this hybrid approach has been demonstrated to enhance the efficiency of SBO, when evaluated on a real-world setting in both the predictive (see Section 7.3) and reactive scheduling (see Section 7.4). A scheduling method that can handle a wider range of problems is important, because changes due to product mix and production volume modifications are very frequent (see Section 1.1.1) on a real-world production shop floor. For example, when the product mix and volume are changed from week to week, a workstation previously identified as the bottleneck may shift to another production stage and it has been shown that the hybrid approach is particularly efficient in dealing with this problem (see Section 7.3), compared to the other approaches tested in this thesis.

8.2 Contributions to knowledge

In order to cope with the challenges faced by manufacturing companies today, a new, generic, simulation-based, optimisation system framework for solving real-world, complex scheduling problems has been proposed. Furthermore, a hybrid meta-heuristic scheduling algorithm that combines priority dispatching rules and genetic encoding which efficiently enhance the simulation-based optimisation has been proposed. The proposed system has been implemented in a real-world industrial environment and the novel, hybrid genetic representation has been demonstrated effective for a complex scheduling problem using real-world data. The scientific contributions to knowledge from this research include the following:

- The research study has shown that the contextual conditions (see Chapter 6) are indeed important for real-world scheduling problems. Industrial companies are not only facing complex scheduling problems, but also need a new toolset to support their daily scheduling tasks. Scheduling algorithms need to be integrated into a scheduling system and the algorithms may in fact be affected by the contextual conditions of the real-world scheduling deployment, such as deviations in updating a schedule (see Section 6.7) or operators requesting a low variant mix not directly related to the sequence-dependent setup times (see Section 6.11).
- A new, web services-based industrial scheduling system, called OSS (see Chapter 4 and Section 4.4), has been proposed and its applicability demonstrated by a comprehensive industrial implementation and evaluation (see Chapter 6). The research has gone beyond existing studies of simulation-based scheduling applications, as OSS is ready to be used for further research of reactive

scheduling decision support systems and optimisation methods using discrete-event simulation. Particularly, OSS supports further comparisons of new scheduling algorithms, because more algorithms can be added to the algorithm library and evaluated utilising the OPTIMISE platform.

- This study has proposed a generic, novel hybrid genetic representation (see Chapter 5 and Section 5.2.1) for real-world, complex, hybrid flow shop scheduling problems that combines the GA with dispatching rules. Such a hybrid GA representation is believed to be unique, as it has the capability to cope with direct, indirect, and hybrid approaches flexibly. This flexibility is important since different scheduling approaches may be advantageous in different scheduling problems. Furthermore, current scheduling problems may change over time in real-world production systems due to their requirement to be flexible and, therefore, scheduling approaches previously shown to be outstanding may no longer be suitable under the new conditions. The proposed optimisation algorithm can use the different scheduling approaches within the same algorithm and may be kept generic and intact upon changes, due to the fact that most changes will be made to a discrete-event simulation model. Furthermore, it has been shown that the proposed hybrid approach efficiently and flexibly enhances the simulation-based optimisation for both predictive (see Section 7.3) and reactive scheduling (see Section 7.4) based on scenarios using real-world data.

The contributions to knowledge and practice from an industrial perspective could be summarised with the words: validity, generality, simplicity, and flexibility. These terms are further explained and presented as follows:

- Demonstrated that characteristics of real-world scheduling problems, such as multiple objectives and multiple constraints, can all be considered (see Section 6.1) when using simulation-based optimisation within a reasonable time period (see Chapter 6 and Chapter 7).
- The OSS (see Section 4.4), as implemented on top of the generic OPTIMISE platform (see Section 4.3), supports near optimal and real-time scheduling with the help of SBO. OSS utilises the existing core components to support running

the parallel and distributed simulation evaluations via the Internet and enable both SMEs and multinational enterprises, with or without simulation resources, to utilise the system.

- The common users of the system do not need to consider which scheduling approach the system uses, because the functionality remains the same (see Section 6.6).
- The OSS supports experimentation not directly related to the scheduling, i.e., what-if scenarios, which would extend the use of simulation models built for scheduling. This is an important aspect to the scheduling as well, because real-world production systems tend to change over time. Furthermore, the cost of building simulation models for scheduling is reduced in relation to its use. (See Section 4.1 and Section 4.3)

8.3 Future work

There are two possible future research directions, based on the results of this research study:

8.3.1 Benchmark problems

Future research within production scheduling needs to consider more complex scheduling problems, in order to reduce the gap between theory and practice. Consequently, there is a need for a standardised set of complex, hybrid flow shop scheduling problems based on real-world characteristics, so that researchers can compare and evaluate the effectiveness of their algorithms. The hybrid flow shops considered would need to have multiple production stages, parallel machines, multiple scheduling objectives, multiple constraints, and characteristics such as sequence-dependent setup times, i.e., without the unrealistic simplifications usually made. This kind of benchmark problem resembles the use of benchmark functions in optimisation algorithm design and benchmark data sets in data mining research, but will be more specific to real-world complex production scheduling.

8.3.2 Schedule robustness and hybrid methods

This research has proposed a system architecture that is divided into two parts: the predictive schedule created by the simulation-based optimisation and the real-time reactive dispatching. The scheduling robustness methodologies used in this research are mainly the predictive-reactive approach with the periodic rescheduling, but also the complete reactive approach with dispatching rules. A possible extension of this work is to consider scheduling robustness, compare and investigate proactive (robust schedules), completely reactive (heuristics), predictive-reactive, and hybrid approaches for complex hybrid flow shops, preferably on benchmark problems as mentioned above. Furthermore, research could be carried out to gain insight into how hybrid methods could efficiently use predictive schedule information from the simulation-based optimisation together with reactive heuristics that take care of the immediate dispatching.

8.3.3 Integration

Scheduling systems integration with information systems (e.g., ERP systems) and shop floor systems (e.g. machines) has been identified as important, but is outside the immediate scope of this research. However, it is no easy task, because production systems may have old equipment or restrictions to do so. Full integration with the shop floor is desired to overcome the difficulties of information delay, e.g., not being reactive enough in response to machine failures. However, additional information may be needed, such as the estimated time of failure, so that the system can take this into account when generating a new schedule. Consequently, research into how to execute the integration and what information from the systems is important to acquire in order to find different alternatives of system integration is needed.

References

Allahverdi, A., Ng, C. T., Cheng, T. C. E. and Kovalyov, M. Y. (2008). "A survey of scheduling problems with setup times or costs”, European Journal of Operational Research 187(3), pp. 985-1032.

Andersson, M., Persson, A., Grimm, H., and Ng, A. (2007). “A Web-based simulation optimization system for industrial scheduling”, In Proceedings of the 2007 Winter Simulation Conference, Washington D.C., USA, 9-12 December 2007, pp. 1844-1852.

Andersson, M., Ng, A., and Grimm, H. (2008). “Simulation Optimization for industrial scheduling using hybrid genetic representation”, In Proceedings of the 2008 Winter Simulation Conference, Miami, Florida, USA, 7-10 December 2008, pp. 2004-2011.

Andersson, M. (2011). “Industrial scheduling with evolutionary algorithms using a hybrid representation”, MSc thesis (University of Skövde).

April, J., Glover, F., Kelly, J. and Laguna M. (2003). “Practical introduction to simulation optimization”, Proceedings of the 2003 Winter Simulation Conference, New Orleans, Louisiana, pp. 71–78.

April, J., Better, M., Glover, F. and Kelly, J. (2004). “New advances and applications for marrying simulation and optimization”, Proceedings of the 2004 Winter Simulation Conference, Washington, DC, pp. 80–86.

Aytug, H., Khouja, M. and Vergara, F. E. (2003). "Use of genetic algorithms to solve production and operations management problems: A review", International Journal of Production Research 41(17), pp. 3955-4009.

Aytug, H., Lawley, M. A., McKay, K., Mohan, S. and Uzsoy, R. (2005). "Executing production schedules in the face of uncertainties: A review and some future directions", European Journal of Operational Research 161(1), pp. 86-110.

Baker, K. R. and Trietsch D. (1974). “Introduction to Sequencing and Scheduling”, John Wiley and Sons, New York.

Baker, K. R. and Trietsch D. (2009). “Principles of Sequencing and Scheduling”, Wiley Publishing.

Barman S. (1997). “Simple priority rule combinations: an approach to improve both flow time and tardiness”, International Journal of Production Research, Volume 35, Number 10, 1 October 1997 , pp. 2857-2870.

Bean, J., Birge, J., Mittenthal, J., Noon, C. E. (1991). "Matchup Scheduling with Multiple Resources, Release Dates and Disruptions", Operations Research 39(3), pp. 470-483.

Berry, A and Vamplew, P (2004). “PoD Can Mutate: A Simple Dynamic Directed Mutation Approach for Genetic Algorithms”, In: AISAT2004: International Conference on Artificial Intelligence in Science and Technology, 21-25 November 2004, Hobart, Tasmania, Australia.

Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., Weglarz, J. (2001). “Scheduling Computer and Manufacturing Processes”, 2nd ed. Springer, Berlin.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P. and Schulenburg S. (2003). “Hyper-Heuristics: An Emerging Direction in Modern Search Technology”, Handbook of Metaheuristics, pp. 457-474.

Johnson, B., Onwuegbuzie, A. and Turner, L. A. (2007). "Toward a Definition of Mixed Methods Research", Journal of Mixed Methods Research 1(2), pp. 112-133.

Byrne, J., Heavey, C. and Byrne, P.J. (2010). "A review of Web-based simulation and supporting tools", Simulation Modelling Practice and Theory 18(3), pp. 253-276.

Bäck, T., Fogel, D. B., Whitley, D., Angeline, P.J. (1997a). “Mutation”, In Bäck, T., Fogel, D. B. and Michalewicz, Z. (Ed.) Handbook of Evolutionary Computation, Oxford University Press, Oxford.

Bäck, T. (1997b). “Binary strings”, In Bäck, T., Fogel, D. B. and Michalewicz, Z. (Ed.) Handbook of Evolutionary Computation, Oxford University Press, Oxford.

Church, L.K., Uzsoy R. (1992). “Analysis of periodic and event-driven rescheduling policies in dynamic shops”, International Journal of Computer Integrated Manufacturing, Vol.5, pp.153-163.

Conway, R. W., Maxwell, W. L. and Miller L. W. (1967). “Theory of scheduling”, Addison-Wesley Pub. Co.

CVS (2012). “DaimlerChrysler”, Available from: http://e-opt.informs.org/ resources/casestudy.cfm?id=9, Assessed: 2012-12-03.

Dangelmaier, W., Mahajan, K. R., Aufenanger., M. and Seeger, T. (2007). “Simulation Assisted Match-up Rescheduling of Flexible Production Systems Subject to Execution Exceptions”, In Proceedings of the 2007 Winter Simulation Conference, pp. 1805-1813.

Dangelmaier, W., Mahajan, K., R., Seeger, T., Klöpper, B. and Aufenanger., M. (2006). “Simulation Assisted Optimization and Real-Time Control Aspects of Flexible Production Systems Subject to Disturbances”, In Proceedings of the 2006 Winter Simulation Conference, pp. 1785-1795.

Daniels, R. L. and P. Kouvelis (1995). "Robust scheduling to hedge against processing time uncertainty in single-stage production", Manage. Sci. 41(2), pp. 363-376.

Davis, L. (Ed.). (1991). “Handbook of genetic algorithms”, New York: Van Nostrand Reinhold.

Davis, L. (1985). “Applying adaptive algorithms to epistatic domains”, in: Proceedings of the International Joint Conference on Artificial Intelligence, pp.162–164.

Davis, W. J. (1998). “On-line simulation: the need and the evolving research requirements”, In Simulation Handbook (Ed, Banks, J.), John Wiley and Sons, Inc., pp. 465-516.

De Snoo, C., Van Wezel, W. and Jorna, R. J. (2011). "An empirical investigation of scheduling performance criteria." Journal of Operations Management 29(3), pp. 181-193.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2000). “A fast and elitist multi-objective genetic algorithm NSGA-II”, *KanGAL Report 2000001*, Indian Institute of Technology Kanpur, India.

Deep, K. and M. Thakur (2007). "A new mutation operator for real coded genetic algorithms", Applied Mathematics and Computation 193(1), pp. 211-230.

Delfoi (2012). “Delfoi Planner”, Available from: http://www.delfoi.com/, Assessed: 2012-12-03.

Denzin, N. K. (2009). “The Research Act: A Theoretical Introduction to Sociological Methods”, Aldine Transaction.

Engin, O., Ceran, G. and Yimlas, M. K. (2011). "An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems", Applied Soft Computing 11(3), pp. 3056-3065.

Falkenauer E., Bouffouix S. (1991). “A genetic algorithm for job shop”, in: Proceedings of the IEEE International Conference on Robotic and Automation, pp. 824-829.

Ferris, T. L. J. (2009). “On the methods of research for systems engineering”, In proceeding of the 7th Annual Conference on Systems Engineering Research 2009 (CSER 2009).

Fishwick, P. A. (1996). Web-based simulation: some personal observations. Proceedings of the 28th conference on Winter simulation. Coronado, California, USA, IEEE Computer Society**,** pp. 772-779.

Framinan, J. M. and Ruiz R. (2010). "Architecture of manufacturing scheduling systems: Literature review and an integrated proposal", European Journal of Operational Research 205(2), pp. 237-246.

Frantzén, M., Ng, A.H.C., Moore, P. (2011). “A simulation-based scheduling system for real-time optimization and decision making support”, Robotics and Computer-Integrated Manufacturing, Volume 27, Issue 4, August 2011, pp. 696-705.

Frantzén, M., Ng, A.H.C., Moore, P. (2010). “A Scheduling system for real-time decision making support using simulation-based optimization”, In Proceedings of the 20th International Conference on Flexible Automation and Intelligent Manufacturing, July 12-14, 2010, Oakland, USA. pp. 1012-1019.

Garey, M. R. and Johnson D. S. (1979). “Computers and Intractability: A Guide to the Theory of NP-Completeness”, W. H. Freeman \\&amp; Co.

Gary, K., Kempf, K., Smith, S., Uzsoy, R., (1992). "Assessing the Quality of Production Schedules", Proceedings of the Intelligent Scheduling Systems Symposium, San Francisco, CA, November 1992.

Graham, R. L., Lawler, E. L. Lenstra, J.K. and Rinnooy Kan, A.H.G. (1979). “Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey”, Annals of Discrete Mathematics. E. L. J. P.L. Hammer and B. H. Korte, Elsevier. Volume 5, pp. 287-326.

Graves S.C. (1981). “A review of production scheduling”, Operations Research 29 (4), pp. 646-675.

Groover, M. P. (2000). “Automation, Production Systems, and Computer-Integrated Manufacturing”, 2nd Ed, Prentice Hall PTR.

Gupta J. N. D. (1988). “Two-Stage, Hybrid Flowshop Scheduling Problem”, The Journal of the Operational Research Society, Vol. 39, No. 4 (Apr., 1988), pp. 359-364.

Gupta, J. N. D., Hariri A. M. A. and Potts C.N. (1997). "Scheduling a two-stage hybrid flow shop with parallel machines at the first stage", Annals of Operations Research 69(0), pp. 171-191.

Gupta J. N. D. and Stafford Jr. E. F. (2006). “Flowshop scheduling research after five decades”, European Journal of Operational Research, Vol.169, pp. 699-711.

Gusikhin, O., Rychtyckyj, N. and Filev, D. (2007). “Intelligent systems in the automotive industry: applications and trends”, Knowledge and information systems, pp: 147-168.

Heilala, J., Montonen, J., Kuula, T., Usenius, T., Maantila, M. and Sillanpää, J. (2011) “User-Centric Development Of Simulation Based Manufacturing Operation Planning

And Scheduling System”, 2011 IEEE International Symposium on Assembly and Manufacturing (ISAM), pp. 1-6.

Herrmann, J. W. (2006) “Rescheduling strategies, policies, and methods”, In Herrmann, J. W. (Ed.) Handbook of scheduling. Springer Science+Business Media, Inc, New York.

Higgins, P. G. (1996). “Using Graphics to Display Messages in an Intelligent Decision Support System”, In Proceedings of the Second Melbourne Workshop on Intelligent Decision Support Systems, pp. 32-38.

Holland, J. H. (1962). “Outline for a logical theory of adaptive systems”, Journal of ACM 3, pp. 297–314

Holland, J. H. (1975). “Adaptation in Natural and Artificial Systems”, Ann Arbor, MI: University of Michigan Press.

Holthaus, O. (1997). "Design of efficient job shop scheduling rules", in Proceedings of the 21st International Conference on Computers and Industrial Engineering, Computers & Industrial Engineering 33(3-4), pp. 249-252.

Horn, S., Weigert, G., Schönig, P. and Thamm, G. (2006). ”Application of Simulation-Based Scheduling in a Semiconductor Backend Facility”, Proceedings of ESTC 2006 - 1st Electronics System integration Technology Conference, pp. 1122-1126.

ILOG (2012). “ILOG solver”, Available from:http://www01.ibm.com/software/ websphere/products/optimization/, Assessed: 2012-12-03.

