Design and development of a real-time scheduling system in a sensorised job shop using cloud-based simulation with mobile device access

by

Stephan Snyman

Dissertation presented for the degree of Doctor of Philosophy in Engineering (Industrial Engineering) in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof JF Bekker

December 2019

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2019

Acknowledgements

Prof. James Bekker, for your knowledge and guidance throughout the project. I would also like to thank you for the continued support throughout the duration of the project.

The USMA group for the support over the past three years.

Friends and family for the encouragement and support.

Lastly, I would like to thank God for His strength and love, and without whom I would not have successfully completed this project.

Abstract

Traditional scheduling approaches used in manufacturing systems address scheduling problems before production commences, which poses possible problems when the system is interrupted by unexpected events. Managers must then react in a timely fashion, by developing a new or revised schedule to mitigate the effects of the interruptions on the productivity of the system. This can be done by a real-time scheduling system, which is used in conjunction with the actual manufacturing system. Technological advances, including cloud-based computing, the omnipresence of mobile devices, and the improved capabilities of sensor networks, have opened up the opportunity to design a real-time scheduling system, as well as create software architectures to support such a system.

The purpose of this research project is therefore to develop a prototype of a real-time simulation scheduling system, which will serve as a decision support tool for real-time rescheduling of machine steps in a job shop. The prototype incorporates a cloud-based information system for the storage of data and a cloud-based simulation scheduler that generates schedules. It also includes web pages for logging data changes and selecting a new schedule and sensors that keep track of the movement of jobs through the job shop.

The preliminary test results of the developed simulation scheduler suggest that metaheuristics should be considered to generate schedules, due to the metaheuristics outperforming the common dispatching rules. The model was then expanded from the single-objective to the multi-objective domain, which is a better representation of the real-world job shop environment. Several metaheuristics were adapted to solve the bi-objective job shop scheduling problem, after which comparison tests were conducted. The tests revealed that the NSGAII performed best of all the metaheuristics and it was selected for further implementation.

The final phase of this research project was to implement a newly developed ranking and selection procedure for discrete stochastic simulation problems, called *MMY*. The MMY procedure finds the minimum number of simulation

replications for each solution, while guaranteeing that the probability of correct selection of the best solutions exceeds a desired value. In this study, MMY finds the best simulated schedules while the probability of correct selection is guaranteed.

Opsomming

Tradisionele skeduleringsbenaderings wat gebruik word in vervaardigingstelsels spreek skeduleringsprobleme, voordat produksie begin, aan. Dit kan moontlike probleme veroorsaak wanneer die stelsel onderbreek word deur onverwagte gebeurtenisse. Bestuurders moet dan betyds reageer deur 'n nuwe of hersiene skedule te ontwikkel om sodoende die uitwerking van die onderbrekings op die produktiwiteit van die stelsel te versag. Dit kan gedoen word deur 'n reële tyd skeduleringstelsel, wat gebruik word in samewerking met die werklike vervaardigingstelsel. Baie tegnologiese vooruitgange, insluitende wolkgebaseerde rekenaars, die alomteenwoordigheid van mobiele toestelle en die verbeterde vermoëns van sensornetwerke, het die geleentheid gebied om die reële tyd skeduleringstelsel te ontwerp, asook die geleentheid vir die skep van sagtewareargitekture om hierdie skeduleringstelsels te ondersteun.

Die doel van hierdie navorsingsprojek is dus om 'n prototipe van 'n reële tyd simulasie skeduleringstelsel te ontwikkel, wat sal dien as 'n besluit hulpmiddel vir die reële tyd herskedulering van masjienstappe in 'n werkswinkel. Die prototipe bevat 'n wolkgebaseerde inligtingstelsel vir die stoor van data asook 'n wolkgebaseerde simulasieskeduleerder wat skedules genereer. Dit sluit ook webbladsye in om dataveranderinge aan te teken en sensors wat die beweging van werk deur die werkswinkel volg.

Die voorlopige toetsuitslae van die ontwikkelde simulasie skeduleerder stel voor dat metaheuristieke oorweeg moet word om skedules te genereer, weens die metaheuristieke wat beter as die algemene versendingsreëls presteer. Die model kan dan uitgebrei word van die enkel-doelwit tot die multi-doelwit domein, wat 'n beter voorstelling van die werklike werkswinkelomgewing is. Verskeie metaheuristieke is aangepas om die bi-doelwit werkswinkel skeduleringsprobleem op te los, waarna vergelykingstoetse uitgevoer kon word. Die uitkoms van die toetse het getoon dat die NSGAII die beste presteer het van al die metaheuristieke en daarom is dit gekies vir verdere implementering.

Die finale fase van hierdie navorsingsprojek was om 'n nuut ontwikkelde rangskik-en-kies prosedure vir diskrete stogastiese simulasieprobleme, genaamd MMY, te implementeer. Die MMY prosedure vind die minimum aantal simulasie replikasies vir elke oplossing, terwyl dit die waarskynlikheid van korrekte keuse

waarborg. In hierdie studie vind die MMY prosedure die beste gesimuleerde skedules terwyl die waarskynlikheid van korrekte keuse waarborg word.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ABS	Agent based systems
ACO	Ant colony optimisation
COP	Constraint Optimisation Problem
CSP	Constraint Satisfaction Problem
FCFS	First-come-first-served
FL	Fuzzy logic
GA	Genetic algorithm
GCP	Google Cloud Platform
IoT	Internet of Things
JSP	Job shop scheduling problems
KPI	Key performance indicator
MBIPP	Mixed binary integer programming problems
MIJF	Most-important-job-first
NN	Neural networks
OPM	Object Process Methodology
PSO	Particle swarm optimisation
SJSP	Stochastic job-shop scheduling problems
TS	Tabu search
TSP	Travelling Salesman Problem
VNS	Variable neighbourhood search
VPN	Virtual Private Network

WIP	Work in progress

Symbols

δ_b^*	The indifference-zone value for objective b
$\mathbb{Q}_{\mathbb{T}}$	The set of all possible true relaxed Pareto sets
$\overline{L}$	Average lateness of jobs
$\overline{T}$	Average tardiness of jobs
$\overline{W}$	Average queuing/waiting time of jobs in the job-shop
$\xi C_{\max}$	The stochastic makespan
ξC_{ij}	The stochastic completion time of operation $o_{i,j}$
ξp_{ijk}	The stochastic processing time of operation $o_{i,j}$ on machine k, subjected to independent normal distributions
ξs_{ij}	The stochastic starting time of operation $o_{i,j}$
A	Set of precedence constraints $(i,j) \rightarrow (i,h)$
$C_{\max}$	Maximum makespan
C_{ij}	Completion time of operation $o_{i,j}$
d_i	Due date of job i
H_0	A large positive number used for a penalty factor
i	The job index
J	Set of jobs $\{J_1, J_2, \ldots, J_n\}$
j	The operation index
k	The machine index
L_j	Lateness of job i
M	Set of machines $\{M_1, M_2, \ldots, M_m\}$
m	Total number of machines

n	Total number of jobs
n_0	The number of simulation replications at the first stage
O	The set of operations
$o_{i,j}$	Operation j of job i
P^*	The minimum required value for $P(\text{CS})$
p_{ijk}	Processing time of operation $o_{i,j}$ on machine k
Q	The true Pareto set based on μ_{ib} ($i \in S$ and $b \in B$)
Q_c	The true non-Pareto set based on μ_{ib} ($i \in S$ and $b \in B$)
Q_R	The true relaxed Pareto set
Q_{IZ}	The true Pareto set with IZ
Q_{IZ}^c	The true non-Pareto set with IZ
r_i	Release time of job i
S_p	The observed Pareto set based on $\overline{X}_{ib}(N_i)$ ($i \in S$ and $b \in B$)
S_p^c	The observed non-Pareto set based on $\overline{X}_{ib}(N_i)$ ($i \in S$ and $b \in B$)
s_{ij}	Starting time of operation $o_{i,j}$
S_{IZ}	The observed Pareto set with IZ
S_{IZ}^c	The observed non-Pareto set with IZ
T_i	Tardiness of job i
T_{count}	Total number of tardy jobs
W_i	Queuing/waiting time of job i

Chapter 1

Introduction

This chapter provides a description of the project background, a problem statement, and the objectives that need to be accomplished to successfully complete the project. The project scope and proposed research methodology will also be discussed.

1.1 Project background

Manufacturing is described by Groover (2013), as "the transformation of materials into items of greater value by means of one or more processing and/or assembly operations". The processes to accomplish the transformation involve a combination of machinery, tools, power, and manual labour. From Groover's definition of manufacturing, the process can be considered a complex engineering endeavour, where the coordination of people, material, equipment, and information to accomplish a manufacturing goal demands considerable time and effort.

According to Mitsuishi *et al.* (2008), the manufacturing system life cycle can be divided into the design, planning, implementation, operation and termination phases. The coordination challenges that are faced in the design and implementation phases can be overcome by careful planning; however such challenges continue to persist over the operational phase. For many manufacturing systems, scheduling is such a challenge, which can be attributed to the complex, dynamic and stochastic environments exhibited by these systems.