Jabar, M. A., Sidi, F., Selamat, M. H., Azim Abd Ghani, A. and Ibrahim, H. (2009). “An Investigation into Methods and Concepts of Qualitative Research in Information System Research”, Journal of Computer and Information Science, Vol. 2, No. 4, pp. 47-54.

Jahangirian, M., Eldabi, T., Naseer, A., Stergioulas, L. K., Young, T. (2010). "Simulation in manufacturing and business: A review", European Journal of Operational Research 203(1), pp. 1-13.

Jayamohan, M. S. and Rajendran C. (2004). "Development and analysis of cost-based dispatching rules for job shop scheduling", European Journal of Operational Research 157(2), pp. 307-321.

Jin, H., Ibrahim, S., Bell, Tim, Qi, L., Cao, H., Wu, S. and Shi, X. (2010). “Tools and Technologies for Building Clouds”, Cloud Computing, Antonopoulos, N. and Gillam, L. (Eds.), Springer London, pp. 3-20.

Kellegöz, T., Toklu, B., Wilson, J. (2008). "Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem", Applied Mathematics and Computation 199(2), pp. 590-598.

Kempf, K., R. Uzsoy, Smith S. and Gary K., (2000). "Evaluation and comparison of production schedules", Computers in Industry 42(2), pp. 203-220.

Khalil, R. A., Stockton, D., Kang, P.S. and Mukhongo, L. (2012). “A Multi-Objective Optimization Approach Using Genetic Algorithms for Quick Response to Effects of Variability in Flow Manufacturing”, International Journal of Advanced Computer Science and Applications, 3 (9), pp. 12-17.

Kianfar, K., Fatemi Ghomi, S. M. T. and Oroojlooy Jadid, A. (2012). "Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA", Engineering Applications of Artificial Intelligence 25(3), pp. 494-506.

Kim, I., Watada J., Shigaki I. (2007). "A comparison of dispatching rules and genetic algorithms for job shop schedules of standard hydraulic cylinders", Soft Comput. 12(2), pp. 121-128.

Kiran A. S. (1998). “Simulation and scheduling”, In “Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice”, ed. J. Banks, John Wiley and Sons, Inc., New York, pp. 677-717.

Koh, K.-H., R. de Souza, Ho, N-C. (1996). "Database driven simulation/simulation-based scheduling of a job-shop", Simulation Practice and Theory 4(1), pp. 31-45.

Korejo, I., Yang, S., Li, C. (2010). ”A Directed Mutation Operator for Real Coded Genetic Algorithms”, Applications of Evolutionary Computation, C. Chio, S. Cagnoni, C. Cottaet al, Springer Berlin Heidelberg 6024, pp. 491-500.

Kumar, S. and Nottestad D. A. (2006). "Integrated simulation application design for short-term production scheduling", IIE Transactions 38(9), pp. 737 - 748.

Laguna M., Marti R. (2003). “Scatter Search: Methodology and Implementation in C”, Operations Research/Computer Science Interfaces Series, Vol. 24., Kluwer Academic Publishers.

Land, A. H. and Doig A. G. (1960). "An Automatic Method of Solving Discrete Programming Problems", Econometrica 28(3), pp. 497-520.

Law, A. M. and McComas, M. G. (2000). “Simulation-based optimization”, Proceedings of the 2000 Winter Simulation Conference, pp. 46-49.

Law, A. M. and Kelton, W. D. (2000). “Simulation modeling and analysis.”, 3rd Ed., McGraw-Hill Book Co – Singapore.

Lawrence, S. R. and Sewell E. C. (1997). "Heuristic, optimal, static, and dynamic schedules when processing times are uncertain", Journal of Operations Management 15(1), pp. 71-82.

Leon, J., Wu, D., Storer, R. H. (1994). "Robustness Measures and Robust Scheduling for Job Shops", IIE Transactions 26(5), pp. 32-43.

Li, H. and Wang, D. (2008). “Parallel simulation-based optimization on block planning of container terminals”, In Proceedings of Chinese Control and Decision Conference (CCDC), pp. 3224-3228.

Li, L., Yumin, Q. F. M. and Kai, Y. (2012). ”Simulation-based modular scheduling system of semiconductor manufacturing”, In da Rosa Righi, R. (Ed.) Production Scheduling, InTech, Rijeka, Croatia, Available from: http://www.intechopen.com/books/production-scheduling/simulation-based-modular-scheduling-system-of-semiconductor-manufacturing, Assessed: 2012-10-01.

Liaw, C.-F. (2000). "A hybrid genetic algorithm for the open shop scheduling problem", European Journal of Operational Research 124(1), pp. 28-42.

Lowry, R. (2012). “Concepts and applications of inferential statistics”, Vassar College, Available from: http://www.vassarstats.net/textbook/index.html, Assessed 2012-12-14.

Luo, Y.-C., C.-H. Chen, Enver, Y. (2000). “Distributed web-based simulation optimization”, Proceedings of the 32nd conference on Winter simulation, Orlando, Florida, Society for Computer Simulation International, pp. 1785-1793.

Lövås, G., G. (2006). ”Statistik – metoder och tillämpningar”, 1st Ed., Liber AB, Malmö.

Mann, H. B. and Whitney, D. R. (1947). “On a test of whether one of two random variables is stochastically larger than the other”, Annals of Mathematical Statistics, pp. 50-60.

Matsuura, H., Tsubone, H., Kanezashi, M. (1993). "Sequencing, dispatching, and switching in a dynamic manufacturing environment", International Journal of Production Research 31(7), pp. 1671-1688.

McKay, K. N., Safayeni, F. R., Buzacott, J. A. (1995). "‘Common sense’ realities of planning and scheduling in printed circuit board production", International Journal of Production Research 33(6), pp. 1587-1603.

McKay, K. N. and Wiers V. C. S. (1999). "Unifying the theory and practice of production scheduling", Journal of Manufacturing Systems 18(4), pp. 241-255.

McKay, K., Pinedo, M., Webster, S. (2002). "Practice-focused research issues for scheduling systems", Production and Operations Management 11(2), pp. 249-258.

McKay, K. N. and Buzacott J. A. (2000). "The application of computerized production control systems in job shop environments", Computers in Industry 42(2-3), pp. 79-97.

McKay, K. N. and Black G. W. (2007). "The evolution of a production planning system: A 10-year case study", Computers in Industry 58(8–9), pp. 756-771.

McKay, K. N. and Wiers, V. C. S. (2003). "Integrated decision support for planning, scheduling, and dispatching tasks in a focused factory", Computers in Industry 50(1), pp. 5-14.

Morton, T. E. and Pentico, D. W. (1993). “Heuristic scheduling systems: with applications to production systems and project management”, John Wiley, NY.

Myers, M. D. (1997). “Qualitative research in information systems”, MIS Quarterly (21:2), pp. 241-242.

Nearchou, A. C. (2004). "The effect of various operators on the genetic search for large scheduling problems", International Journal of Production Economics 88(2), pp. 191-203.

Ng, A., Grimm, H., Lezama, T., Persson, A., Andersson, M. and Jägstam, M. (2008). “OPTIMISE: An Internet-Based Platform for Metamodel-Assisted Simulation Optimization”, Recent Advances in Communication Systems and Electrical Engineering, X. Huang, Y-S. Chen and S-L. Ao (eds), Springer, pp. 281-296.

Ng, A., Grimm, H., Lezama, T., Persson, A., Andersson, M. and Jägstam, M. (2007). “Web Services for Metamodel-Assisted Parallel Simulation Optimization”, In Proceedings of the IAENG International Conference on Internet Computing and Web Services (ICICWS'07), Hong Kong, 21-23 March 2007, pp. 879-885.

Ochoa, G., Vazquez-Rodriguez, J. A., Petrovic, S., Burke, E. (2009). “Dispatching rules for production scheduling: A hyper-heuristic landscape analysis”, Evolutionary Computation, in Proceedings of the Eleventh conference on Congress on Evolutionary Computation, pp. 1873-1880.

Panwalkar, S. S. and Iskander W. (1977). "A Survey of Scheduling Rules", Operations Research 25(1), pp. 45-61.

Pierreval, H. and Mebarki, N. (1997). "Dynamic scheduling selection of dispatching rules for manufacturing system", International Journal of Production Research 35(6), pp. 1575-1591.

Pinedo, M. L. (2005) “Planning And Scheduling In Manufacturing And Services”, Springer Science+Business Media.

Pinedo M. (2008) “Scheduling: Theory, Algorithms, and Systems”, Prentice Hall, 3rd Ed., New York, NY.

Pinedo, M., Samroengraja, R., and Yen, B. P-C. (1994). “Design Issues with Regard to Scheduling Systems in Manufacturing”, in Leondes T. L. (Ed.) Control and Dynamic Systems, Academic Press, pp. 203-238.

Reswick, J. B. (1994). “What Constitutes Valid Research? Qualitative vs. Quantitative Research”, Journal of Rehabilitation Research & Development, pp. vii-ix.

Ribas I., Leisten R. and Framiñan J. M. (2010). “Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective”, Journal of Computers & Operations Research, Vol.37, pp. 1439-1454.

Robson, C. (2011). “Real world research: A resource for users of social research methods in applied settings”, 3rd Ed., John Wiley & Sons.

Rogers, A. and Prugel-Bennet, A. (1999). “Modelling the dynamics of a steady state genetic algorithm”, In Banzhaf, W. and Reeves, C. (Eds.) Foundations of genetic algorithms, Morgan Kaufmann Publishers, Inc., San Fransisco, CA. pp. 57-68.

Roser, C., Nakano, M. and Tanaka, M. (2002). “Shifting bottleneck detection”, In proceeding of 2002 Winter Simulation Conference, pp.1079-1086.

Roundy, R. O., Maxwell, W. L., Herer, Y. T., Tayur, S. R., Getzler, A. W. (1991). "A Price-Directed Approach to Real-Time Scheduling of Production Operations", IIE Transactions 23(2), pp. 149-160.

Ruiz, R. and Vázquez-Rodríguez, J. A. (2010). "The hybrid flow shop scheduling problem", European Journal of Operational Research 205(1), pp. 1-18.

Sanchez, S. M.(2000) “Robust design: Seeking the best of all possible worlds”, In Proceedings of the 2000 Winter Simulation Conference, pp.69-76.

Sankar, S. S., Ponnanbalam, S. G. and Rajendran C. (2003). “A multiobjective genetic algorithm for scheduling a flexible manufacturing system”, International Journal in Advanced Manufacturing Technologies, Vol.22, pp. 229-236.

Sastry, K., Goldberg, D. and Kendall G. (2005) “Genetic Algorithms”, In Burke, Edmund K.; Kendall, Graham (Eds.) Introductory Tutorials in Optimization and Decision Support Techniques, Springer Science+Business Media, Inc, New York.

Shapiro, J. F. (1979). “A Survey of Lagrangean Techniques for Discrete Optimization”. Annals of Discrete Mathematics. E. L. J. P.L. Hammer and B. H. Korte, Elsevier. Volume 5, pp. 113-138.

Sivakumar, A. I. and Gupta, A. K. (2006). ”Online Multiobjective Pareto Optimal Dynamic Scheduling of Semiconductor Back-End Using Conjunctive Simulated Scheduling”, IEEE Transactions on Electronics Packaging Manufacturing, Vol.29, No.2, pp. 99-109.

Smith, J. S., Wysk, R. A., Sturrock, D. T., Ramaswamy, S. E., Smith, G. D. and Joshi, S. B. (1994). “Discrete event simulation for shop floor control”, Proceedings of the 1994 Winter Simulation Conference, pp. 962-969.

Son, Y. J., Wysk, R. A. and Jones A. T. (2003). "Simulation-based shop floor control: Formal model, model generation and control interface", IIE Transactions (Institute of Industrial Engineers) 35(1), pp. 29-48.

Stockton, D. J., Khalil, R.A. and Mukhongo, L. M. (2012). “Autonomous Planning using the Basic Principles of Gene Transcription Regulatory Control”, American Journal of Operational Research, 2(5), pp. 66-80.

Stockton, D. J., Quinn, L. and Khalil, R.A. (2004a). “Use of genetic algorithms in operations management Part 1: applications”, Proceedings of the Institution of

Mechanical Engineers -- Part B -- Engineering Manufacture, Vol. 218 Issue 3, pp. 315-327.

Stockton, D. J., Quinn, L. and Khalil, R.A. (2004b). “Use of genetic algorithms in operations management Part 2: results”, Proceedings of the Institution of Mechanical Engineers -- Part B – Engineering Manufacture, Vol. 218 Issue 3, pp. 329-343.

Stoop, P. P. M. and Weirs, V. C. S. (1996). “The complexity of scheduling in practice”, International Journal of Operations and Production management, 16(10), pp. 37-53.

Suh, W. (2005). “Web application development methodologies”, In Travers, J. (Ed.) Web Engineering: Principles and techniques, Idea Group Publishing, London, pp. 76-96.

Suwa, H., Fujiwara, T. (2007). “A new hybrid rescheduling policy based on cumulative delay”, In proceeding of 19th International Conference on Production Research.

Syswerda, G. (1991). “Schedule optimization using genetic algorithms”, In L. Davis (Ed.) Handbook of genetic algorithms, New York: Van Nostrand Reinhold, pp. 332–349.

Talbi, E.-G. (2009). “Metaheuristics : from design to implementation”, John Wiley & Sons.

Tanev, I. T., Uozumi, T., Morotome, Y. (2004). "Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provider approach", Applied Soft Computing 5(1), pp. 87-100.

Tay, J. C. and Ho, N. B. (2008). "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems", Computers &amp; Industrial Engineering 54(3), pp. 453-473.

Tolio, T., Ceglarek, D., ElMaraghy, H.A., Fischer, A., Hug, S.J., Laperrière, L., Newman, S.T., Váncza, J. (2010). "SPECIES—Co-evolution of products, processes and production systems." CIRP Annals - Manufacturing Technology 59(2), pp. 672-693.

Veith, T. L., Kobza, J. E. and Koelling, C. P. (1999). "Netsim: Java™-based simulation for the World Wide Web", Computers & Operations Research 26(6), pp. 607-621.

Vieira, G.E., Herrmann, J.W., and Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies, and methods", Journal of Scheduling, 6(l), pp. 39-62.

Vignier A., Billaut J.-C. and Proust C. (1999). “Les problèmes d'ordonnancement de type flow-shop hybride : état de l'art”, RAIRO - Operations Research 33, pp. 117-183.

Wan, Y. W. (1995). "Which is better, off-line or real-time scheduling?", International Journal of Production Research 33(7), pp. 2053-2059.

Whitley D. (1997). “Permutation”, In Bäck, T., Fogel, D. B. and Michalewicz, Z. (Ed.) Handbook of Evolutionary Computation, Oxford University Press, Oxford.

Wiedemann, T. (2001). “Simulation application service providing (SIM-ASP)”, Proceedings of the 33nd conference on Winter simulation, Arlington, Virginia, IEEE Computer Society, pp. 623-628.

Wiers,, V. C. S. (1997). "Human-Computer Interaction in Production Scheduling: Analysis and Design of Decision Support Systems for Production Scheduling Tasks", PhD thesis (Eindhoven University of Technology).

Voß, S. and Woodruff, D. L. (2000). “Optimization Software Class Libraries”, Secaucus, NJ, USA, Kluwer Academic Publishers.

Yang, J. and Chang, T.S. (1998). “Multiobjective Scheduling for IC Sort and Test with a Simulation Testbed”, IEEE Transactions on Semiconductor Manufacturing, Vol 11 (2), pp. 304-315.

Yin, R. K. (2003). “Case study research: Design and methods”, 3rd Ed., Thousand Oaks, CA: Sage.

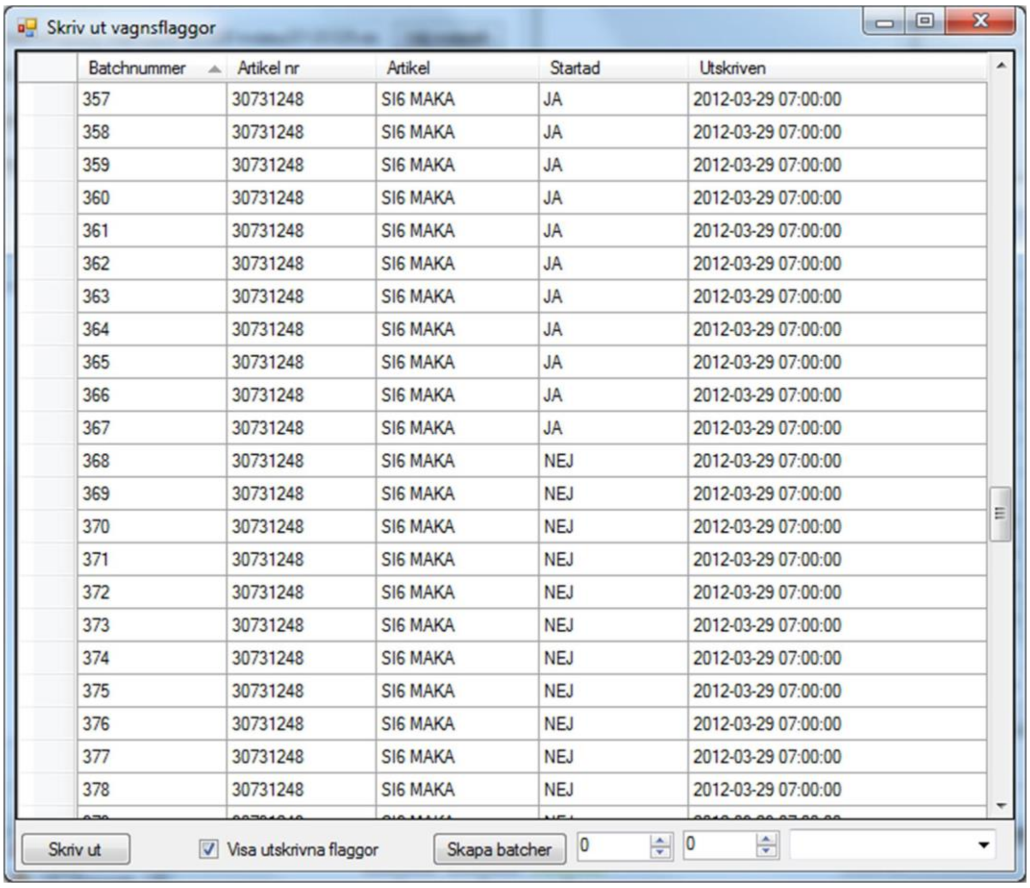
Yin, R.K. (2007). ”Fallstudier: design och genomförande” (Original title: Case study research), Malmö, Sweden, Liber AB.

Yoo, T., Cho, H. and Yücesan, E. (2009). "Web Services-Based Parallel Replicated Discrete Event Simulation for Large-Scale Simulation Optimization." Simulation 85(7), pp. 461-475.

Zhang, W., Yin C., Liu J. and Linn R.J. (2005). "Multi-job lot streaming to minimize the mean completion time in m-1 hybrid flowshops", International Journal of Production Economics 96(2), pp. 189-200.

Appendix A: Carrier Flag Client

The carrier flag client holds all of the current job information (already produced, currently produced and not yet started) by communicating with OIS, see Figure A. 1.


Skriv ut vagnsflaggor

Batchnummer	Artikel nr	Artikel	Startad	Utskriven
357	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
358	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
359	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
360	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
361	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
362	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
363	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
364	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
365	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
366	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
367	30731248	SI6 MAKA	JA	2012-03-29 07:00:00
368	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00
369	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00
370	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00
371	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00
372	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00
373	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00
374	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00
375	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00
376	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00
377	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00
378	30731248	SI6 MAKA	NEJ	2012-03-29 07:00:00

Figure A. 1 Carrier flag program.

The carrier flag is important because it lets the user print those carrier flags that hold the job information. The carrier flags have some necessary production information and a barcode of the job ID. Once the carrier flags have been printed, the batches will be tagged with “available for production” at the first production stage. In order to control that duplicates of the same flag have not been printed, the program will set an attribute to each flag printed.

Appendix B: Dispatching Client User-interface

Start screen

The start screen, displayed in Figure B. 1, shows the different options used by the operators in the production line.

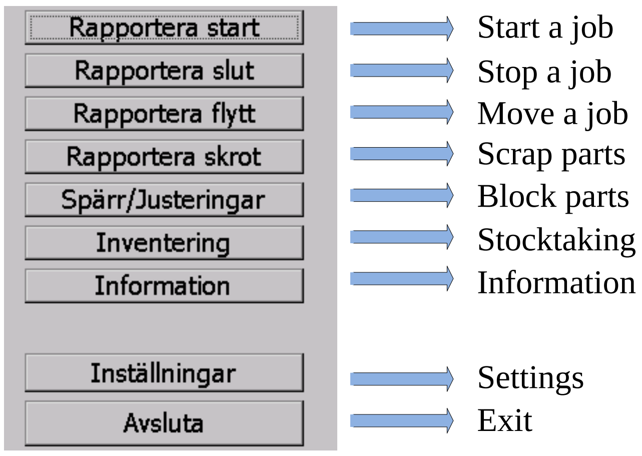
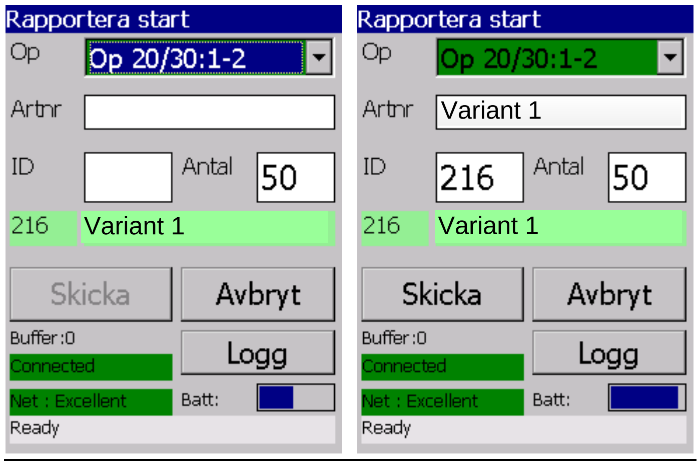


Figure B. 1 Start screen.

The most frequently used options are the first two buttons, i.e., to start or stop a job in a machine. It is possible move a job or parts of a job to other destinations, scrap parts, block parts, do stocktaking or fetch information about a job. The last two buttons are used for system settings and to exit the program. Report start and stop of a job can almost entirely be handled only using the barcode scanner. The other options, which are not used as frequently, may use barcode scanning for the jobs' id numbers.

Start or stop a job

When the operator needs to start jobs at the machines, no consideration needs to be taken whether a direct-, indirect- or hybrid representation is used at the machine. The operator gets the same behaviour no matter what. The screens of starting a job can be seen in Figure B. 2.


Rapportera start
Op: Op 20/30:1-2
Artnr:
ID:
Antal: 50
216 Variant 1
Buffer:0
Connected
Net : Excellent
Batt:
Ready

Rapportera start
Op: Op 20/30:1-2
Artnr: Variant 1
ID: 216
Antal: 50
216 Variant 1
Buffer:0
Connected
Net : Excellent
Batt:
Ready

Figure B. 2 Start a job screen.

When the operator selects a machine, e.g., by barcode scanning, the expert suggestion is displayed for the operator in the green field on the left side in the figure. The colour green means that the job is available at the production stage and the colour red means that the job is not available. The first number “216” stands for the job ID number and the second text string “Variant 1” is the name of the product variant. The selection of the job to start, e.g., by scanning a barcode of the carrier, is shown on the right side in Figure B. 2. The screens used for stopping jobs are similar to the screens for starting jobs.

Confirm screens

Confirm screens, e.g., Figure B. 3, show an example of a confirmation screen, in which the operator may see the settings made and confirm the decisions.

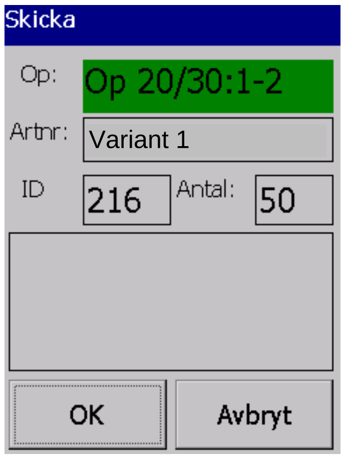

Skicka
Op: Op 20/30:1-2
Artnr: Variant 1
ID 216
Antal: 50

Figure B. 3 Confirm screen.

Move a job

The operations to move a job depend on the area from which the move is made, because different areas contain different levels of details for the jobs, see Figure B. 4.

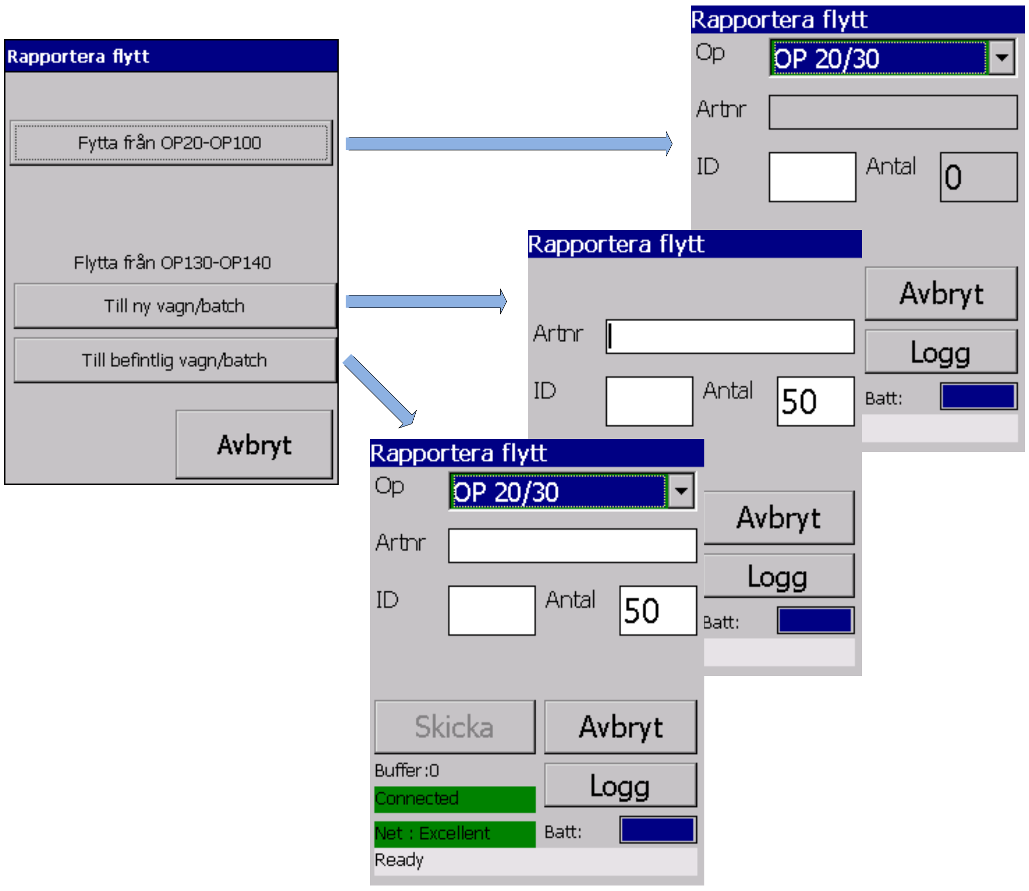


Figure B. 4 Move a job.

The first option relates to the information that can be found when moving a job from the first part of the line, i.e., production stages 1-7, as described in Chapter 6. The only information needed is the job ID and to which destination the parts will be moved. The other two options relate to parts at the end of the line and depend on whether the product variants are going to be moved to an existing carrier or onto a new carrier. The different options require a different level of information detail, because some stages do not have information about the job ID-number (partial information). This is also the case in the following screenshots for other menus.

Scrap parts

The screens of scrapping parts can be seen in Figure B. 5.

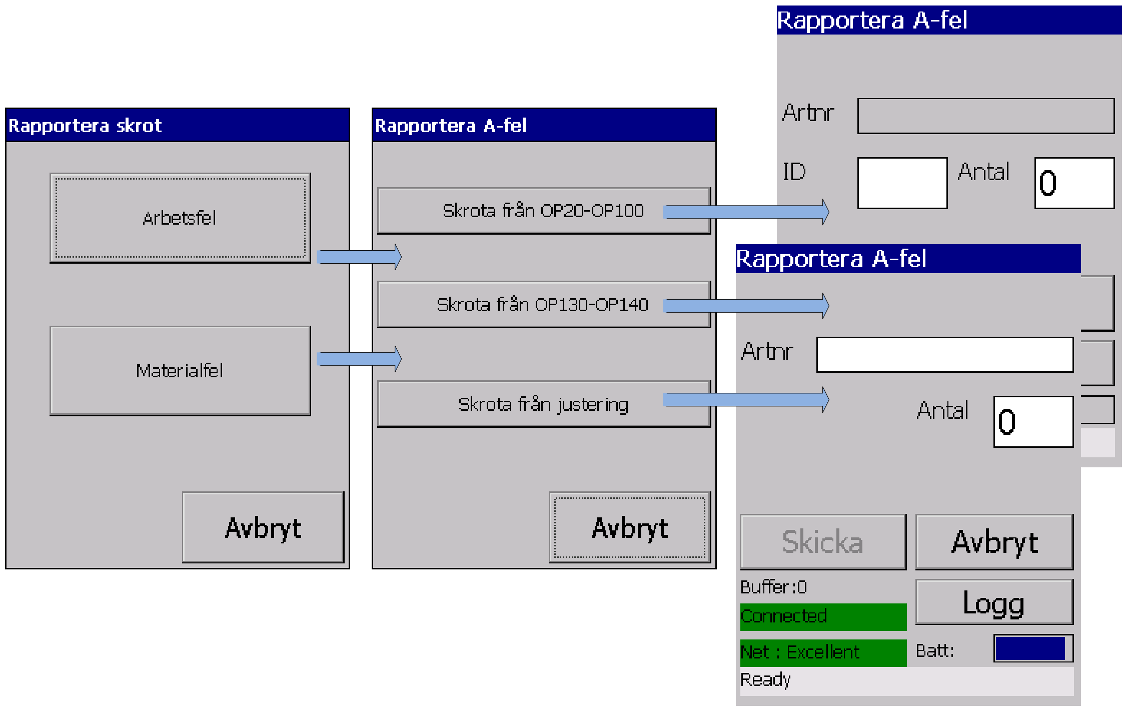
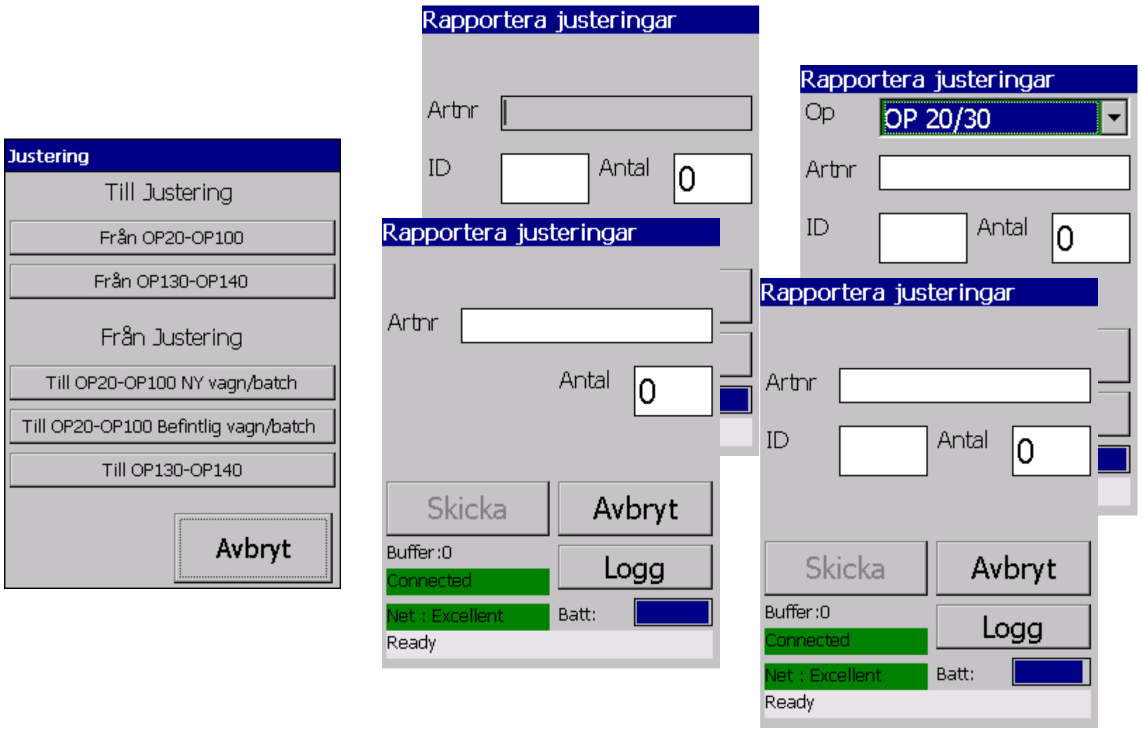


Figure B. 5 Confirm screen.

The first choice made by the operator is to decide whether the scrapped parts depend on processing errors (“Arbetsfel”) or material quality problems (“Materialfel”). The decision priamarily depends on who is going to pay for the scrapped parts, the company or the supplier. The next step is to select the area in the production line from which the parts are located, and depending on this choice, screens will be displayed with a different level of detail.

Block parts

When parts need to be blocked (put on hold) for quality control or adjustments, two options are possible: (1) parts from the first part of the line or (2) parts from the last part of the line, as displayed top left in Figure B. 6.