Traditional approaches used to address the scheduling problems involve creating and evaluating schedules prior to commencing production. However, according to Suwa and Sandoh (2012), uncertainties that are not expected or taken into account at the planning phase will possibly cause delays on these schedules. Common uncertainties that occur in a manufacturing system include machine operator absence, material shortage, machine failure and new orders. In such scenarios, the manager must react quickly by selecting a new or revised schedule to ensure that production continues, while still maintaining the required performance level. *On-line scheduling*, which is a real-time decision-making process, attempts to address the shortcomings of the traditional approaches, by performing scheduling concurrently with the production process. Computer simulation is often used to assist with scheduling, especially of complex, discrete, dynamic, stochastic processes and this is generally known as *simulation-based real-time scheduling*.

Stankovic *et al.* (2012) state that "typically, a real-time system consists of a controlling system and a controlled system". In an automated factory, the controlled system is the

factory floor with its resources and equipment; while the controlling system is the computer and human interfaces that manage and coordinate the activities on the factory floor. The controlling system interacts with its environment based on the information available about the environment from various sensors attached to it (Stankovic *et al.*, 2012). It is essential that the state of the environment, as perceived by the controlling system, be consistent with the actual state of the environment. Many technological advances, including cloud-based computing, the omnipresence of mobile devices, and the improved capabilities of sensor networks, have offered the opportunity of designing the *real-time scheduling system* as described by Stankovic *et al.* (2012), as well as the opportunity for the creation of software architectures to support these *real-time scheduling systems*.

1.2 Problem statement

Traditional scheduling approaches used in manufacturing systems address scheduling problems before production commences, which poses possible problems when the system is interrupted by unexpected events. Managers must then react in a timely fashion, by developing a new or revised schedule to mitigate the effects of the interruptions on the productivity of the system. This can be done by a *real-time scheduling system*, that is used in conjunction with the operational manufacturing system.

The purpose of this research is therefore to design and develop a **prototype** of a **real-time cloud-based simulation scheduling system** of a **sensorised job shop** with **mobile device access**, to serve as a **decision support tool** for **rescheduling** machine steps in a *job shop*. The machines in the *job shop* are monitored, and the progress of orders can be captured in real time as the machine steps of each order are executed.

1.3 Objectives

To achieve the stated purpose of the research, the following objectives were set:

1. Development of an operational simulation scheduler which will act as a digital twin of the real-world environment and that generates new production schedules for the job shop according to common dispatching rules.
2. Creating web pages for the capturing and displaying of data.
3. Establish a virtual job shop that captures the movement of jobs through the shop in real time.

4. Configure a cloud-based server to host the web pages, simulation scheduler and information system.
5. Develop a simulation scheduler using a single objective for scheduling, then expand it to the multi-objective optimisation domain.
6. Implement a newly developed ranking and selection procedure, called Procedure MMY to select the best simulated schedules while the probability of correct selecting is guaranteed.

1.4 Contributions

The contributions of this research project can be described as:

- The research documents a complete high-level architecture, that can be used by future researchers to replicate the system, and to make improvements.
- The research documents the integration of typical modern technologies into a single operational system.
- The research documents the use of uncommon scheduling objectives that are strongly conflicting.
- The research documents the first implementation of Procedure MMY (Yoon, 2018) to a simulated job shop scheduling problem.

Furthermore, the author will be designing a real-time stochastic scheduling system for a job shop environment. This system will include (i) a simulation model in the cloud, to alter the schedule; (ii) an information system that will provide relevant data to the simulation model; and (iii) functioning sensors that are able to capture operation status changes in the information system. The overall system will be accessible from a mobile device.

1.5 Scope

The study assumes a *job shop* that processes orders of different kinds, with few to many operations per job. Orders are most of the time unique, *i.e.* a few instances of a one-of-a-kind product are manufactured by the job shop. The data required for the thesis will be gathered using the developed sensors and web pages, after which the data will be stored in the information system. A fictitious job shop will be created where machines may fail and operators may be absent, which will dictate the availability/unavailability of resources. The size of the job shop is limited to ten machines and no more than two of a type. There

may be any number of jobs entering the system, but the required operations per job are limited to machine capabilities. Also, the required number of physical products per job is limited to 10. Processing times will be dictated by fixed distributions.

1.6 Research methodology

This section describes the proposed research methodology in order to fulfil the stated objectives and desired purpose of the thesis. As stated in Section 1.2, the purpose of the research is to design and develop a prototype scheduling system; therefore, the research can be classified as a design project. The research methodology that will be used for this research project is the *Engineering Design Process*, which is graphically illustrated in Figure 1.1.

The first step of the methodology is **Ask**, where the need and constraints of the research project are identified. The need and constraints of the research project were defined in Sections 1.1 to 1.5, which document the problem background, problem statement, objectives, contributions and the scope of the research project. The next step is **Research**, where the defined problem must be researched. This step will be addressed in Chapter 2, by conducting an extensive literature review. The review will be conducted to gain knowledge

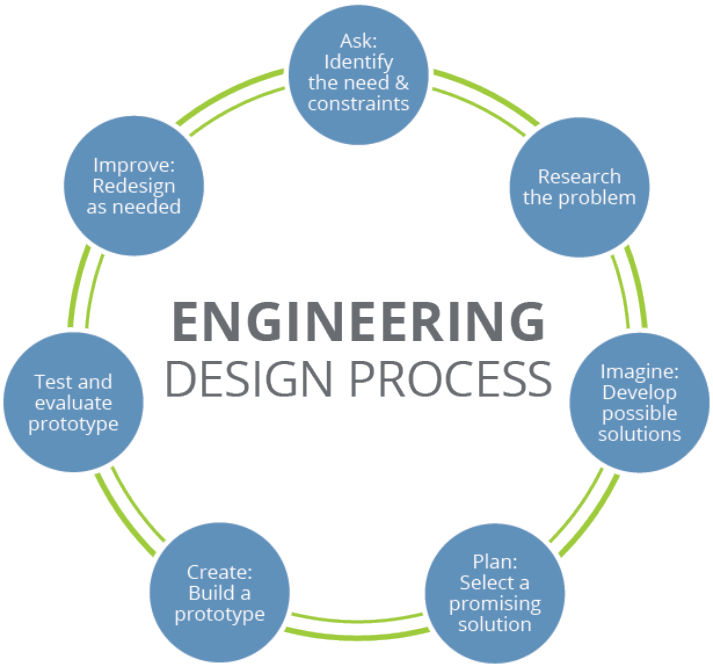


Figure 1.1: Engineering Design Process steps (Teach Engineering, 2016).

of *simulation-based real-time scheduling systems* as well as *scheduling of job shops*. The review will focus on defining *real-time scheduling systems*, while also providing characteristics of such systems. Literature regarding *job shops* will be used to identify performance indicators and techniques used in the rescheduling process. Methods and mathematical models for solving the job shop scheduling problem and the use of simulation with these methods will also be investigated.

The third step of the methodology is **Imagine**, where possible solutions are developed. This step will be addressed in Chapters 3 and 4, where the required functionality of the possible solutions are defined and different functional and quality scenarios are identified. Business goals and drivers are also identified and documented which serve as motivation for commercial implementation. The fourth step is **Plan**, where a promising solution is selected. This step will be addressed in Chapter 4, where an architecture of the promising solution is defined. The architecture is constructed with the Object Process Methodology that generates a graphical as well as semantic representation of the architecture. The system architecture defines the overall systems structure-behaviour combination, which enables it to attain its function while embodying the architect's concept.

The fifth step of the methodology is **Create**, where a prototype of the promising solution is developed. This step will be addressed in Chapter 5, which documents the development of all the different components of the proposed system, as well as describing how each component will be integrated.

The sixth step of the methodology is **Test**, where the prototype is analysed and evaluated. This step will be addressed and the results will be provided in Appendix B and C, where verification and operational tests will be conducted. The verification tests will be used to determine whether the scheduler generates the correct schedules according to each dispatching rule; while the operational test will be used to determine whether the scheduler will be able to adapt the schedules when a disruption occurs. The last step of the methodology is **Improve**, where the prototype will go through a redesign phase if needed. This step will be addressed in Chapters 6 and 7, where shortcomings of the prototype will be identified and improvements introduced. One of the improvements that will be addressed is the expansion of the simulation scheduler from the single-objective to multi-objective optimisation domain, which will provide a realistic and more useful representation of a real-world job shop environment.

Together with the Engineering Design Process research methodology, there are eight technical processes that support design methodologies, as shown in Table 1.1. The different technical processes, their purpose and where each process is addressed, are illustrated.

Table 1.1: Technical processes required when conducting a design project (Evbuomwan *et al.*, 1996).

Technical process	**Purpose**	**Chapter**
Stakeholder requirements definition	Define the requirements for a system that can provide the services needed by users and other stakeholders in a defined environment	Chapter 4
Requirements analysis	Transform the stakeholder, requirement-driven view of desired services into a technical view of a required product that could deliver those services	Chapter 4
Architectural design	Synthesise a solution that satisfies system requirements	Chapter 4
Implementation	Realise the system elements	Chapter 5
Integration	Assemble a system that is consistent with the architectural design	Chapter 5
Verification	Confirm that the specified design requirements are fulfilled by the system	Chapter 6 Chapter 7 Appendix B
Transition	Establish a capability to provide services specified by stakeholders' requirements in the operational environment	Appendix C
Validation	Provide objective evidence that the services provided by a system when in use comply with stakeholders' requirements, achieving its intended use in its intended operational environment	N/A

1.7 Deliverables envisaged

At the completion of the study, the following deliverables will be available:

- A simulation model of a fictitious *job shop*, accessible only through the web.
- An information system used for storing the progress of jobs in real time.
- Functioning sensors used to capture operation status changes.