Op OP 20/30

Antal 0

Buffer :0

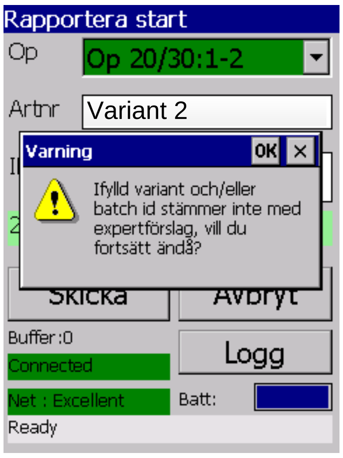
Net : Excellent

Figure B. 6 Block parts.

When the quality of parts has been approved, there are three options that the operator may choose: (1) if the parts are to be moved to a new carrier at the beginning of the line, (2) if the parts are to be moved to an existing carrier at the beginning of the line or (3) if the parts are to be moved to the end of the line.

Fault control and robustness

When the soft constraints are broken by the operators, e.g., trying to override an expert suggestion as displayed in Figure B. 7, a warning will appear at the screen together with a loud beep.



Ifylld variant och/eller batch id stämmer inte med expertförslag, vill du fortsätt ändå?

Figure B. 7 Warning when trying to break soft constraints.

When the hard constraints are broken, e.g., trying to start a job with the wrong tooling equipment, a similar warning will appear at the screen that will inform the user that the chosen operation is not valid. There is also some robustness implemented in the PDA-program and information about the PDA-status, see Figure B. 8.

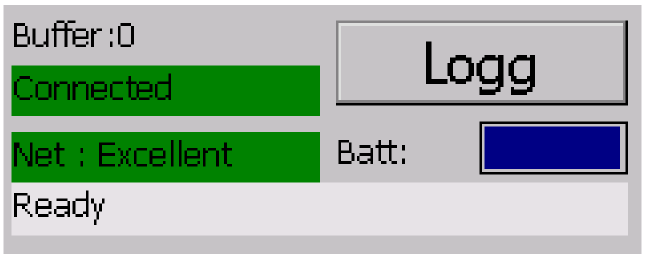


Figure B. 8 Robustness and PDA-status.

The “buffer” displays the number of in queue messages to be sent to OIS, which is usually zero, unless the wireless connection displays something other than “Connected”. There is also a log-file of the last messages sent to OIS. Furthermore, if the PDA loses its power or if it for some reason becomes deadlocked and the PDA needs to be restarted, all information is saved onto the hard-drive which will automatically be executed upon restart.

Stocktaking

Maintaining control of the WIP in the production line is important and an efficient way to do a quick stocktaking has been implemented, see Figure B. 9.

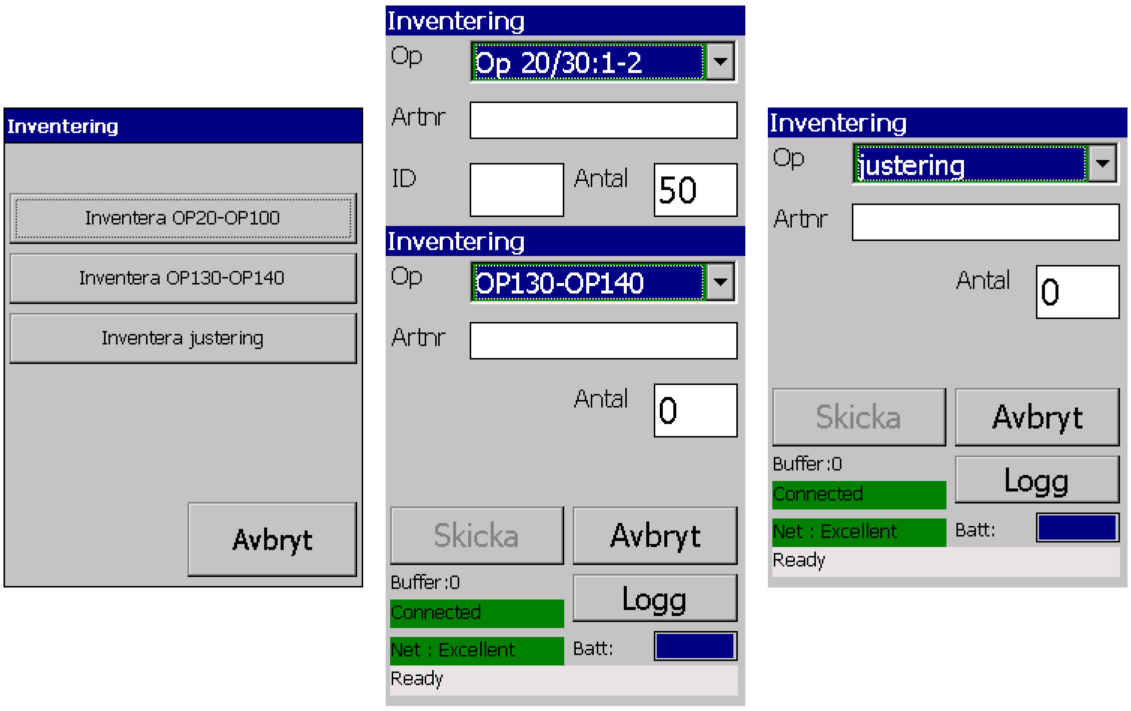
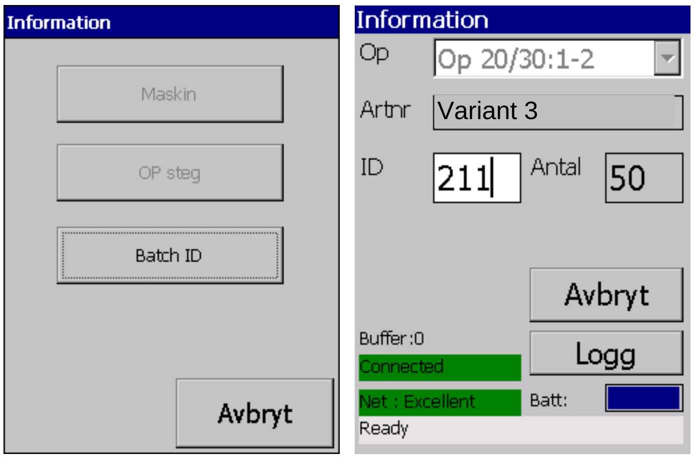


Figure B. 9 Stocktaking menus.

Similar to the other menues, the level of detail is different depending on the area in which the stocktaking is being carried out. There are currently three areas: (1) beginning of the line, (2) end of the line and (3) adjustments area (parts on hold). The stocktaking was carried out in a manual manner before, and the barcode scanner makes this process faster and more detailed.

Information

The information function allows the user to control the status of the production line, see Figure B. 10.



Op: Op 20/30:1-2
Artnr: Variant 3
ID: 211
Antal: 50
Buffer :0
Connected
Net : Excellent
Batt:
Ready

Figure B. 10 Information menu.

On the left side of Figure C.10 there are three options available: (1) machine information, (2) production stage information, and (3) job ID information. The first two options are supported by the line status program and are therefore not supported directly in the PDA at the moment. The last option, job ID information, is supported in the PDA because it gives the user the possibility to check information about a job. When a job ID is scanned by the barcode reader, information about the position (machine or production stage), variant name and number of parts on the carrier (job ID) is displayed for the user, as shown on the right in Figure B. 10. This helps the shift leader to maintain control over the WIP, in case a carrier has been misplaced or if there is a desire just to control that the operators have followed the work procedure properly.

Appendix C: Predictive Scheduling Performance Results Scenario 1

The results of scenario 1 of all replications of the predictive scheduling are presented here.

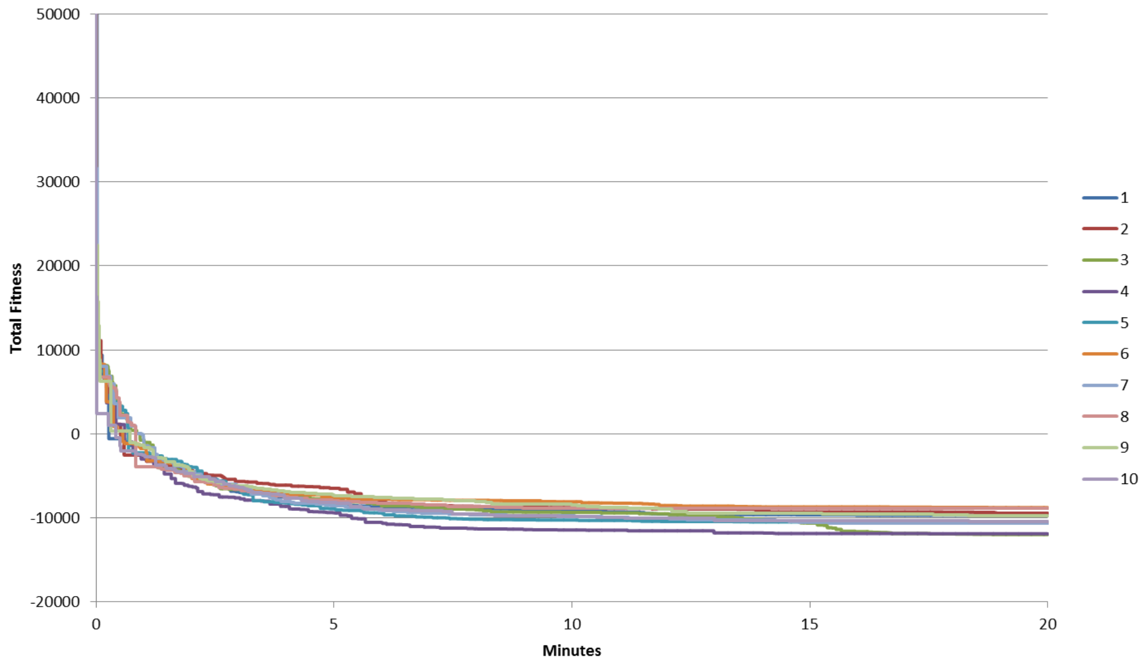


Figure C. 1 FHYB performance on scenario 1.

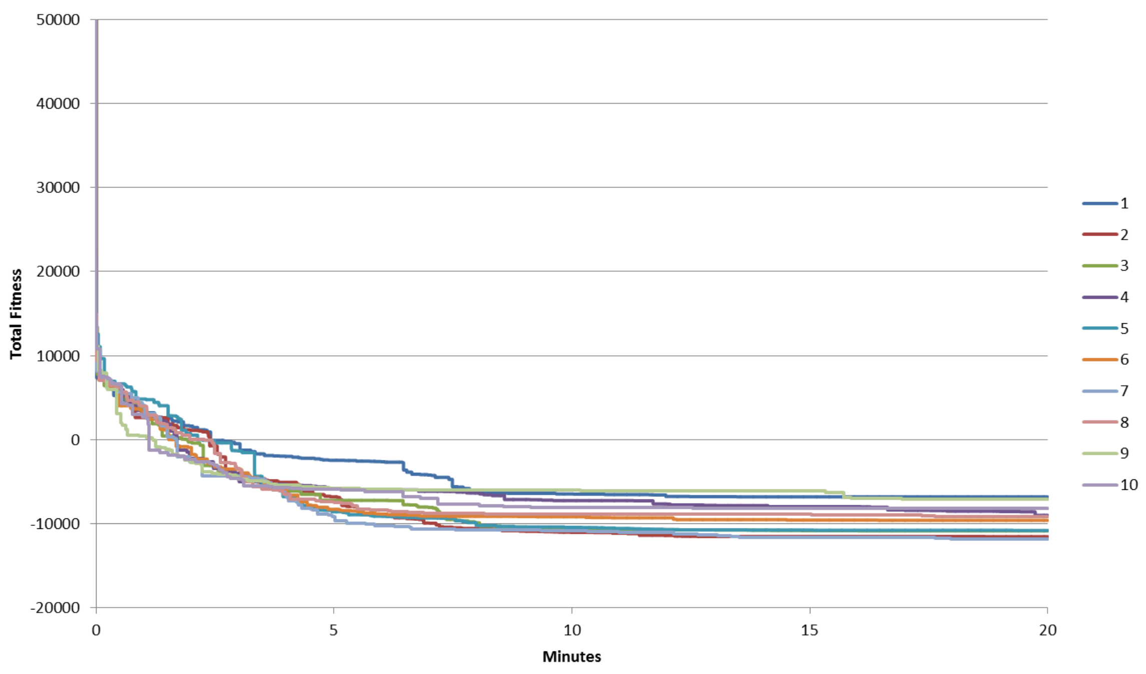


Figure C. 2 HYB performance on scenario 1.

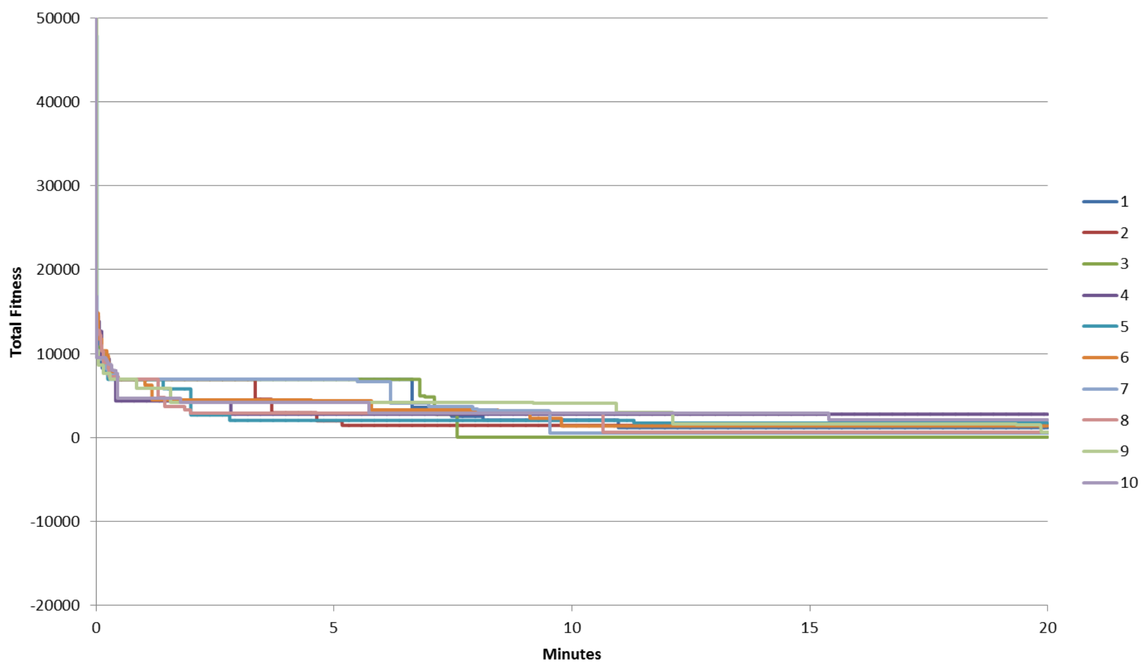


Figure C. 3 PDR performance on scenario 1.

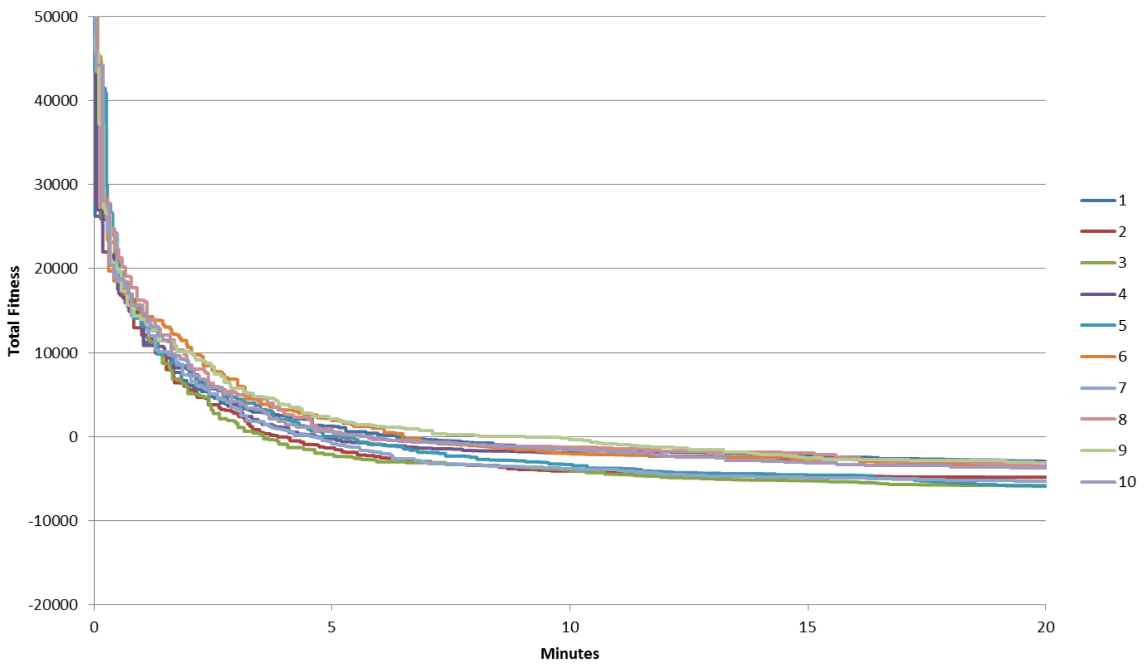


Figure C. 4 PS performance on scenario 1.

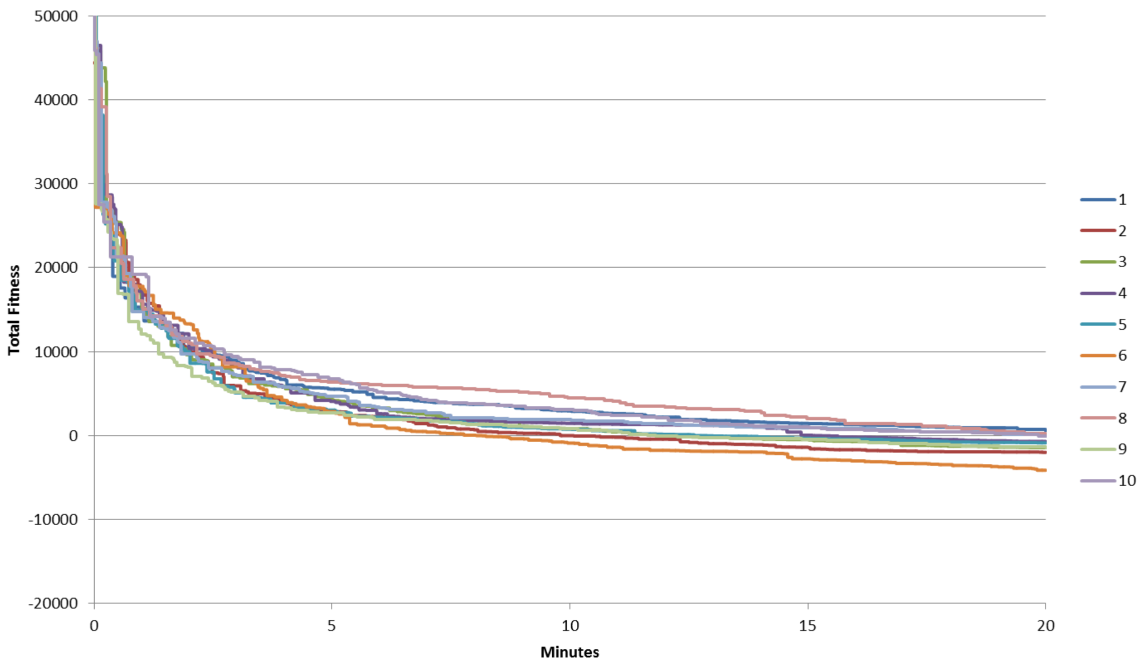


Figure C. 5 NPS performance on scenario 1.

Appendix D: Predictive Scheduling Performance Results Scenario 2

The results of scenario 2 of all replications of the predictive scheduling are presented here.

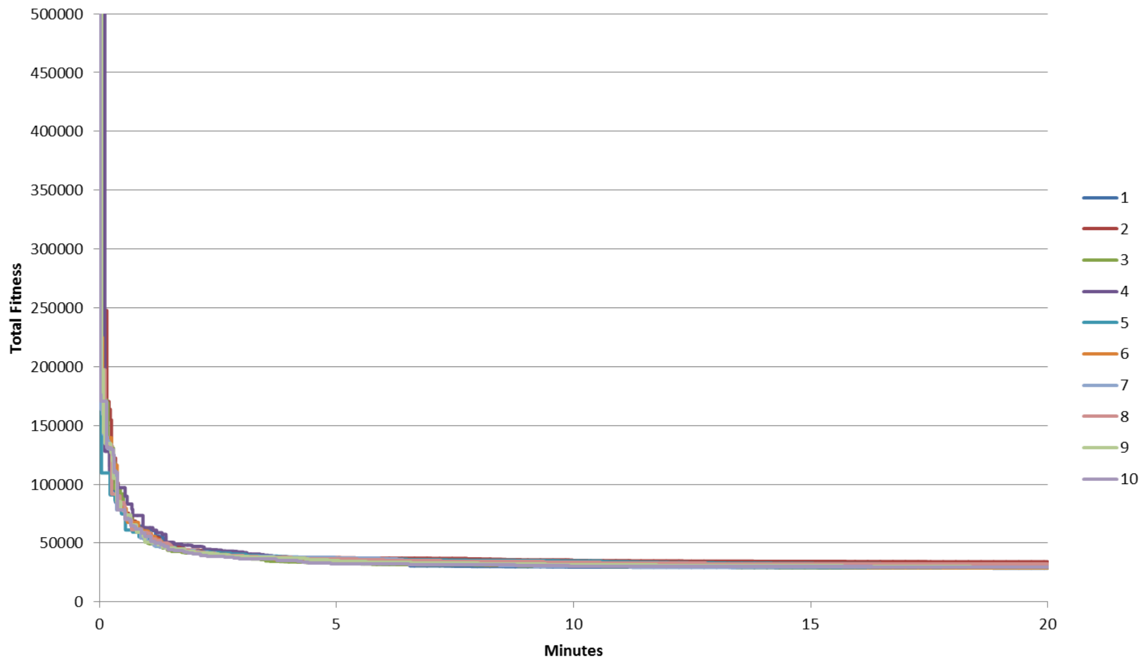


Figure D. 1 FHYB performance on scenario 2.

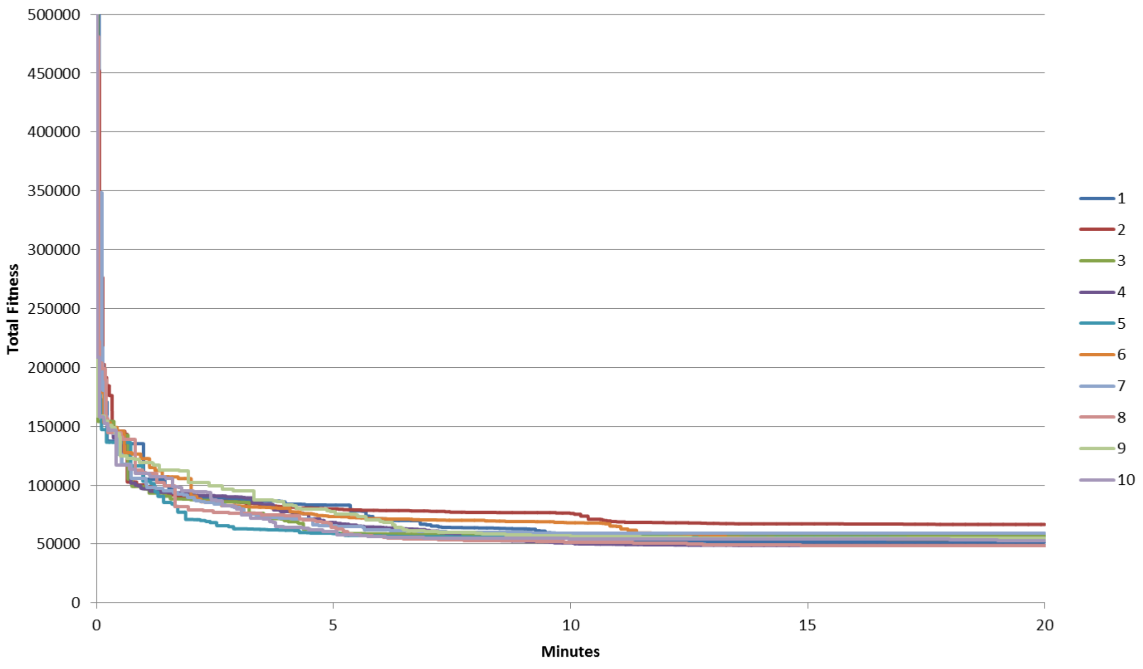


Figure D. 2 HYB performance on scenario 2.

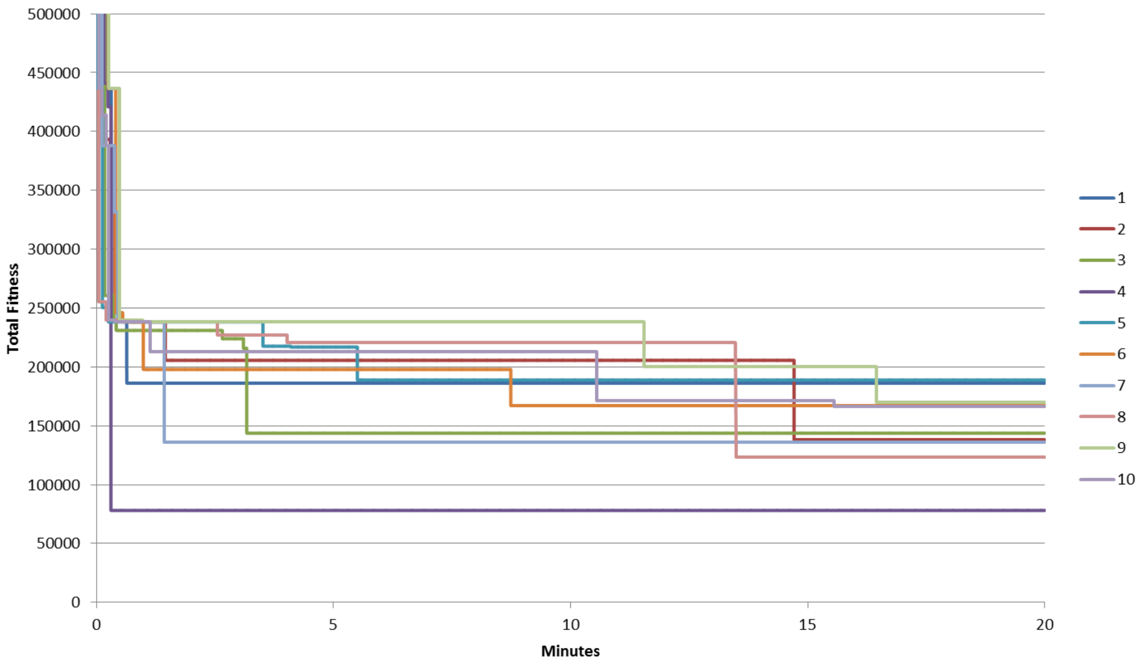


Figure D. 3 PDR performance on scenario 2.

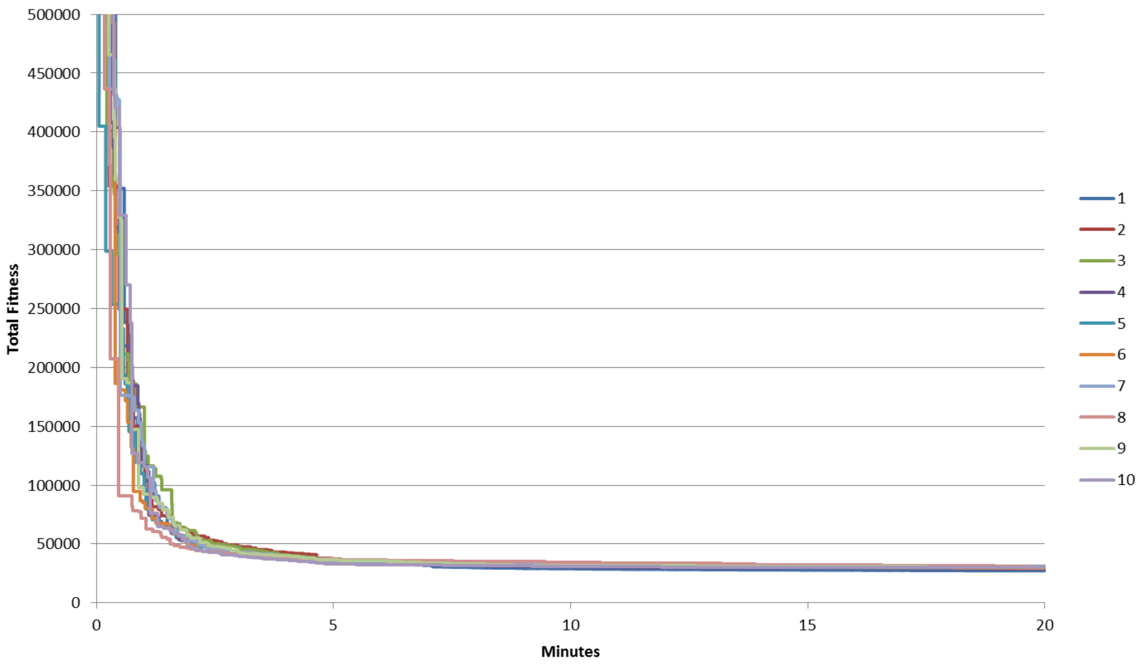


Figure D. 4 PS performance on scenario 2.

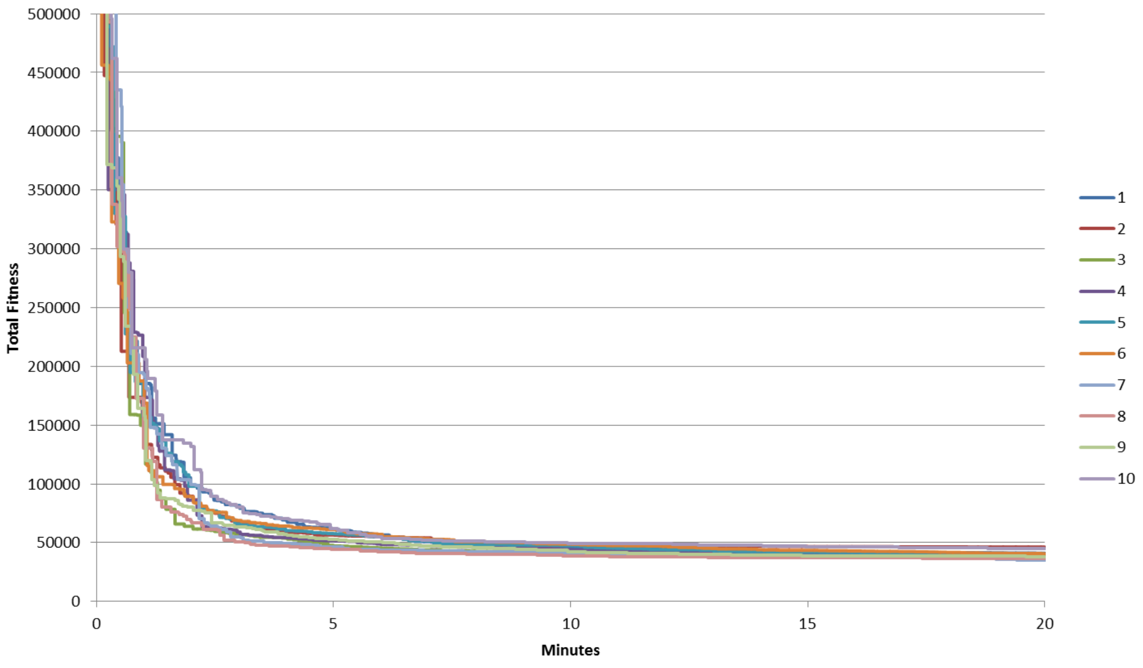


Figure D. 5 NPS performance on scenario 2.

Appendix E: Reactive Scheduling Replication Results

The results on a weekly basis of all replications of the reactive scheduling divided into the four objectives are presented here. The first objective is presented in Table E. 1, which is the number of missed target levels per week. There are totally seven target levels.

Table E. 1 Reactive scheduling results for the objective missed target levels.

Missed TL	Real-world	Indirect	Direct	FHYB
Replication 1	1.0	0.6	0.4	0.4
Replication 2	NA	0.4	0.4	0.4
Replication 3	NA	0.8	0.6	0.4
Replication 4	NA	0.8	0.4	0.4
Replication 5	NA	0.8	0.4	0.4
Replication 6	NA	0.6	0.4	0.4
Replication 7	NA	0.6	0.4	0.4
Replication 8	NA	0.8	0.4	0.4
Replication 9	NA	0.6	0.4	0.4
Replication 10	NA	0.4	0.4	0.4
Average(σ)	1.0	0.6 (0.1)	0.4 (0.1)	0.4 (0)

The second objective is presented in Table E. 2 and comprises the relative makespan compared to the real-world result. A positive number means the number of hours (production time) saved when using a certain approach.

Table E. 2 Reactive scheduling results for the objective makespan.

Makespan rel	Real-world	Indirect	Direct	FHYB
Replication 1	0.0	14.1	6.6	8.7
Replication 2	NA	12.2	4.8	8.7
Replication 3	NA	9.0	7.5	9.0
Replication 4	NA	13.0	5.4	7.7
Replication 5	NA	11.2	7.8	6.3
Replication 6	NA	13.5	6.4	7.2
Replication 7	NA	13.7	7.0	9.5
Replication 8	NA	12.8	5.5	7.6
Replication 9	NA	12.4	6.3	5.7
Replication 10	NA	8.0	8.2	7.8
Average(σ)	0.0	12 (1.9)	6.5 (1.1)	7.8 (1.1)

The third objective is presented in Table E. 3 and comprises the total number of hours that jobs have been below the different FGI security levels. A positive number means that parts have been late.