- A demonstrator integrating the above.

1.8 Structure of the document

A brief summary of each of the chapters in the thesis is provided next.

Chapter 1 *Introduction*
This chapter provides the background to the thesis. Included in the chapter is the project background, problem statement, project scope and the proposed research methodology to achieve the stated objectives and successfully complete the thesis.

Chapter 2 *Literature review*
This chapter starts by investigating *job shop* scheduling, to identify appropriate scheduling methods as well as performance indicators for *job shops*. Following this, the mathematical models for minimising the performance indicators, as well as the methods for solving JSPs, will be discussed. Thereafter, stochastic scheduling will be introduced and characteristics of such scheduling problems will be defined. Finally, real-time scheduling systems and the use of simulation as a *real-time scheduling* method will be investigated along with the basic principles.

Chapter 3 *Architectural Design*
This chapter defines the architectural design process and concepts that are required for the successful development of an architecture. The chapter also makes reference to the framework required for the level of detail that should be considered by the developer in the development process.

Chapter 4 *Implementation of architectural design*
This chapter provides the implementation of the architectural design literature, where the appropriate stakeholders, viewpoints and perspective are selected. Furthermore, business goals and drivers, as well as the scope of the proposed system, will be defined. Finally, a software development method will be chosen to guide the development of the proposed system.

Chapter 5 *Development of proposed system*
This chapter documents the development process of the proposed system. The components that were developed for the system include an information system where all required data

will be stored, the web pages used for logging data changes, a simulation model that mimics the behaviour of a job shop, and finally, sensors for logging real-time data.

Chapter 6 *Incorporation of metaheuristics in simulation scheduler*
This chapter describes the incorporation of metaheuristics in the simulation scheduler. The metaheuristics that were implemented are *simulated annealing* and a variation of the *2-opt* algorithm. Detailed analysis and comparison of metaheuristics and dispatching rules are also provided.

Chapter 7 *Formulation and testing of bi-objective simulation scheduler*
This chapter will document the expansion of the simulation scheduler from the single-objective to the multi-objective optimisation domain. The chapter will start off with the selection and discussion of the objectives that will be used in the scheduler. Thereafter, the metaheuristics that are going to be incorporated into the simulation scheduler will be discussed. Comparison tests of the performance of each metaheuristic will then follow. Finally, the chapter will document the implementation of a newly developed ranking and selection technique, *MMY*, to determine the approximate Pareto set and guarantee probability of correct selection of the best simulated schedules.

Chapter 8 *Project conclusion*
This chapter describes the conclusion of the research project. It will contain a summary of the work that was covered in the project, as well as the research contributions of this project. Future research opportunities will also be discussed. This chapter will end off with a self-assessment and reflection of the project.

1.9 Chapter summary

This chapter serves as an introduction to the thesis and contains a project background, problem statement, project scope and the proposed research methodology to successfully complete the project. The structure of the thesis is also provided. The succeeding chapter will contain a literature review on *real-time scheduling systems*, as well as scheduling of jobs in the *job shop* environment.

Chapter 2

Literature review

This chapter starts by investigating *job shop* scheduling, to identify appropriate scheduling methods as well as performance indicators for *job shops*. Following this, the mathematical models for minimising the performance indicators, as well as the methods for solving JSPs, will be discussed. Thereafter, stochastic scheduling will be introduced and characteristics of such scheduling problems will be defined. Finally, real-time scheduling systems and the use of simulation as a *real-time scheduling* method will be investigated along with the basic principles.

2.1 Job shop scheduling

According to Murthy (2005), the organisation of manufacturing systems, as well as the planning and control of production greatly depends on the type of production. Aspects of production management in fulfilling specific requirements of a plant and the management approach to problems of inventory, machine selection, machine setting, tooling, routing, scheduling, loading, follow-up and general control will differ depending on the type of production system (Murthy, 2005).

Miltenburg (2005), Papadopoulos *et al.* (2009) and Murthy (2005) refer to three types of production systems:

1 **Job shop or job production:** In this system, products are manufactured to meet the requirements of a specific order. The manufacturing of the product will take place as per the specifications given by the customer.

2 **Batch production:** In this system, a number of identical products are manufactured either to meet the specific order or to satisfy the demand. When the production of plant and equipment is terminated, the plant and equipment can be used for producing similar products.

3 **Continuous production:** This system is the specialised manufacture of identical products on which machinery and equipment are fully engaged. Continuous production is normally associated with large quantities and high rate of demand.

Figure 2.1 illustrates the characteristics of intermittent (job and batch production) and continuous (mass and flow production) production systems. As observed in Figure 2.1, job production has major differences in product variety but very low product volumes; batch

production has large product variety but low product volumes; and continuous production has little product variation but high product volumes.

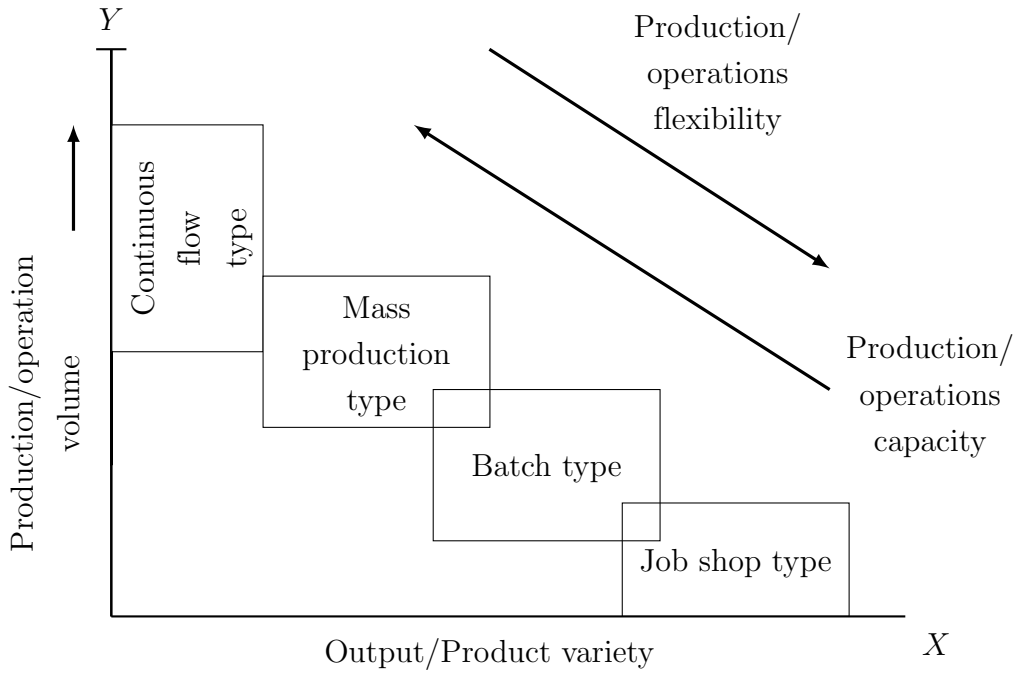


Figure 2.1: Characteristics of production systems (Murthy, 2005).

2.1.1 Identifying dispatching rules and performance indicators

According to Wisner (2016), scheduling jobs in a process-focused facility or *job shop* involves a number of factors:

- If more work is accepted per day than the organisation can complete per day, then the overall work in progress (WIP) inventories will increase, causing shop congestion, an erosion of the firm's output rate, and a lengthening of job completion times.
- If completion times or dates are promised to customers, then estimates of lead times for each job must be determined, and jobs must be started far enough in advance to complete the job by the promised date.
- Facilities can finish more jobs per period and satisfy more customers if they work on the shortest jobs first. However, longer jobs will ultimately be completed late or behind schedule.
- More highly valued customers may require earlier completion dates which will then be given processing priority in the shop. This will in turn make it more difficult to estimate accurate completion dates for other jobs.

- Interruptions such as machine breakdowns, employee absence, poor raw-material quality, and processing errors, can cause unforeseen delays in processing.

The job scheduling process therefore includes machine-level controlling measures, such as sequencing, dispatching rules and performance indicators. The dispatch rules guide the production sequence of the jobs within the shop, and ensure that operators know what job to process next when the machine becomes available. Dispatch rules, as mentioned by Dominic *et al.* (2004) and Wisner (2016), include:

1) shortest process time,
2) earliest due date,
3) minimum slack time per operation,
4) critical ratio,
5) first-come-first-served (FCFS), and
6) most-important-job-first (MIJF).

Table 2.1 provides a description for each of the dispatch rules.

Table 2.1: Description of each dispatch rule (Dominic *et al.*, 2004; Wisner, 2016).

Dispatch rule	**Description**
Shortest process time	The job with the shortest process time is processed first.
Earliest due date	The job with the earliest due date is processed first.
Minimum slack time	The job with the minimum slack time per remaining operation is processed first.
Critical ratio	The job with the smallest critical ratio is processed first. The critical ratio can be calculated using $\frac{Time\ until\ due\ date}{Remaining\ process\ time}$.
FCFS	The job arriving first at a workstation is processed first.
MIJF	Jobs are prioritised based on the importance of the customer.

According to Pezzella *et al.* (2008), one of the most difficult problems in the area of job shop scheduling problems (JSP), where a set of jobs must be processed on a set of machines, is the decision as to how to sequence the operations on the machines. This

decision is dictated by the job shop performance indicators that need to be optimised. Table 2.2 provides common job shop performance indicators.

Table 2.2: Common job shop performance indicators (Dominic *et al.*, 2004; Wisner, 2016).

Performance indicator	**Description**
Average flow time	The flow time begins when a job arrives at the shop, and ends when it leaves. This flow time is averaged over a number of jobs.
Average queue time	The queue time is the total flow time minus the process time of the job. The queue time is then averaged over a number of jobs.
Average job lateness	Lateness is the difference between the completion date and the due date. The lateness is averaged over a number of jobs.
Average job tardiness	Tardiness is the amount of time a job finishes beyond the due date. Tardiness is averaged over a number of jobs.
Makespan	Makespan is the total elapsed time to complete a number of jobs.

2.2 Mathematical models of job shop scheduling

This section serves to discuss mathematical models for the minimisation of job shop performance indicators, as seen in Table 2.2. The JSP is referred to in literature as an NP-hard problem (Ganesh, 2012; Pardalos, 2013; Xhafa and Abraham, 2008), when more than three resources/machines are present. French (1982); Gen *et al.* (2009); Jensen (2001); Kuhpfahl (2015); Zhang *et al.* (2008) define the environment of the JSP as follows:

- There is a set J of n jobs $\{J_1, J_2, \ldots, J_n\}$ that needs to be processed on a set M of m machines $\{M_1, M_2, \ldots, M_m\}$.
- Each job i consists of a finite and predetermined sequence of j operations $\overline{O}_i = \{o_{i,1}, o_{i,2}, \ldots, o_{i,j_i}\}$, where the operation order is fixed.
- An operation may only be assigned to an available machine forming part of the set M.
- A machine can process at most one operation at a time, and no preemption can take place.

- Each operation $o_{i,j}$ has a fixed processing time $p_{i,j}$.
- The aim is to find a schedule for processing these n jobs on the m machines.

According to Sadeh (1991), job shop scheduling must be considered as a *Constraint Satisfaction Problem* (CSP) or a *Constraint Optimisation Problem* (COP). The constraints applicable in the JSP are the *precedence constraints* specified by the process routings, and *capacity constraints* that prevent resources from being allocated to more operations than they can process at one time.

Figure 2.2 illustrates a JSP where there are four jobs on five machines. Each node in the figure represents an operation and is labelled with the name of the operation ($o_{i,j}$), where i is the job index and j is the operation index; the k-th resource/machine required to process the operation is indicated by M_k. The processing time (p_{ijk}) of each operation is shown as a number in each node. The arrows in the figure represent the precedence constraints and the broken lines represent the capacity constraints. In this example it is assumed that each machine can only process one operation at a time, hence the use of a capacity constraint. Therefore, if more than one operation is competing for a machine, all but one must wait, as they cannot be processed at the same time.

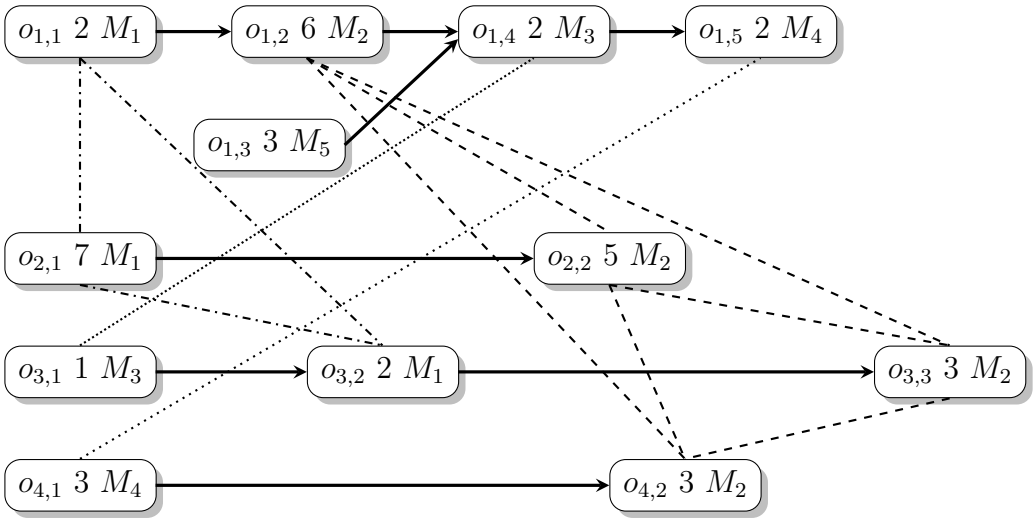


Figure 2.2: A sample job shop problem (Sadeh, 1991).

An example of the precedence constraints in Figure 2.2 is the precedence between $o_{3,1}$, $o_{3,2}$ and $o_{3,3}$. The precedence between these operations ensure that $o_{3,1}$ is processed before $o_{3,2}$ and $o_{3,3}$. Furthermore, operations $o_{1,2}$, $o_{2,2}$, $o_{3,3}$ and $o_{4,2}$, all need to be processed on machine M_2, hence the capacity constraint.

An optimal schedule for a job shop can be generated by solving the mathematical representation of the job shop. The purpose of the mathematical model of the job shop

is to minimise the performance indicators (as described in Table 2.2) in order to obtain an optimal schedule. This is achieved by determining best values for job shop scheduling decision variables, which include $Y_{iji'j'}$ that generate the sequence between operations $o_{i,j}$ and $o_{i',j'}$, as well as X_{ijk} for machine selection for operation $o_{i,j}$.

Mathematical models for the minimisation of the performance indicators are discussed in this section. The notation used throughout the section and thesis to describe the mathematical models are:

d_i	Due date of job i.
i	The job index.
j	The operation index.
k	The machine index.
m	Total number of machines.
n	Total number of jobs.
$o_{i,j}$	Operation j of job i.
p_{ijk}	Processing time of operation $o_{i,j}$ on machine k.
r_i	Release time of job i.
s_{ij}	Starting time of operation $o_{i,j}$.
A	Set of precedence constraints $(i,j) \rightarrow (i,h)$.
C_{ij}	Completion time of operation $o_{i,j}$.
$C_{\max}$	Maximum makespan.
H_0	A large positive number.
J	Set of jobs $\{J_1, J_2, \ldots, J_n\}$.
L_i	Lateness of job i.
$\overline{L}$	Average lateness of jobs.
M	Set of machines $\{M_1, M_2, \ldots, M_m\}$.
O	The set of operations.
T_i	Tardiness of job i.
$\overline{T}$	Average tardiness of jobs.
T_{count}	Total number of tardy jobs.
W_i	Queuing/waiting time of job i.
$\overline{W}$	Average queuing/waiting time of jobs in the job shop.

$$X_{ijk} = \begin{cases} 1, & \text{if operation } o_{i,j} \text{ is processed on machine } k \\ 0, & \text{otherwise.} \end{cases}$$

$$Y_{iji'j'k} = \begin{cases} 1, & \text{if operation } o_{i,j} \text{ precedes operation } o_{i',j'} \text{ on machine } k \\ 0, & \text{otherwise.} \end{cases}$$

Average flow time

Nasr and Elsayed (1990) introduced a model to minimise the average flow time in a job shop. This model can be structured as

$$\text{Minimise} \sum_i \frac{C_{ij}}{n}$$

subject to

$$\sum_{k\,=\,1}^{m} X_{ijk} \;=\; 1, \quad \forall\, i = 1, \ldots, n, \tag{2.1}$$

$$C_{i'j'} - C_{ij} + H_0(1 - Y_{iji'j'k}) + H_0(1 - X_{ijk}) + H_0(1 - X_{i'j'k}) \geq p_{i'j'k}$$
$$\forall\, i, i' = 1, \ldots, n, \; \textit{and}\; k = 1, \ldots, m, \tag{2.2}$$

$$C_{ij} - C_{i'j'} + H_0 Y_{iji'j'k} + H_0(1 - X_{ijk}) + H_0(1 - X_{i'j'k}) \geq p_{ijk}$$
$$\forall\, i, i' = 1, \ldots, n, \; \textit{and}\; k = 1, \ldots, m, \tag{2.3}$$

$$s_{ih} \;-\; s_{ij} \geq \sum_k (X_{ijk} p_{ijk}), \quad \text{for all } (i,h) \to (i,j) \in A,$$
$$k = 1, \ldots, m, \tag{2.4}$$

where $C_{ij},\; p_{ijk} \geq 0$, n is the number of jobs over which the average is determined, and $X_{ijk},\; Y_{iji'j'k} = 0$ or 1. Constraint (2.1) ensures that only one machine is assigned to each operation, while constraints (2.2) and (2.3) ensure that a machine cannot simultaneously process more than one job at any time. Constraint (2.4) ensures the precedence given to the operations of each job is not violated.

The rescheduling process when minimising the *average flow time* will now be discussed using an example from a study by Satake *et al.* (1999). The example makes use of data presented in Table 2.3. The table describes the allocation of the operations of each job to a specific machine. There are three operations per job, which can be assigned to three different machines.