Table E. 3 Reactive scheduling results for the objective shortage hours.

Shortage hours	Real-world	Indirect	Direct	FHYB
Replication 1	0.0	3015.3	0	0
Replication 2	NA	43923.5	0	0
Replication 3	NA	3765.2	0	0
Replication 4	NA	3765.2	0	0
Replication 5	NA	3765.2	0	0
Replication 6	NA	2415.3	0	0
Replication 7	NA	708.9	0	0
Replication 8	NA	3765.2	0	0
Replication 9	NA	1265.4	0	0
Replication 10	NA	8415.3	0	0
Average(σ)	0.0	7480.5 (12305.5)	0 (0)	0 (0)

The fourth objective is presented in Table E. 4 and comprises the total setup time (hrs).

Table E. 4 Reactive scheduling results for the objective setup time.

Setup time	Real-world	Indirect	Direct	FHYB
Replication 1	NA	28.7	24.9	28.6
Replication 2	NA	23.4	23.4	26.2
Replication 3	NA	26.6	23.9	27.2
Replication 4	NA	24.1	25.0	26.5
Replication 5	NA	23.2	24.9	28.4
Replication 6	NA	27.4	24.7	27.2
Replication 7	NA	26.1	24.9	25.4
Replication 8	NA	25.6	24.7	28.4
Replication 9	NA	22.1	21.4	29.2
Replication 10	NA	24.5	20.6	28.2
Average(σ)	NA	25.2 (2)	23.8 (1.5)	27.5 (1.2)

Appendix F: Predictive and Reactive Scheduling Hypothesis Tests

Since combined dispatching rules at different stages have been demonstrated to perform better than simple PDRs according to the literature review, the PDR optimisation method was used to test real-world complex scheduling scenarios. At the same time, PDRs are good at adapting to situations where there are uncertainties and still keep the machines utilised. In the literature review, it has also been found that direct representations (PS and NPS) are better than indirect representations of using PDRs for complex scheduling scenarios. Hybrids between direct- and indirect representations have shown better results compared to dispatching rules, especially with increasing shop complexity. Furthermore, Burke at al., (2003) mean that hyper-heuristics are able to handle a wider range of problems and may lead to more general systems. A Genetic Algorithm was proposed that handles direct-, indirect- and hybrid representations within the same genetic representation, i.e., a hybrid genetic representation, which handles different scheduling approaches. The scheduling problem is complex, so it is believed that direct representations will be good and that hybrid representations may be able to handle a wider range of problems. Furthermore, it is believed that grouping jobs in the genetic algorithm may lead to better schedules, especially when there are sequence-dependent setup times.

Since the results of the FHYB and PS are quite close in both the predictive and realised scheduling results, hypothesis tests are made. These tests will show to what degree of confidence one is better than the other using a directional test, or to determine if the result of one method is different from the other, i.e., bi-directional test. The samples are independent and randomly selected from the population. The Mann-Whitney test is used since there are relatively few samples, i.e., replications. However, to complement the Mann-Whitney test, the two-sample t-test is also used for hypothesis testing to show the outcome, if the data is assumed to be normally distributed. If the p-value is higher than 0.05 (5%) the hypothesis is rejected, i.e., null hypothesis is true.

Setup reduction mutation

Different settings on the setup reduction mutation are tested. The results are compared between variant group mutation and setup group mutation for both the optimisation methods PS and NPS. The first hypothesis test is for total fitness on scenario 1, which clearly shows that it is not possible to prove that one is better than the other when using PS, see Table F. 1.

Table F. 1 PS setup reduction hypothesis test for scenario 1.

Hypothesis test, H_1: Setup groups is not equal to variant groups on scenario 1				
Total fitness	PS variant groups	PS setup groups	p-value Mann-Whitney	p-value Two-Sample T-Test
Replication 1	-4018.6	-3772.1	0.9698	0.864
Replication 2	-4757.7	-3125.9		
Replication 3	-4949.1	-3527.7		
Replication 4	-5173.8	-5326.1		
Replication 5	-4922.8	-3349.5		
Replication 6	-4122.6	-5885.1		
Replication 7	-2485.1	-2989.1		
Replication 8	-3646.8	-5927.6		
Replication 9	-3907.9	-4818.1		
Replication 10	-2997.7	-3076.9		

The second hypothesis test is for total fitness on scenario 1, which shows that using setup group mutation is better than variant group mutation for NPS, see Table F. 2. There is a probability of 0.0226 (2.3%) that the hypothesis is not true.

Table F. 2 NPS setup reduction hypothesis test for scenario 1.

Hypothesis test, H_1: Setup groups is better (lower) than variant groups on scenario 1				
Total fitness	NPS variant groups	NPS setup groups	p-value Mann-Whitney	p-value Two-Sample T-Test
Replication 1	-536.8	-27.3	0.0226	0.019
Replication 2	720.5	-1426.0		
Replication 3	-491.3	177.2		
Replication 4	1131.4	89.9		
Replication 5	1110.2	-4131.4		
Replication 6	298.7	-873.9		
Replication 7	-1155.2	-704.5		
Replication 8	-238.9	-1448.5		

Replication 9	1909.7	-1996.7		
Replication 10	-409.0	586.8		

The third hypothesis test is for total fitness on scenario 2, which shows that using setup group mutation is clearly better than variant group mutation for PS, see Table F. 3.

Table F. 3 PS setup reduction hypothesis test for scenario 2.

Hypothesis test, H_1: Setup groups is better (lower) than variant groups on scenario 2				
Total fitness	PS variant groups	PS setup groups	p-value Mann-Whitney	p-value Two-Sample T-Test
Replication 1	33235.6	29844.7	0.0001	0.000
Replication 2	33054.7	29505.1		
Replication 3	33670.0	30362.4		
Replication 4	32242.3	29999.2		
Replication 5	33000.9	30843.9		
Replication 6	33780.1	29435.6		
Replication 7	34561.4	28617.3		
Replication 8	31212.3	31082.4		
Replication 9	32354.0	29467.4		
Replication 10	34205.6	27290.1		

The fourth hypothesis test is for total fitness on scenario 2, which shows that using setup group mutation is clearly better than variant group mutation for NPS, see Table F. 4.

Table F. 4 NPS setup reduction hypothesis test for scenario 2.

Hypothesis test, H_1: Setup groups is better (lower) than variant groups on scenario 2				
Total fitness	NPS variant groups	NPS setup groups	p-value Mann-Whitney	p-value Two-Sample T-Test
Replication 1	52237.5	45148.5	0.0002	0
Replication 2	46915.4	38260.0		
Replication 3	43882.7	36453.4		
Replication 4	52144.2	35305.7		
Replication 5	46833.0	40703.4		
Replication 6	49991.5	40704.9		
Replication 7	46797.3	40834.3		
Replication 8	52037.0	38806.0		
Replication 9	46683.6	46227.2		
Replication 10	51875.3	41057.3		

Predictive scheduling

The first hypothesis test is for total fitness on scenario 1 for FHYB and HYB, which clearly shows that it is not possible to prove that one is better than the other, see Table F. 5.

Table F. 5 Predictive scheduling hypothesis test for scenario 1.

Hypothesis test, H_1: FHYB is not equal to HYB on scenario 1				
Total fitness	HYB	FHYB	p-value Mann-Whitney	p-value Two-Sample T-Test
Replication 1	-6807.5	-9714.2	0.4274	0.299
Replication 2	-11530.6	-9437.8		
Replication 3	-10861.6	-11975.7		
Replication 4	-8995.2	-11901.5		
Replication 5	-10790.5	-10516.2		
Replication 6	-9598.6	-8777.4		
Replication 7	-11860.4	-10678.6		
Replication 8	-9194.3	-8901.6		
Replication 9	-7088.1	-9729.6		
Replication 10	-8203.8	-10428.2		

The second hypothesis test is for total fitness on scenario 1, which clearly shows that FHYB is better than PS, see Table F. 6.

Table F. 6 Predictive scheduling hypothesis test for scenario 1.

Hypothesis test, H_1: FHYB is better (lower) than PS on scenario 1				
Total fitness	PS	FHYB	p-value Mann-Whitney	p-value Two-Sample T-Test
Replication 1	-3076.9	-9714.2	0.0001	0.000
Replication 2	-4818.1	-9437.8		
Replication 3	-5927.6	-11975.7		
Replication 4	-2989.1	-11901.5		
Replication 5	-5885.1	-10516.2		
Replication 6	-3346.1	-8777.4		
Replication 7	-5326.1	-10678.6		
Replication 8	-3527.7	-8901.6		
Replication 9	-3125.9	-9729.6		
Replication 10	-3772.1	-10428.2		

The third hypothesis test is for total fitness on scenario 2 for FHYB and PS, which clearly shows that it is not possible to prove that one is better than the other, see Table F. 7.

Table F. 7 Predictive scheduling hypothesis test for scenario 2.

Hypothesis test, H_1: PS is not equal to FHYB on scenario 2				
Total fitness	PS	FHYB	p-value Mann-Whitney	p-value Two-Sample T-Test
Replication 1	27290.1	28710.9	0.7337	0.274
Replication 2	29467.4	34123.5		
Replication 3	31082.4	28964.1		
Replication 4	28617.3	31346.7		
Replication 5	29435.6	31745.1		
Replication 6	29505.1	29006.4		
Replication 7	30843.9	29285.3		
Replication 8	29999.2	32237.1		
Replication 9	30381.6	29578.6		
Replication 10	29844.7	29173.4		

Reactive scheduling

The following hypothesis test is for the setup time which clearly shows that PS is better than FHYB, see Table F. 8.

Table F. 8 Reactive scheduling hypothesis test for the objective setup time.

Hypothesis test, H_1: PS is better (lower) than FHYB				
Setup time	PS	FHYB	p-value Mann-Whitney	p-value Two-Sample T-Test
Replication 1	24.9	28.6	0.0001	0.000
Replication 2	23.4	26.2		
Replication 3	23.9	27.2		
Replication 4	25.0	26.5		
Replication 5	24.9	28.4		
Replication 6	24.7	27.2		
Replication 7	24.9	25.4		
Replication 8	24.7	28.4		
Replication 9	21.4	29.2		
Replication 10	20.6	28.2		

The second hypothesis test is for the makespan, which clearly shows that FHYB is better than PS, see Table F. 9. There is a probability of 0.0156 (1.6%) that the hypothesis is not true.

Table F. 9 Reactive scheduling hypothesis test for the objective makespan.

Hypothesis test, H_1: FHYB is better (higher) than PS				
Makespan rel	PS	FHYB	p-value Mann-Whitney	p-value Two-Sample T-Test
Replication 1	6.6	8.7	0.0156	0.011
Replication 2	4.8	8.7		
Replication 3	7.5	9.0		
Replication 4	5.4	7.7		
Replication 5	7.8	6.3		
Replication 6	6.4	7.2		
Replication 7	7.0	9.5		
Replication 8	5.5	7.6		
Replication 9	6.3	5.7		
Replication 10	8.2	7.8		

Appendix G: Simulation model data

The settings and data described in this Appendix is the data used for the simulation model described in Chapter 6 and the data used in the experiments presented in Chapter 7.

Production variant information

Product variants product groups (A or B), their affiliated target levels and their security levels in the finished goods inventory.

Table G. 1 Product variant information

Variant	**Group**	**Setup group**	**Target level group**	**Security level**
Variant 1	A	1	TL2	73
Variant 2	A	2	TL1	0
Variant 3	A	2	TL1	0
Variant 4	A	2	TL1	68
Variant 5	A	2	TL1	68
Variant 6	A	2	TL1	0
Variant 7	A	3	TL1	0
Variant 8	A	3	TL1	55
Variant 9	A	4	TL2	55
Variant 10	A	4	TL2	576
Variant 11	A	4	TL2	507
Variant 12	B	5	TL3	412
Variant 13	B	6	TL4	498
Variant 14	B	6	TL5	258
Variant 15	B	6	TL6	258
Variant 16	B	6	TL7	86

Product variants production stages

The different product variants mapping to their necessary production stages.

Table G. 2 Product variants production stages

Variants	**PS1**	**PS2**	**PS3**	**PS4**	**PS5**	**PS6**	**PS7**	**PS8**	**PS9**	**PS10**
Variant 1	YES	-	-	YES	YES	YES	YES	YES	YES	YES
Variant 2	YES	-	-	YES	YES	YES	YES	YES	YES	YES
Variant 3	YES	-	-	YES	YES	YES	YES	YES	YES	YES
Variant 4	YES	-	-	YES	YES	YES	YES	YES	YES	YES
Variant 5	YES	-	-	YES	YES	YES	YES	YES	YES	YES
Variant 6	YES	-	-	YES	YES	YES	YES	YES	YES	YES
Variant 7	YES	-	-	-	YES	YES	YES	YES	-	YES
Variant 8	YES	-	-	-	YES	YES	YES	YES	-	YES
Variant 9	YES	-	-	YES	YES	YES	YES	YES	YES	YES
Variant 10	YES	-	-	YES	YES	YES	YES	YES	YES	YES
Variant 11	YES	-	-	YES	YES	YES	YES	YES	YES	YES
Variant 12	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
Variant 13	YES	-	YES	-	YES	YES	YES	YES	-	YES
Variant 14	YES	-	YES	YES	YES	YES	YES	YES	YES	YES
Variant 15	YES	-	YES	-	YES	YES	YES	YES	-	YES
Variant 16	YES	-	YES	YES	YES	YES	YES	YES	YES	YES

Machine settings

The following table shows whether the machines, e.g., PS1M1 (production stage 1, parallel machine 1), are on or off and what product variants they are set to produce.

Table G. 3 Machine settings

Variants	**PS1M1**	**PS1M2**	**PS1M3**	**PS1M4**	**PS1M5**	**PS2M1**	**PS3M1**	**PS4M1**	**PS4M2**	**PS4M3**
Operation-status	ON	ON	ON	ON	OFF	ON	ON	ON	ON	ON
Variant 1	YES	-	-	YES	YES	-	-	YES	YES	YES
Variant 2	YES	-	-	YES	YES	-	-	YES	YES	YES
Variant 3	YES	-	-	YES	YES	-	-	YES	YES	YES
Variant 4	YES	-	-	YES	YES	-	-	YES	YES	YES
Variant 5	YES	-	-	YES	YES	-	-	YES	YES	YES
Variant 6	YES	-	-	YES	YES	-	-	YES	YES	YES
Variant 7	YES	-	-	YES	YES	-	-	-	-	-
Variant 8	YES	-	-	YES	YES	-	-	-	-	-
Variant 9	YES	-	-	YES	YES	-	-	YES	YES	YES
Variant 10	YES	-	-	YES	YES	-	-	YES	YES	YES
Variant 11	YES	-	-	YES	YES	-	-	YES	YES	YES
Variant 12	-	YES	YES	-	-	YES	YES	-	-	YES
Variant 13	-	YES	YES	-	-	-	YES	-	-	-
Variant 14	-	YES	YES	-	-	-	YES	-	-	YES
Variant 15	-	YES	YES	-	-	-	YES	-	-	-
Variant 16	-	YES	YES	-	-	-	YES	-	-	YES

Variants	**PS5M1**	**PS5M2**	**PS5M3**	**PS6M1**	**PS6M2**	**PS6M3**	**PS6M4**	**PS6M5**	**PS6M6**	**PS6M7**
Operation-status	ON	ON	ON	OFF	OFF	ON	ON	ON	ON	ON
Variant 1	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 2	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 3	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 4	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 5	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 6	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 7	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 8	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 9	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 10	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 11	-	-	YES	-	YES	-	YES	YES	YES	-
Variant 12	YES	-	-	-	-	-	-	-	-	YES
Variant 13	-	YES	-	YES	-	YES	YES	-	-	-
Variant 14	-	YES	-	YES	-	YES	YES	-	-	-
Variant 15	-	YES	-	YES	-	YES	YES	-	-	-
Variant 16	-	YES	-	YES	-	YES	YES	-	-	-

Variants	**PS7M1**	**PS7M2**	**PS7M3**	**PS8M1**	**PS9M1**	**PS9M2**	**PS10M1**	**PS10M2**
Operation-status	ON	ON	ON	ON	ON	ON	ON	ON
Variant 1	YES	YES	-	YES	YES	YES	YES	YES
Variant 2	YES	YES	-	YES	YES	YES	YES	YES
Variant 3	YES	YES	-	YES	YES	YES	YES	YES
Variant 4	YES	YES	-	YES	YES	YES	YES	YES
Variant 5	YES	YES	-	YES	YES	YES	YES	YES
Variant 6	YES	YES	-	YES	YES	YES	YES	YES
Variant 7	YES	YES	-	YES	-	-	YES	YES
Variant 8	YES	YES	-	YES	-	-	YES	YES
Variant 9	YES	YES	-	YES	YES	YES	YES	YES
Variant 10	YES	YES	-	YES	YES	YES	YES	YES
Variant 11	YES	YES	-	YES	YES	YES	YES	YES
Variant 12	-	-	YES	YES	YES	YES	YES	YES
Variant 13	-	YES	-	YES	-	-	YES	YES
Variant 14	-	YES	-	YES	YES	YES	YES	YES
Variant 15	-	YES	-	YES	-	-	YES	YES
Variant 16	-	YES	-	YES	YES	YES	YES	YES

Machine availability

The availability of machines is presented here. However, the experiments executed in Chapter 7 used deterministic settings, i.e., the processing time of the machines was divided by the availability number, as shown in the Table.