Table 2.3: Sample 3-job/3-machine job shop problem (reproduced from Satake *et al.* (1999)).

	Operation		
Job	1	2	3
1	1	2	3
2	1	3	2
3	2	1	3

The example will be described by minimising the average flow time over jobs one (J_1) and three (J_3). A non-optimal schedule can be seen in Figure 2.3. The flow time of J_1 and J_3 are 19 and 22 time units respectively, which results in an average flow time of 20.5 time units.

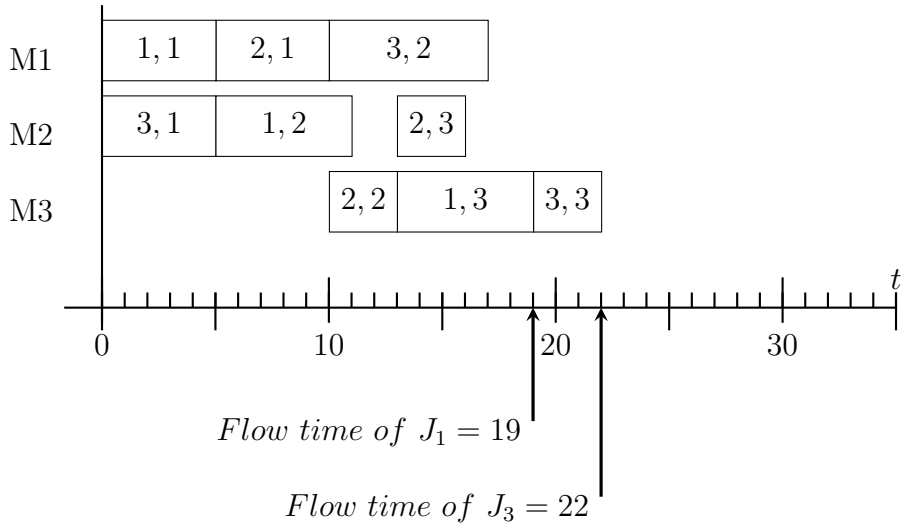


Figure 2.3: An example of a non-optimal schedule for the minimisation of average flow time

To decrease the average flow time of these two jobs, both jobs must have higher priority than job two (J_2), and therefore be scheduled prior to J_2. The sequence of operations on each machine can therefore change from that shown in Figure 2.3 to the sequence seen in Figure 2.4. This will ensure that both J_1 and J_3 will be processed prior to J_2, and therefore minimise the waiting time of operations from J_1 and J_3. In Figure 2.4, it can be seen that the flow time of J_1 and J_3 changed to 21 and 15 time units, respectively. The flow time of J_1 may have increased, but the average flow time decreased from 20.5 time units to 18 time units. Therefore, Figure 2.4 shows an example of a better average flow time schedule.

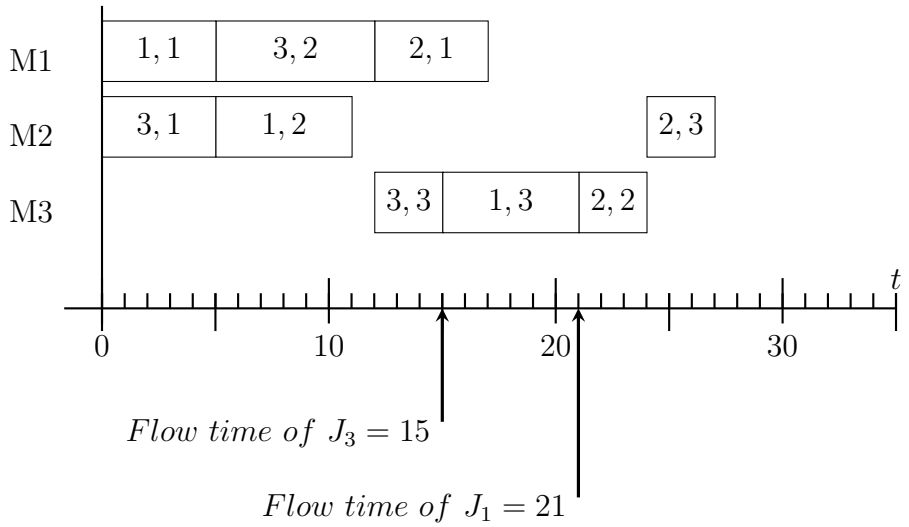


Figure 2.4: An example of an improved average flow time schedule

Average queue time

Wisner (2016) defines the calculation of the average queue time, also known as the average waiting time, as

$$W_i = \sum_j (C_{ij} - \sum_k (p_{ijk})), \quad i = 1, \ldots, n, \tag{2.5}$$

$$\begin{aligned} \text{Minimise } \overline{W} &= \frac{\sum_{i=1}^{n} W_j}{n} \\ &= \frac{\sum_{i=1}^{m} \sum_j (C_{ij} - \sum_k (p_{ijk}))}{n}, \end{aligned} \tag{2.6}$$

subject to

$$s_{ih} - s_{ij} \geq \sum_k (X_{ijk} p_{ijk}), \quad \text{for all } (i,h) \rightarrow (i,j) \in A, \; k = 1, \ldots, m, \tag{2.7}$$

where $\overline{W}$ represents the average queueing time, and n represents the number of jobs over which the average queue time is calculated. (2.5) calculates the waiting time experienced by each job, which is then averaged in (2.6) to calculate the average waiting time of the jobs. Constraint (2.7) ensures the precedence given to the operations of each job is not violated.

The rescheduling process when minimising the *average queue time* will now be discussed with reference to the example described in Table 2.3. If the objective of the rescheduling is to minimise the average queue time of J_1 and J_2, the rescheduling process must start by

identifying the waiting time of both jobs. An example of a non-optimal schedule regarding the average waiting time of J_1 and J_2 is illustrated in Figure 2.5.

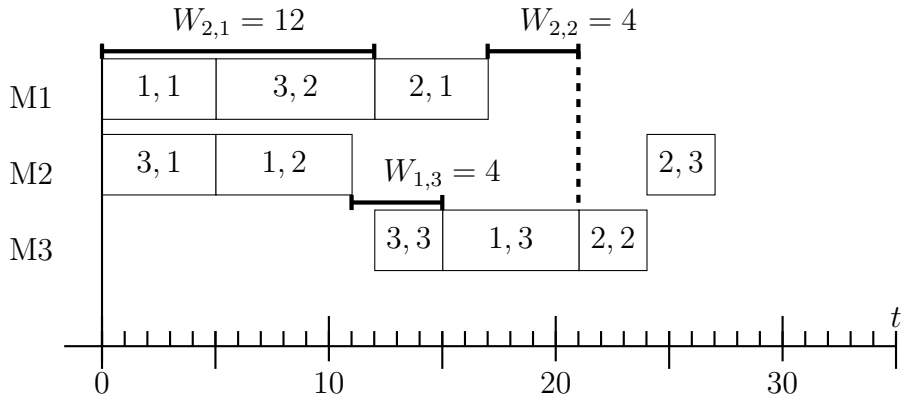


Figure 2.5: An example of a non-optimal schedule for the minimisation of average queuing time

In Figure 2.5, it can be seen that the queuing time of J_1 and J_2 are 4 and 16 time units, respectively. This results in an average queuing time of 10 time units. To decrease the average queuing time, the schedule must be altered by providing J_1 and J_3 with higher priorities than J_3. This will ensure that J_1 and J_2 are processed prior to J_3 and ultimately decrease the queuing time of J_1 and J_2 caused by J_3.

With the priorities of both J_1 and J_2 set higher than J_3, the sequence of each machine will change to the sequence as seen in Figure 2.6. In the figure, it can be seen that all the operations of J_1 and J_2 will be processed prior to the operations of J_3, except when the operations of J_3 are the only operations available for processing. As seen in Figure 2.6, the new schedule decreased the waiting time of J_1 and J_2 to 3 and 5 time units, respectively.

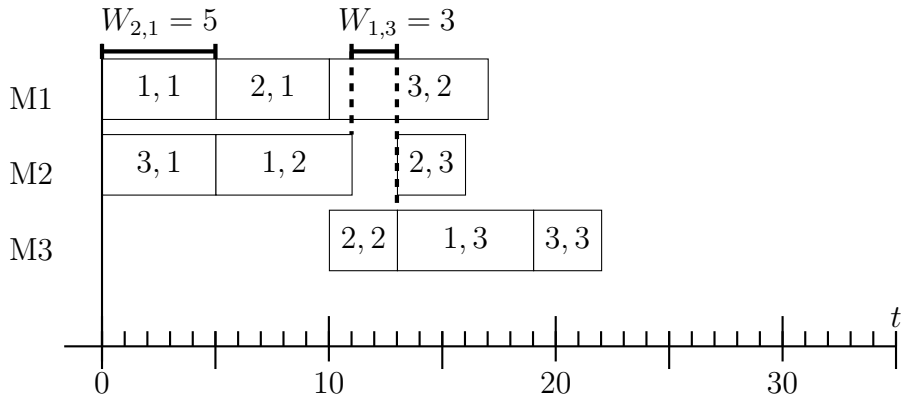


Figure 2.6: An example of an improved average queuing time schedule

This then equates to an average waiting time of 4 time units, which has decreased from the previous schedule. Therefore, Figure 2.6 illustrates an example of a better schedule with regard to average queuing time of J_1 and J_2.

Average job tardiness

Esposito (2005) introduces a mathematical model to minimise the average tardiness of jobs in a job shop, as well as a model to minimise the total number of tardy jobs in the job shop. The model for minimising the average tardiness of jobs within the job shop can be structured as

$$\text{Minimise} \quad \overline{T} = \frac{1}{n}\sum_{i=1}^{n} T_i, \quad i = 1, \ldots, n, \tag{2.8}$$

subject to

$$s_{ih} - s_{ij} \geq \sum_{k}(X_{ijk}p_{ijk}), \qquad \text{for all } (i,h) \rightarrow (i,j) \in A, \; k = 1, \ldots, m, \tag{2.9}$$

where

$$T_i = \max\{0, \sum_{j}(C_{ij}) - d_i\}, \quad i = 1, \ldots, n. \tag{2.10}$$

The model for minimising the total number of tardy jobs within the job shop is then

$$\text{Minimise} \quad T_{\text{count}} = \sum_{i=1}^{n} u_i, \tag{2.11}$$

subject to

$$s_{ih} - s_{ij} \geq \sum_{k}(X_{ijk}p_{ijk}), \qquad \text{for all } (i,h) \rightarrow (i,j) \in A, \; k = 1, \ldots, m, \tag{2.12}$$

where

$$u_j = \begin{cases} 1, & \text{if } T_i \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

(2.8) is used to determine and minimise the average tardiness of jobs, while (2.10) is used to determine the tardiness of each. (2.10) also ensures that the tardiness will never be negative. (2.11) is used to determine the total number of tardy jobs due to the sequence of the schedule. Constraints (2.9) and (2.12) ensure the precedence given to the operations of each job is not violated.

The rescheduling process when minimising the average job tardiness will now be discussed with reference to the example described in Table 2.3. To describe the rescheduling process, due dates for each job are required. Therefore, for the explanation, due date for J_1, J_2 and J_3 were chosen to be 15, 17, and 20 respectively. Figure 2.7 illustrates a non-optimal schedule for the minimisation of the average job tardiness.

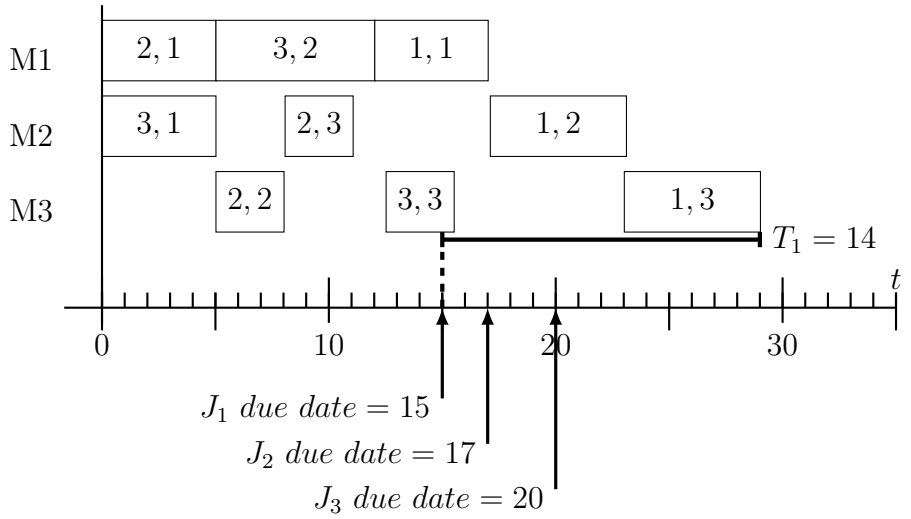


Figure 2.7: An example of a non-optimal schedule for the minimisation of average job tardiness

In Figure 2.7, both J_2 and J_3 are completed before their due dates, and therefore, both jobs have no tardiness. On the other hand, J_1 was completed 14 time units late, and therefore has a tardiness of 14. The average tardiness of the three jobs can then be calculated as 4.67 time units. The average tardiness of this schedule can be decreased by processing J_1 earlier and moving J_2 and J_3 later, because J_2 and J_3 have slack time available. The sequence of the jobs on each machine can therefore change to the sequence seen in Figure 2.8. The tardiness of J_1, J_2 and J_3 changed to 4, 0 and 2 respectively. The average tardiness of the three jobs has therefore decreased from 4.67 time units to 2 time units. Figure 2.8 shows an improved schedule regarding the average tardiness of all three jobs.

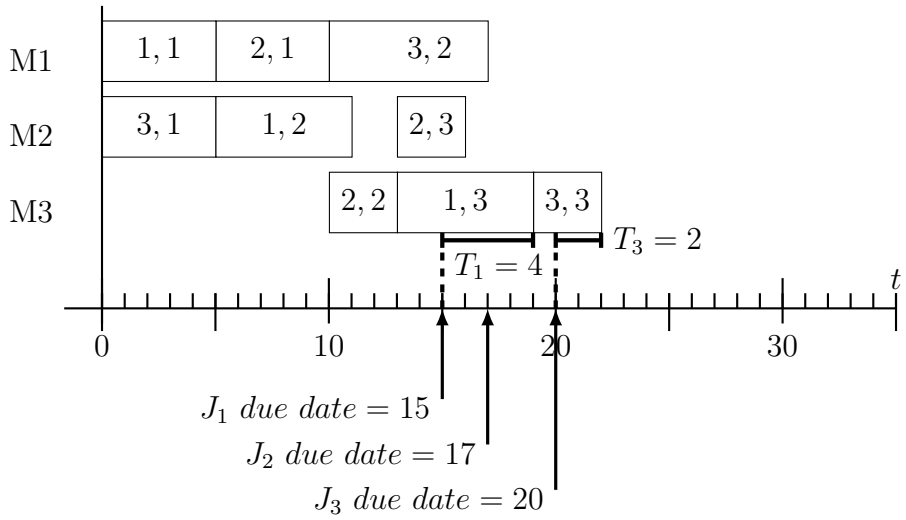


Figure 2.8: An example of an improved average tardiness schedule

Average job lateness

Jensen (2001) describes the improved robustness and flexibility of minimising the lateness of a job, rather than minimising the tardiness of a job. Due to the fact that $T_i = \max\{L_i, 0\}$, minimising L_i will also minimise T_i. The difference between these performance indicators can be observed in the minimising process. When T_i is minimised, the minimisation process will stop when $T_i = 0$ is reached, while when L_i is minimised, the minimisation process will continue even if $L_i \leq 0$, which means that J_i is actually early.

The model for minimising the average lateness of jobs within the job shop can be structured as

$$\text{Minimise } \overline{L} = \frac{1}{n}\sum_{i=1}^{n} L_i, \quad i = 1, \ldots, n, \tag{2.13}$$

subject to

$$s_{ih} - s_{ij} \geq \sum_{k}(X_{ijk}p_{ijk}), \qquad \text{for all } (i,h) \rightarrow (i,j) \in A, \; k = 1, \ldots, m, \tag{2.14}$$

where

$$L_i = \sum_{j}(C_{ij}) - d_i, \quad i = 1, \ldots, n. \tag{2.15}$$

The average lateness of a number of jobs can be calculated using (2.13), while the individual lateness of each job is calculated using (2.15), which allows the lateness to be negative. Constraint (2.14) ensures the precedence given to the operations of each job is not violated.

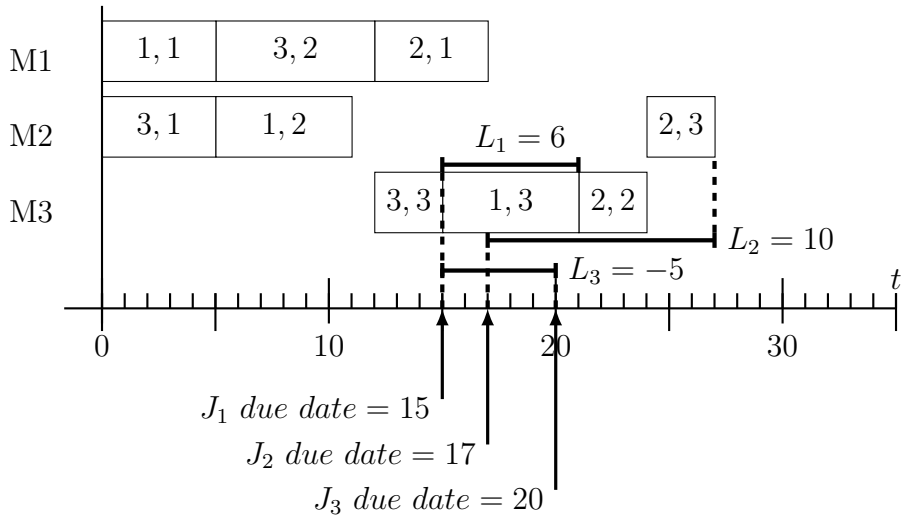


Figure 2.9: An example of a non-optimal schedule for the minimisation of average job lateness

The rescheduling process when minimising the average job lateness will now be discussed with reference to the example described in Table 2.3. To describe the rescheduling process, due dates for each job are also required as described in the average job tardiness rescheduling process. These two processes are similar, except that $L_i \leq 0$. Figure 2.9 illustrates a non-optimal schedule for the minimisation of average job lateness. The due dates used in the example are the same as those used in the rescheduling process of the average job tardiness (*i.e.* 15 for J_1, 17 for J_2, and 20 for J_3).