Table G. 4 Availability of machines

Operation	MTBF (min)	MTTR (min)	Availability
PS0M1	90	15	0.85714
PS0M2	90	10	0.90000
PS0M3	170	10	0.94444
PS1M1	72	8	0.90000
PS1M2	55	10	0.84615
PS1M3	60	15	0.80000
PS1M4	72	12	0.85714
PS1M5	72	8	0.90000
PS2M1	72	8	0.90000
PS3M1	80	8	0.90909
PS4M1	50	10	0.83333
PS4M2	50	8	0.86207
PS4M3	50	12	0.80645
PS5M1	60	8	0.88235
PS5M2	80	14	0.85106
PS5M3	60	14	0.81081
PS6M1	65	25	0.72222
PS6M2	60	15	0.80000
PS6M3	68	12	0.85000
PS6M4	60	7	0.89552
PS6M5	50	12	0.80645
PS6M6	60	25	0.70588
PS6M7	60	15	0.80000
PS7M1	68	12	0.85000
PS7M2	68	12	0.85000
PS7M3	70	10	0.87500
PS8M1	150	5	0.96774
PS9M1	140	6	0.95890
PS9M2	140	7	0.95238
PS10M1	72	8	0.90000
PS10M2	72	8	0.90000

Processing times

Table G. 5 Processing times (seconds) of product variants at different machines

	Variant 1	Variant 2	Variant 3	Variant 4	Variant 5	Variant 6	Variant 7	Variant 8
PS1M1	84.575	84.3	84.3	84.3	84.3	84.3	97.55	97.075
PS1M2	91	91.25	91.25	91.25	91.25	91.25	102	102
PS1M3	86	86.5	86.5	86.5	86.5	86.5	101.5	101.5
PS1M4	78.65	81	81	81	81	81	96.675	96.675
PS1M5	78.75	78.75	78.75	78.75	78.75	78.75	193.9	193.9
PS2M1	-	-	-	-	-	-	-	-
PS3M1	-	-	-	-	-	-	-	-
PS4M1	58.525	57.5	57.5	57.5	57.5	57.5	-	-
PS4M2	58.225	63.5	63.5	63.5	63.5	63.5	-	-
PS4M3	61	63.5	63.5	63.5	63.5	63.5	-	-
PS5M1	-	-	-	-	-	-	-	-
PS5M2	-	-	-	-	-	-	-	-
PS5M3	39.6	39.6	39.6	39.6	39.6	39.6	38.91	38.91
PS6M1	-	-	-	-	-	-	-	-
PS6M2	88.5	85.125	85.125	85.125	85.125	85.125	85.625	85.625
PS6M3	-	-	-	-	-	-	-	-
PS6M4	80	80	80	80	80	80	80	80
PS6M5	138	152.9	152.9	152.9	152.9	152.9	159	159
PS6M6	69.6	69.55	69.55	69.55	69.55	69.55	69.75	69.75
PS6M7	-	-	-	-	-	-	-	-
PS7M1	33.3	33.3	33.3	33.3	33.3	33.3	33.3	33.3
PS7M2	33.3	33.3	33.3	33.3	33.3	33.3	33.3	33.3
PS7M3	-	-	-	-	-	-	-	-
PS8M1	13.6	15.4	15.4	15.4	15.4	15.4	13.6	13.6
PS9M1	32	32.5	32.5	32.5	32.5	32.5	-	-
PS9M2	32	30.9	30.9	30.9	30.9	30.9	-	-
PS10M1	30.4	32.5	32.5	32.5	32.5	32.5	32.7	32.7
PS10M2	24.4	24.4	24.4	24.4	24.4	24.4	23.5	23.5

	Variant 9	Variant 10	Variant 11	Variant 12	Variant 13	Variant 14	Variant 15	Variant 16
PS1M1	84.575	84.575	84.575	-	-	-	-	-
PS1M2	91	91	91	95	99.75	88	100.5	99.75
PS1M3	86	86	86	87.5	96.5	84.5	97	96.5
PS1M4	78.65	78.65	78.65	-	-	-	-	-
PS1M5	157.5	157.5	157.5	185	180	180	180	180
PS2M1	-	-	-	123.9	-	-	-	-
PS3M1	-	-	-	48.11	48.47	51.5	48.47	48.47
PS4M1	58.525	58.525	58.525	-	-	-	-	-
PS4M2	58.225	58.225	58.225	-	-	-	-	-
PS4M3	61	61	61	61.4	-	61.2	-	61.2
PS5M1	-	-	-	49.25	43.1	43.1	43.1	43.1
PS5M2	-	-	-	-	43.2	43.2	43.2	43.2
PS5M3	40.1	40.1	40.1	-	-	-	-	-
PS6M1	-	-	-	-	206.4	206.4	206.4	206.4
PS6M2	88.5	88.5	88.5	-	-	-	-	-
PS6M3	-	-	-	-	111	105.5	105.5	111
PS6M4	80	80	80	-	80.5	86.5	88	80.5
PS6M5	138	138	138	-	-	-	-	-
PS6M6	69.6	69.6	69.6	-	-	-	-	-
PS6M7	-	-	-	98.5	-	-	-	-
PS7M1	33.3	33.3	33.3	-	-	-	-	-
PS7M2	33.3	33.3	33.3	-	44	44	44	44
PS7M3	-	-	-	54.65	44.5	49.8	49.8	44.5
PS8M1	13.6	13.6	13.6	13.6	13.6	13.6	13.6	13.6
PS9M1	32	32	32	32.5	-	32	-	32
PS9M2	32	32	32	32	-	32	-	32
PS10M1	30.4	30.4	30.4	42	37	37	37	37
PS10M2	24.4	24.4	24.4	42	37	23.5	37	37

Sequence-dependent setup times

The sequence-dependent setup times (minutes) are presented here. Some of the sequence-dependent setup times will not be used because the product variants may not be allowed to be produced in the machines due to the “machine settings” presented earlier. The setup times for the bottleneck production stages are presented in detail.

The sequence-dependent setup times for production stage 1 and machine 2 can be seen in the Table. All of the machines in production stage 1 have the same sequence-dependent setup times.

Table G. 6 Sequence-dependent setup time production stage 1

PS1M2	V 1	V 2	V 3	V 4	V 5	V 6	V 7	V 8	V 9	V 10	V 11	V 12	V 13	V 14	V 15	V 16
V 1	0	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	60	60	60	60	60
V 2	11.7	0	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	60	60	60	60	60
V 3	11.7	11.7	0	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	60	60	60	60	60
V 4	11.7	11.7	11.7	0	11.7	11.7	11.7	11.7	11.7	11.7	11.7	60	60	60	60	60
V 5	11.7	11.7	11.7	11.7	0	11.7	11.7	11.7	11.7	11.7	11.7	60	60	60	60	60
V 6	11.7	11.7	11.7	11.7	11.7	0	11.7	11.7	11.7	11.7	11.7	60	60	60	60	60
V 7	11.7	11.7	11.7	11.7	11.7	11.7	0	11.7	11.7	11.7	11.7	60	60	60	60	60
V 8	11.7	11.7	11.7	11.7	11.7	11.7	11.7	0	11.7	11.7	11.7	60	60	60	60	60
V 9	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	0	11.7	11.7	60	60	60	60	60
V 10	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	0	11.7	60	60	60	60	60
V 11	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	0	60	60	60	60	60
V 12	60	60	60	60	60	60	60	60	60	60	60	0	11.7	11.7	11.7	11.7
V 13	60	60	60	60	60	60	60	60	60	60	60	11.7	0	6	6	6
V 14	60	60	60	60	60	60	60	60	60	60	60	11.7	6	0	6	6
V 15	60	60	60	60	60	60	60	60	60	60	60	11.7	6	6	0	6
V 16	60	60	60	60	60	60	60	60	60	60	60	11.7	6	6	6	0

Production stage 2 has no sequence-dependent setup time. Production stage 3 has 5 minutes setup time between the two different product variants. Production stage 4 has 6.1 minutes setup time between products within the same product group (A or B) and the setup time between the groups is 20 minutes. Setup times for production stage 5 can be seen in Table

Table G. 7 Sequence-dependent setup time production stage 5

PS5M1	V 1	V 2	V 3	V 4	V 5	V 6	V 7	V 8	V 9	V 10	V 11	V 12	V 13	V 14	V 15	V 16
V 1	0	5	5	5	5	5	5	5	5	5	5	1440	1440	1440	1440	1440
V 2	5	0	5	5	5	5	5	5	5	5	5	1440	1440	1440	1440	1440
V 3	5	5	0	5	5	5	5	5	5	5	5	1440	1440	1440	1440	1440
V 4	5	5	5	0	5	5	5	5	5	5	5	1440	1440	1440	1440	1440
V 5	5	5	5	5	0	5	5	5	5	5	5	1440	1440	1440	1440	0
V 6	5	5	5	5	5	0	5	5	5	5	5	1440	1440	1440	1440	1440
V 7	5	5	5	5	5	5	0	5	5	5	5	1440	1440	1440	1440	1440
V 8	5	5	5	5	5	5	5	0	5	5	5	1440	1440	1440	1440	1440
V 9	5	5	5	5	5	5	5	5	0	5	5	1440	1440	1440	1440	1440
V 10	5	5	5	5	5	5	5	5	5	0	5	1440	1440	1440	1440	1440
V 11	5	5	5	5	5	5	5	5	5	5	0	1440	1440	1440	1440	1440
V 12	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	0	90	90	90	90
V 13	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	90	0	6	6	6
V 14	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	90	6	0	6	6
V 15	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	90	6	6	0	6
V 16	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	90	6	6	6	0

PS5M2	V 1	V 2	V 3	V 4	V 5	V 6	V 7	V 8	V 9	V 10	V 11	V 12	V 13	V 14	V 15	V 16
V 1	0	5	5	5	5	5	5	5	5	5	5	1440	1440	1440	1440	1440
V 2	5	0	5	5	5	5	5	5	5	5	5	1440	1440	1440	1440	1440
V 3	5	5	0	5	5	5	5	5	5	5	5	1440	1440	1440	1440	1440
V 4	5	5	5	0	5	5	5	5	5	5	5	1440	1440	1440	1440	1440
V 5	5	5	5	5	0	5	5	5	5	5	5	1440	1440	1440	1440	7.5
V 6	5	5	5	5	5	0	5	5	5	5	5	1440	1440	1440	1440	1440
V 7	5	5	5	5	5	5	0	5	5	5	5	1440	1440	1440	1440	1440
V 8	5	5	5	5	5	5	5	0	5	5	5	1440	1440	1440	1440	1440
V 9	5	5	5	5	5	5	5	5	0	5	5	1440	1440	1440	1440	1440
V 10	5	5	5	5	5	5	5	5	5	0	5	1440	1440	1440	1440	1440
V 11	5	5	5	5	5	5	5	5	5	5	0	1440	1440	1440	1440	1440
V 12	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	0	1440	1440	1440	1440
V 13	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	0	6	6	6
V 14	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	6	0	6	6
V 15	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	6	6		6
V 16	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	6	6	6	

PS5M3	V 1	V 2	V 3	V 4	V 5	V 6	V 7	V 8	V 9	V 10	V 11	V 12	V 13	V 14	V 15	V 16
V 1	0	120	120	120	120	120	120	120	5	5	5	1440	1440	1440	1440	1440
V 2	120	0	5	5	5	5	120	120	120	120	120	1440	1440	1440	1440	1440
V 3	120	5	0	5	5	5	120	120	120	120	120	1440	1440	1440	1440	1440
V 4	120	5	5	0	5	5	120	120	120	120	120	1440	1440	1440	1440	1440
V 5	120	5	5	5	0	5	120	120	120	120	120	1440	1440	1440	1440	7.5
V 6	120	5	5	5	5	0	120	120	120	120	120	1440	1440	1440	1440	1440
V 7	120	120	120	120	120	120	0	5	120	120	120	1440	1440	1440	1440	1440
V 8	120	120	120	120	120	120	5	0	120	120	120	1440	1440	1440	1440	1440
V 9	5	120	120	120	120	120	120	120	0	5	5	1440	1440	1440	1440	1440
V 10	5	120	120	120	120	120	120	120	5	0	5	1440	1440	1440	1440	1440
V 11	5	120	120	120	120	120	120	120	5	5	0	1440	1440	1440	1440	1440
V 12	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	0	120	120	120	120
V 13	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	120	0	120	120	120
V 14	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	120	120	0	120	120
V 15	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	120	120	120	0	120
V 16	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	1440	120	120	120	120	0

Production stage 6 has a setup time of 7.5 minutes between product variants within the product group A, 5-15 minutes between product variants within product group B and 15 minutes between the groups. Production stage 7 has a setup time of 0.4 minutes between product variants within the product group A, 0 minutes between product variants within product group B and 45 minutes between the groups. Production stage 8 has no setup time. Production stages 9 and 10 have a setup time of 0.4 minutes between different product variants.

Machine-dependent setup times

The machine-dependent setup times (minutes) are loading times (only affects some of the machines) and start up time required when a machine has been empty in between products.

Table G. 8 Machine-dependent setup times

	Loading time	Start-up time
PS0M1	0	0
PS0M2	0	0
PS0M3	0	0
PS1M1	0	0
PS1M2	0	0
PS1M3	0	0
PS1M4	0	0
PS1M5	0	0
PS2M1	0	0
PS3M1	0	0
PS4M1	0	0
PS4M2	0	0
PS4M3	0	0
PS5M1	0	10
PS5M2	0	10
PS5M3	0	10
PS6M1	0.66666667	0
PS6M2	0.66666667	0
PS6M3	0.66666667	0
PS6M4	0.66666667	0
PS6M5	0.66666667	0
PS6M6	0.66666667	0
PS6M7	0.66666667	0
PS7M1	0.66666667	0
PS7M2	0.66666667	0
PS7M3	0.66666667	0
PS8M1	2	0
PS9M1	2	0
PS9M2	2	0
PS10M1	2	0
PS10M2	2	0

Work in process

The first table shows the work in process in the production line at the start of the simulation (Scenario 1 and Scenario 2 in Chapter 7). The second table shows the target levels of WIP.

Table G. 9 Work in process

Production stage	Variant	Number
PS10	Variant 5	50
PS10	Variant 5	50
PS10	Variant 10	50
PS10	Variant 10	50
PS10	Variant 12	50
PS9	Variant 5	50
PS9	Variant 5	50
PS9	Variant 11	50
PS9	Variant 12	50
PS8	Variant 10	50
PS8	Variant 15	50
PS8	Variant 15	50
PS8	Variant 15	50
PS8	Variant 15	50
PS7	Variant 5	50
PS7	Variant 5	50
PS7	Variant 10	50
PS7	Variant 11	50
PS7	Variant 12	50
PS7	Variant 12	50
PS7	Variant 12	50
PS7	Variant 12	50
PS7	Variant 15	50
PS7	Variant 15	50
PS7	Variant 15	50
PS7	Variant 15	50
PS6	Variant 4	50
PS6	Variant 4	50
PS6	Variant 4	50
PS6	Variant 4	50
PS6	Variant 10	50
PS6	Variant 10	50
PS6	Variant 10	50
PS6	Variant 10	50
PS6	Variant 10	50
PS6	Variant 10	50
PS6	Variant 10	50
PS6	Variant 10	50
PS6	Variant 10	50
PS6	Variant 12	50
PS6	Variant 12	50
PS6	Variant 12	50
PS6	Variant 12	50
PS6	Variant 14	50
PS6	Variant 14	50
PS6	Variant 14	50
PS6	Variant 14	50
PS6	Variant 14	50
PS5	Variant 10	50
PS5	Variant 11	50
PS5	Variant 11	50
PS5	Variant 11	50
PS5	Variant 11	50
PS5	Variant 13	50
PS5	Variant 13	50
PS5	Variant 13	50
PS5	Variant 13	50
PS5	Variant 13	50
PS5	Variant 13	50
PS5	Variant 13	50
PS5	Variant 13	50
PS5	Variant 13	50
PS5	Variant 14	50
PS5	Variant 14	50
PS5	Variant 14	50
PS5	Variant 14	50
PS5	Variant 14	50
PS5	Variant 14	50
PS4	Variant 10	50
PS4	Variant 10	50
PS4	Variant 10	50
PS4	Variant 10	50
PS4	Variant 12	50
PS4	Variant 12	50
PS4	Variant 12	50
PS4	Variant 12	50
PS4	Variant 12	50
PS3	Variant 12	50
PS3	Variant 12	50
PS3	Variant 12	50
PS3	Variant 12	50
PS3	Variant 12	50
PS2	Variant 12	50
PS2	Variant 12	50
PS2	Variant 12	50
PS2	Variant 12	50
PS2	Variant 12	50

Table G. 10 Target levels for WIP

Targets WIP	Week 1	Week 2
Group	**Number**	**Number**
A	1900	1900
B	2450	2450

Finished goods inventory

Information regarding the finished goods inventory is presented here.