From Figure 2.9, it can be seen that the lateness of J_1, J_2 and J_3 are 6, 10, and -5 respectively. The average job lateness can therefore be calculated as 3.67 time units. The average lateness of this schedule can be decreased when decreasing the lateness of J_1 and J_2. The sequence of operations on each machine can therefore be changed to the sequence shown in Figure 2.10. The new schedule decreased the lateness of J_1 and J_2 from 6 and 10 time units to 4 and -2 time units, respectively. The lateness of J_3 has however increased from -5 to 2 time units. The average job lateness of the new schedule is therefore calculated to be 1.67 time units. Figure 2.10 is therefore an example of an improved schedule with regards to the minimisation of the average job lateness.

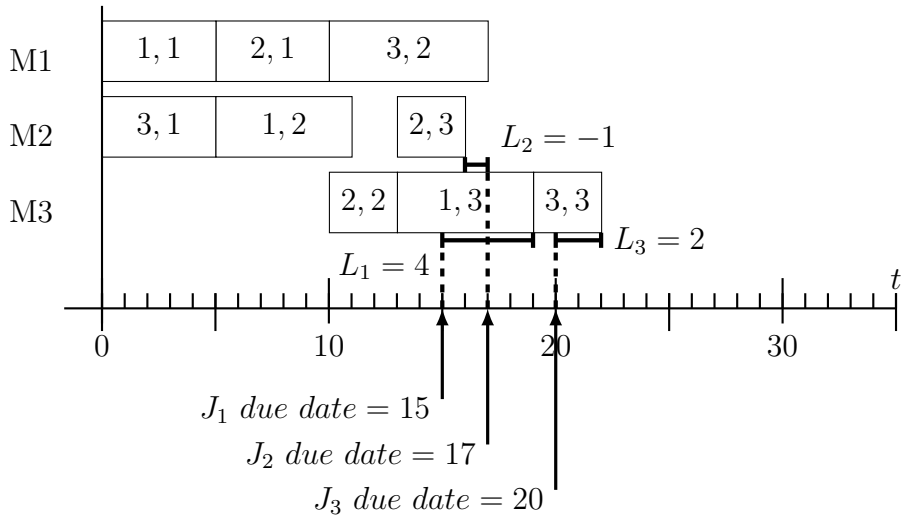


Figure 2.10: An example of an improved average job lateness schedule

Makespan

Kuhpfahl (2015) and Chakraborty (2009) formulated a mathematical model for the minimisation of the makespan of a job shop problem. This model is structured as

$$\text{Minimise } C_{max}$$

subject to

$$s_{ih} - s_{ij} \geq \sum_k (X_{ijk} p_{ijk}), \quad \text{for all } (i,h) \rightarrow (i,j) \in A, \; k = 1, \ldots, m, \tag{2.16}$$

$$C_{\max} - s_{ij} \geq \sum_k (X_{ijk} p_{ijk}), \quad \text{for all } (i,j) \in O, \; k = 1, \ldots, m, \tag{2.17}$$

$$s_{ij} \geq r_j, \quad \text{for all } (i,j) \in O. \tag{2.18}$$

Constraint (2.16) ensures the precedence given to the operations of each job is not violated, while constraint (2.17) ensures that the starting and processing times of a job do not exceed the maximum makespan. Lastly, constraint (2.18) ensures that a job cannot start before its release time. In the objective function, the makespan has to be minimised.

Satake *et al.* (1999) described the rescheduling process for minimising the makespan. A rescheduling example of a three-job four-machine job shop problem, was used in the study and is shown in Table 2.3.

A non-optimal schedule for the described problem can be seen in Figure 2.11. In this schedule, operation three of J_1 ($o_{1,3}$) gives the maximum completion time (*i.e. makespan*), which is found to be $C_{max} = 29$. In order to generate a better schedule, it is essential to

complete J_1 earlier to decrease the makespan. The rescheduling process starts by moving the start time of all operations to as late as possible, without changing the operation sequence on the machines and the makespan (as seen in Figure 2.12). Thereafter, the operations sequence on each machine is changed in order to complete J_1 earlier, as seen in Figure 2.13. If the precedence or machine processing constraints are violated, the new schedule is not adopted by the scheduler; otherwise the scheduler can move each operation earlier without delaying any other operations, which results in a better schedule. Figure 2.14 illustrates an example of an improved schedule.

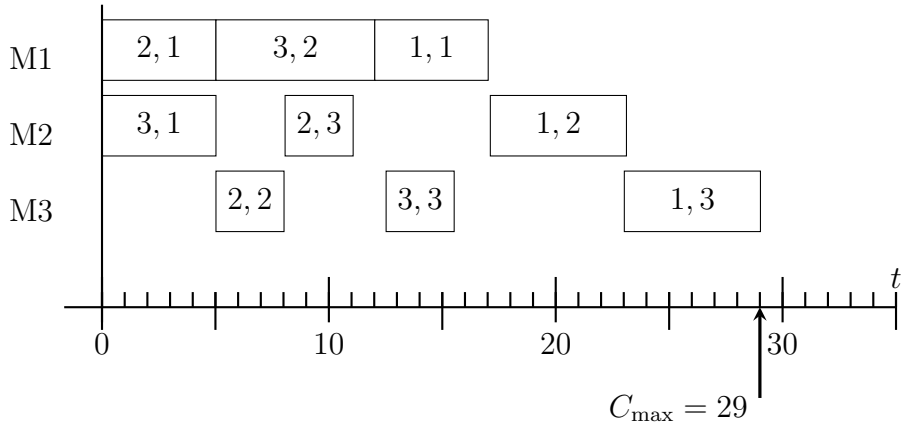


Figure 2.11: An example of a non-optimal schedule for the minimisation of makespan

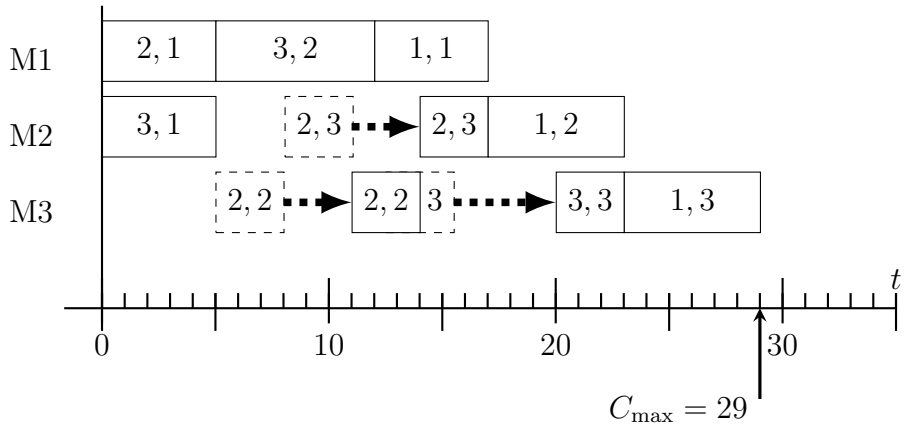


Figure 2.12: An example non-optimal schedule with the start times as late as possible

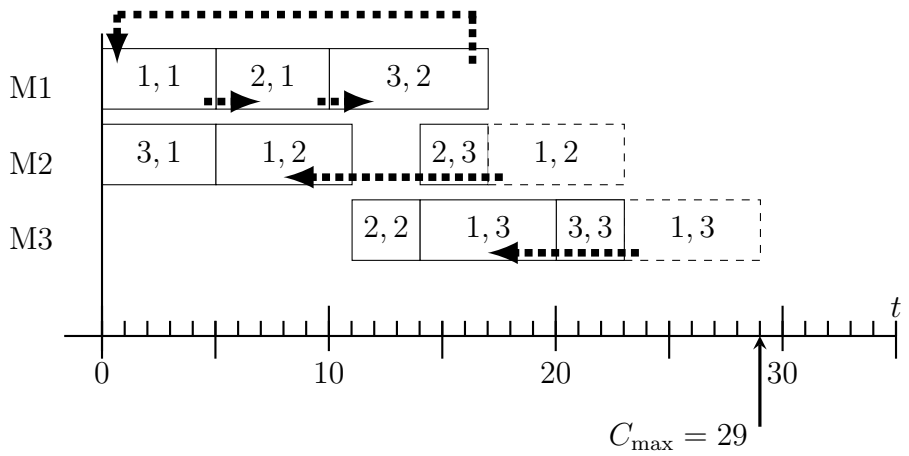


Figure 2.13: Changing the operation sequence on each machine

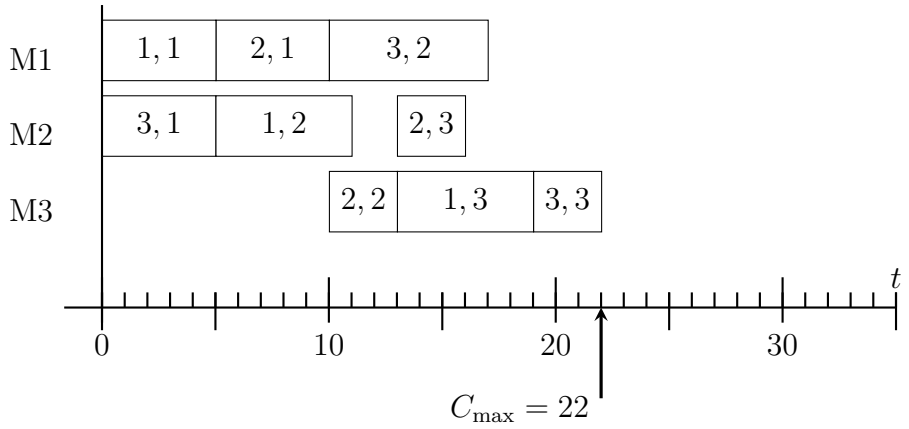


Figure 2.14: An example of an improved makespan schedule

2.3 Methods for job shop scheduling

Due to the advances of powerful computer capacity, several mathematical programming based scheduling methods have been researched. These methods can be divided into two categories, *i.e. exact methods* and *metaheuristics*. These categories will now be described in further detail.