Table G. 11 Physical maximum of the FGI

Group	Number
A	9600
B	115000

Table G. 12 Number of parts in FGI at start of simulation

Variant	Number
Variant 1	374
Variant 2	0
Variant 3	0
Variant 4	222
Variant 5	148
Variant 6	0
Variant 7	0
Variant 8	461
Variant 9	547
Variant 10	715
Variant 11	669
Variant 12	1020
Variant 13	2210
Variant 14	1190
Variant 15	1190
Variant 16	680
-	-
Totally	9052

Table G. 13 Target levels of the finished goods inventory

Target Levels FGI	Week 1	Week 2
Variant	**Number**	**Number**
TL1	380	380
TL2	2421	2421
TL3	1673	1673
TL4	2022	2022
TL5	1045	1045
TL6	1045	1045
TL7	348	348
-	-	-
Totally	8934	8934

Demands

The following demands were used for the two production weeks of Scenario 1.

Table G. 14 Demands Scenario 1

Week 1	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Variant	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**
Variant 1	72.65	36.32	42.38	36.32	133.19	0.00	0.00
Variant 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 4	66.59	36.32	30.27	36.32	36.32	0.00	0.00
Variant 5	66.59	36.32	30.27	36.32	36.32	0.00	0.00
Variant 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 8	90.81	54.49	42.38	60.54	48.43	0.00	0.00
Variant 9	90.81	54.49	42.38	60.54	48.43	0.00	0.00
Variant 10	371.31	297.66	347.10	286.56	496.43	0.00	0.00
Variant 11	334.99	261.33	304.72	250.23	363.24	0.00	0.00
Variant 12	0.00	343.06	0.00	514.59	0.00	0.00	0.00
Variant 13	0.00	514.59	0.00	514.59	0.00	0.00	0.00
Variant 14	0.00	514.59	0.00	514.59	0.00	0.00	0.00
Variant 15	0.00	514.59	0.00	514.59	0.00	0.00	0.00
Variant 16	0.00	171.53	0.00	0.00	0.00	0.00	0.00

Week 2	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Variant	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**
Variant 1	24.22	12.11	12.11	12.11	0.00	0.00	0.00
Variant 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 4	36.32	36.32	30.27	48.43	24.22	0.00	0.00
Variant 5	36.32	36.32	30.27	48.43	24.22	0.00	0.00
Variant 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 8	48.43	42.38	66.59	85.77	0.00	0.00	0.00
Variant 9	48.43	42.38	66.59	85.77	0.00	0.00	0.00
Variant 10	357.19	304.72	308.75	229.04	199.78	0.00	0.00
Variant 11	332.97	292.61	296.65	216.94	199.78	0.00	0.00
Variant 12	0.00	343.06	0.00	514.59	0.00	0.00	0.00
Variant 13	0.00	343.06	0.00	686.12	0.00	0.00	0.00
Variant 14	0.00	171.53	0.00	171.53	0.00	0.00	0.00
Variant 15	0.00	171.53	0.00	171.53	0.00	0.00	0.00
Variant 16	0.00	171.53	0.00	0.00	0.00	0.00	0.00

The following demands were used for the two production weeks of Scenario 2.

Table G. 15 Demands Scenario 2

Week 1	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Variant	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**
Variant 1	72.65	36.32	42.38	36.32	133.19	0.00	0.00
Variant 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 3	100.00	100.00	0.00	0.00	0.00	0.00	0.00
Variant 4	66.59	36.32	30.27	36.32	36.32	0.00	0.00
Variant 5	66.59	36.32	30.27	36.32	36.32	0.00	0.00
Variant 6	100.00	100.00	0.00	0.00	0.00	0.00	0.00
Variant 7	100.00	100.00	0.00	0.00	0.00	0.00	0.00
Variant 8	490.00	54.49	142.00	60.54	48.43	0.00	0.00
Variant 9	90.81	154.00	42.38	60.54	48.43	0.00	0.00
Variant 10	371.31	297.66	347.10	286.56	496.43	0.00	0.00
Variant 11	334.99	261.33	304.72	250.23	363.24	0.00	0.00
Variant 12	0.00	343.06	0.00	514.59	0.00	0.00	0.00
Variant 13	0.00	514.59	0.00	514.59	0.00	0.00	0.00
Variant 14	0.00	514.59	0.00	514.59	0.00	0.00	0.00
Variant 15	0.00	514.59	0.00	514.59	0.00	0.00	0.00
Variant 16	0.00	171.53	0.00	0.00	0.00	0.00	0.00

Week 2	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Variant	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**	**Demand**
Variant 1	24.22	12.11	12.11	12.11	0.00	0.00	0.00
Variant 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 4	36.32	36.32	30.27	48.43	24.22	0.00	0.00
Variant 5	36.32	36.32	30.27	48.43	24.22	0.00	0.00
Variant 6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Variant 8	48.43	42.38	66.59	85.77	0.00	0.00	0.00
Variant 9	48.43	42.38	66.59	85.77	0.00	0.00	0.00
Variant 10	357.19	304.72	308.75	229.04	199.78	0.00	0.00
Variant 11	332.97	292.61	296.65	216.94	199.78	0.00	0.00
Variant 12	0.00	343.06	0.00	514.59	0.00	0.00	0.00
Variant 13	0.00	343.06	0.00	686.12	0.00	0.00	0.00
Variant 14	0.00	171.53	0.00	171.53	0.00	0.00	0.00
Variant 15	0.00	171.53	0.00	171.53	0.00	0.00	0.00
Variant 16	0.00	171.53	0.00	0.00	0.00	0.00	0.00

Buffers and carriers

As explained in Chapter 6, there are three different buffers, production stage buffers, input buffers at the machines and output buffers at the machines. There are also a number of carriers transporting the products variants for different parts of the production line.

Table G. 16 Production stage buffers

PS buffer	Number
PS1	0
PS2	750
PS3	750
PS4	750
PS5	1500
PS6	1500
PS7	800
PS8	500
PS9	500
PS10	500

Table G. 17 Input and output buffers at the machines (number of carriers)

Machine	Input buffer	Output buffer
PS1M1	2	15
PS1M2	2	15
PS1M3	2	15
PS1M4	2	15
PS1M5	2	15
PS2M1	15	15
PS3M1	15	15
PS4M1	15	15
PS4M2	15	15
PS4M3	15	15
PS5M1	15	2
PS5M2	15	2
PS5M3	15	2
PS6M1	3	3
PS6M2	3	3
PS6M3	3	3
PS6M4	3	3
PS6M5	3	3
PS6M6	3	3
PS6M7	3	3
PS7M1	2	5
PS7M2	2	5
PS7M3	2	5
PS8M1	15	15
PS9M1	15	15
PS9M2	15	15
PS10M1	15	15
PS10M2	15	15

Table G. 18 Number of carriers

Carrier loops	Number
PS1-PS5	66
PS6-PS7	34
PS8-PS10	20

Appendix H: Simulation model in C#

The simulation model was built in C#. A flow chart shows how the simulation model has been built. Example code (pseudo code) has also been included in this Appendix. The simulation model is built on the basics of discrete event simulation, in which the event list is used to drive the simulation model, see event list pseudo code below.

Eventlist pseudo code

```
while (eventQueue.Count > 0)
{
   SimulationEvent se = (SimulationEvent)eventQueue.Dequeue();
   if (se.Time >= timeToSimulate)
   {
                  /* We have reached the end of the simulation horizon */
                  break;
   }
   OnEventPreExecution(se);
   se.Execute(se.Time);
   OnEventPostExecution(se);
}
```

Different events are executed, before or after the deletion of events, from the list of events until the simulation horizon, i.e., two weeks, is reached. The different events currently available in the simulation model are:

- ProductVariantArrivedAtMachineEvent
- ProductVariantArrivedAtProductionStageEvent
- ProductionStartedEvent
- ProductionCompletedEvent
- SetupCompleteEvent
- WarmupCompleteEvent
- FgiPickoutEvent
- ControlBuffersEvent

The ProductionStartedEvent, i.e., to the event that starts jobs at the machines, will be explained further. When the ProductionStartedEvent is executed, it will retrieve information about which machine it is and use the rule set for the production stage, i.e., a “schedule” (direct representation) or “dispatching rule” (indirect or hybrid representation). When the rule is “schedule”, the correct variant (next in sequence) has to be available at the input buffer or else try to start a job the next time it is triggered. If a dispatching rule is used, it will select a job (“machine.StartProductVariant”) based on the jobs available in front of the machines, e.g., first-come-first-served (FCFS). After the selection has been made, it is deleted from the list of jobs.

ProductionStartedEvent pseudo code

```
/* This event is triggered when a product variant arrives at a production stage, when the machine is complete, etc
public class ProductionStarted : public SimulationEvent
{
  private Machine machine;
  public ProductionStarted(Machine machine)
  {
        this.machine = machine;
        public override void Execute(float elapsedSimulationTime)
        Buffer buffer = machine.ProductionStage.Buffer;
        DispatchRule dispatch = machine.ProductionStage.DispatchRule;
        if (dispatch == "schedule")
        {
                /* Get the next product variant in the schedule */
                ProductVariant nextProductVariant = GetNextProductVariant();
                foreach (ProductVariant productVariant in buffer)
                {
                        if (productVariant == nextProductVariant)
                        {
                                buffer.Remove(productVariant);
                                machine.StartProductVariant(productVariant);
                        }
                }
        }
        else
        {
                /* Choose a production variant from the buffer with the current dispatching rule. */
                ProductVariant productVariant = machine.ChooseProductGroup(buffer, dispatch);
                /* Make sure that a product variant has been selected */
                if (productVariant != null)
                {
                        buffer.Remove(productVariant);
                        machine.StartProductVariant(productVariant);
                }
        }
  }
}
```

When the job has been selected, it will execute machine.StartProductVariant(productVariant) in the machine code. If a sequence-dependent setup time is required, the event SetupComplete will be added to the event list with the time required for the sequence-dependent setup time and then exit the method. The machine will be triggered again from the SetupCompleteEvent.

Machine pseudo code

```
public class Machine
{
  ProductVariant currentProductVariant;
  string name;
  public Machine(string name)
      {
          this.name = name;
          currentProductVariant = null;
      }
      public void SetCurrentProductVariant(ProductVariant productVariant)
      {
          currentProductVariant = productVariant;
      }
      public void StartProductVariant(ProductVariant productVariant)
      {
          if (productVariant != currentProductVariant)
          {
              double setupTime = getSetupTime(productVariant, currentProductVariant);
              SetupComplete setup = new SetupComplete(this, productVariant);
              /* Add the event to the event queue. */
              AddEvent(setupTime,setup);
              return;
          }
          ProductionCompleted productionCompleted = new ProductionCompleted(this, productionVariant);
          double procTime = getProcTime(productVariant);
          AddEvent(procTime, productionCompleted);
      }
}
```

The SetupCompleteEvent is triggered from the machine code of StartProductVariant. This event will set the current product variant (machine.SetCurrentProductVariant) to the product variant that the machine is about to start. Then the method StartProductVariant is executed once again and the job can be started in the machine. A ProductionCompletedEvent is added to the event list.

SetupCompleteEvent pseudo code

```
public class SetupComplete : public SimulationEvent
{
    private Machine machine;
    private ProductVariant productVariant;
    public SetupComplete(Machine machine, ProductVariant productVariant)
    {
        this.machine = machine;
        this.productVariant = productVariant;
    }
    public override void Execute(float elapsedSimulationTime)
    {
        machine.SetCurrentProductVariant(productVariant);
        machine.StartProductVariant(productVariant);
    }
}
```

Appendix I: Genetic Algorithm

In this Appendix, the example code of the Genetic Algorithm is presented. The Genetic Algorithm has been described in detail in Chapter 5. This pseudo code describes the procedure of the dispatching rule mutation operator.

Genetic algorithm pseudo code for the dispatching rule mutation operator

```
/* Mutate dispatching rules.
*  Scenario is the static(does not change between mutations) input data.
*  SimInput is the current schedule that we want to change.
*/
void DispatchingRuleMutation(Scenario scenario, SimInput mutant)
{
    /* Select a random production stage. */
    int i = ListUtility.RandomIndex(scenario.operationGroups);
    /* Make sure that are at least two allowed dispatching rule. */
    if (scenario.operationGroups[i].allowedDispatchingRules.Count > 1)
    {
        int newDr;
        do
        {
            /* Randomly select a new dispatching rule. */
            newDr = ListUtility.RandomElement(scenario.operationGroups[i].allowedDispatchingRules);
            /* Try again if we selected the same dispatching rule. */
        } while (newDr == mutant.dispatchingRules[i]);
        /* Set the new dispatching rule. */
        mutant.dispatchingRules[i] = newDr;
    }
}
```

Appendix J: List of Publications

The publications made during the research study are listed below. The papers with Andersson or Frantzén[1] are marked in bold.

Journal papers

Frantzén, M., Ng, A.H.C., Moore, P. (2011). "A simulation-based scheduling system for real-time optimization and decision making support", Robotics and Computer-Integrated Manufacturing, Volume 27, Issue 4, August 2011, pp. 696-705.

Dudas, C., **Frantzén, M.**, Ng, A.H.C. (2011). "A synergy of multi-objective optimization and data mining for the analysis of a flexible flow shop", Robotics and Computer-Integrated Manufacturing, Volume 27, Issue 4, August 2011, pp. 687-695.

Book chapters

Ng, A., Grimm, H., Lezama, T., Persson, A., **Andersson, M.** and Jägstam, M. (2008). "OPTIMISE: An Internet-Based Platform for Metamodel-Assisted Simulation Optimization", Recent Advances in Communication Systems and Electrical Engineering, X. Huang, Y-S. Chen and S-L. Ao (eds), Springer, pp. 281-296.

Conference papers

Frantzén, M., Ng, A.H.C., Moore, P. (2010). "A Scheduling system for real-time decision making support using simulation-based optimization", In Proceedings of the 20th International Conference on Flexible Automation and Intelligent Manufacturing, July 12-14, 2010, Oakland, USA. pp. 1012-1019.

Dudas, C., **Frantzén M.**, Ng, A.H.C., (2010). "Simulation-based Innovization for the Analysis of a Machining Line", In Proceedings of the 20th International Conference on Flexible Automation and Intelligent Manufacturing, July 12-14, 2010, Oakland, USA. pp. 991-998.

[1] The author changed his name from Andersson to Frantzén in fall 2008

Andersson, M., Ng, A., and Grimm, H. (2008). “Simulation Optimization for industrial scheduling using hybrid genetic representation”, In Proceedings of the 2008 Winter Simulation Conference, Miami, Florida, USA, 7-10 December 2008, pp. 2004-2011.

Andersson, M., Persson, A., Grimm, H., and Ng, A. (2007). “A Web-based simulation optimization system for industrial scheduling”, In Proceedings of the 2007 Winter Simulation Conference, Washington D.C., USA, 9-12 December 2007, pp. 1844-1852.

Andersson, M., Persson, A., Grimm, H., and Ng, A. (2007). “Simulation-based Scheduling using a Genetic Algorithm with Consideration to Robustness: A Real-world Case Study”, In Proceedings of the 17th International Conference on Flexible Automation and Intelligent Manufacturing, June 18-20, 2007, Philadelphia, USA, pp. 957-964.

Persson, A., Grimm, H., **Andersson, M.** and Ng, A. (2007). “Metamodel-Assisted Simulation-Based Optimization of a Real-World Manufacturing Problem”, In Proceedings of The 17th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM'07), Philadelphia, USA, June 18-20, pp. 950-956.

Persson, A., Grimm, H., Ng, A. and **Andersson, M.** (2007). “Metamodel-Assisted Simulation-Based Optimisation of Manufacturing Systems”, In Proceedings of 5th International Conference on Manufacturing Research (ICMR 2007), Leicester, UK, September 11-13, 2007, pp. 174-178.

Ng, A., Grimm, H., Lezama, T., Persson, A., **Andersson, M.** and Jägstam, M. (2007). “Web Services for Metamodel-Assisted Parallel Simulation Optimization”, In Proceedings of the IAENG International Conference on Internet Computing and Web Services (ICICWS'07), Hong Kong, 21-23 March 2007, pp. 879-885.

Aslam, T., **Andersson, M.**, Ng, A., and De Vin, L. (2006). ”Simulation-based optimisation for complex production systems: an industrial case study”, In Proceedings of International Manufacturing Conference 2006 (IMC23).

Conference short papers

Ng, A., Grimm, H., Persson, A., **Andersson, M.** and Jägstam, M. (2007). “A Platform for Metamodel-Assisted Parallel Simulation Optimisation using Soft Computing Techniques”, In Proceedings of SAIS 2007, The 24rd Annual Workshop of the Swedish Artificial Intelligence Society. Borås, Sweden, May 22-23, 2007, pp. 181-184.