2.3.1 Exact Methods

Exact/integer-based methods for solving the JSP include *mixed binary integer programming problems* (MBIPP), as well as the *branch-and-bound* method. The MBIPP refers to a problem where only some of the variables are required to be binary (Winston and Goldberg,

2004). Conway *et al.* (2012); Nasr and Elsayed (1990); Ruiz *et al.* (2009) and Rey *et al.* (2015) introduce the formulation of a MBIPP to solve the JSP. The introduced formulation contains several binary variables, *i.e.* a sequence variable, as well as a machine selection variable. The sequence variable $Y_{iji'j'}$ generates the sequence between operations $o_{i,j}$ and $o_{i',j'}$, while the machine selection variable X_{ijk} is used to select a machine for operation $o_{i,j}$.

This formulation is used by Trentesaux *et al.* (2013) in the data preparation phase of their study, to develop a benchmark system based on a real production system. This benchmark allows the evaluation of static optimisation performances using traditional operation research tools and the evaluation of the control system's robustness faced with unexpected events. Nasr and Elsayed (1990) also use this problem formulation to investigate the minimisation of the mean flow time in a general job shop machining system. Finally, Pan and Chen (2005) describe the development of MBIPP formulations for the re-entrant JSP, where they developed two-layer division procedures in order to improve the solution speed of the MBIPP formulations.

The other exact method used to solve the JSP is the branch-and-bound method. Brucker *et al.* (1994) describe the branch-and-bound algorithm for the job shop scheduling problem, which is represented by a search tree. Initially, the search tree contains only one node, which is the root and represents all the feasible solutions of the problem. The successors of the root are calculated by fixing *disjunctions*, which are the links between every pair of operations that need to be processed on the same machine (as displayed in Figure 2.15). Each successor is recursively handled in the same way, and the examination of the search tree node stops if it represents only one solution, or the node does not contain an optimal solution.

Figure 2.15 shows an example of a disjunctive graph of a problem with three jobs and two machines. The problem starts at source "0" and ends at sink "•". The figure also displays the disjunctions between the operations that need to be processed on the same machine, *e.g.* between O_{11}, O_{21} and O_{31}, as well as between O_{12} and O_{32}.

Baker (2014) developed a branch-and-bound algorithm to find optimal solutions to the single-machine stochastic scheduling problem, with the objective of minimising the total expected earliness and tardiness costs. The branch-and-bound method was also used in a study by Mascis and Pacciarelli (2002), where the JSP with blocking and/or no-wait constraints, is examined. Mascis and Pacciarelli used a *depth-first* search strategy in the branch-and-bound method, where the exploration progresses node by node down a branch. However, if the previous node fathomed the branch, the search retreats up the branch until it encounters a node that has not been fully explored.

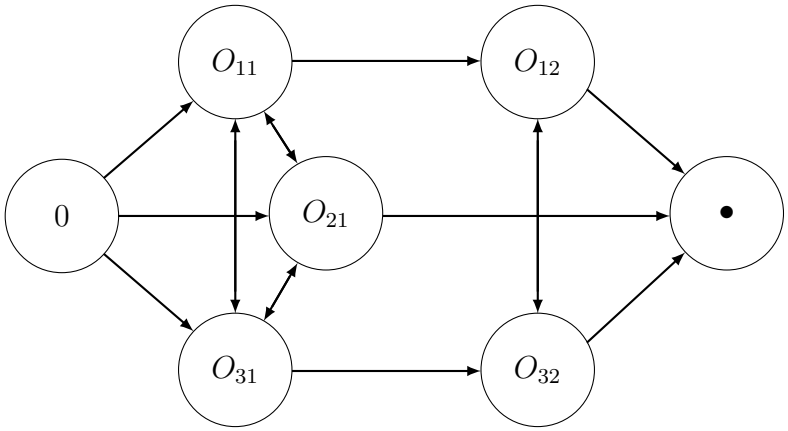


Figure 2.15: A disjunctive graph of a problem with three jobs and two machines

2.3.2 Metaheuristics

Voß *et al.* (2012) state that "a metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions". Furthermore, the family of metaheuristics includes, but is not limited to, tabu search, ant systems, greedy randomised adaptive search, genetic algorithms, neutral networks and simulated annealing. Due to NP-hard and computationally challenging nature of JSPs with larger dimensions, it is still considered to be beyond the reach of today's exact methods (Gonçalves *et al.*, 2005). Therefore, over the last two decades, metaheuristic procedures have been presented to solve the JSP. According to Della Croce *et al.* (1995); Geyik and Dosdogru (2012); Gonçalves *et al.* (2005); Pezzella *et al.* (2008); Suresh and Mohanasundaram (2006); Wang and Zheng (2002); Zhang *et al.* (2008), and Wang *et al.* (2013), the three most commonly used metaheuristics to solve the JSP, include:

- genetic algorithm,
- simulated annealing, and
- tabu search.

These algorithms are the most popular, because they are generic optimisation algorithms, and therefore they can be used for a wide range of optimisation problems. These three metaheuristics will now be discussed in detail, where the algorithm of each metaheuristic will be presented, as well as previous implementations. There are other metaheuristics that have also been used for solving JSPs, but due to the metaheuristics mentioned above being more commonly used, these other metaheuristics will only be mentioned.

Genetic algorithm (GA)

The term *genetic algorithm* was first used by Holland (1975), and is motivated by the theory of evolution, with the survival of the fittest being a fundamental property. GAs have been designed as general search strategies and optimisation methods. Bekker (2016b) describes the analogy when applying the GA to an optimisation problem as: a *population* of possible solutions is created from the problem's *solution space* by varying the values of the *decision variables*. Each individual/*chromosome* in the population is an *encoding* of a solution. Some of the individuals (referred to as *parents*) combine with each other to form new individuals or *offspring*. The combination phase is called the *crossover* phase, and good individuals are more likely to be selected for the phase. This phase is also subjected to random variation (referred to as *mutation*) where some part of the chromosome is randomly changed. The new generation of chromosomes replaces some or all of the previous generation, thus causing the population of solutions to improve over time.

The basic GA is outlined by Bekker (2016b) in Algorithm 1. There are many variations on this algorithm, but the principles are similar.

Algorithm 1 Basic Genetic Algorithm

```
Generate an initial random population and evaluate members. Set t = 1.
repeat
    if the cross-over probability is satisfied then
        Randomly select two individuals to reproduce
        Do cross-over between the members
    else
        Choose any of the parents to be the offspring
        Replace the worst individual in population by new offspring
        For each gene in the offspring
        if mutation probability is satisfied then
            Flip value of gene
        end if
    end if
    Evaluate offspring with the objective function
    if offspring is better than the least fit member in the population then
        Replace the least fit member with the offspring
    end if
    t ← t + 1
until Termination condition
```

The implementation of Algorithm 1 will now be described with reference to a study conducted by Vilcot and Billaut (2008). The study investigated the general JSP with multiple constraints. Although the study aimed to optimise two conflicting objectives simultaneously, the explanation that follows deals with the encoding of the problem and not the result obtained. The steps followed in the encoding of the problem are as follows:

Step 1 *Initial population selection*

The initial population of the study was obtained by a two-step procedure. Firstly, an assignment is determined for each operation, and secondly, a job shop schedule problem is obtained. The initial assignment is determined by using a heuristic that aims to balance the resource workload, which is explained by Dauzère-Pérès and Paulli (1997). After the first step, the JSP was created, which they solved by a greedy algorithm based on a slack rule. A set of candidate operations is denoted by $\mathscr{S}$, which contains all the operations without predecessors, at the start. The candidate operations are sorted in non-decreasing order of their release time. The release time of operation $o_{i,j}$ is the maximum between the time its resource is free for processing operation $o_{i,j}$ plus the set-up time, and the maximum completion time of all predecessors of $o_{i,j}$. The first candidate operation is then scheduled, after which release times are updated and $\mathscr{S}$ is updated. The process then iterates while $\mathscr{S} \neq \emptyset$.

Step 2 *Encoding of an individual*

Each individual is coded by a matrix $\mathscr{C}$ with m rows (one per resource), where $\mathscr{C}_{\ell,k}$ indicates the operation in position k on resource M_ℓ (where $1 \leq \ell \leq m$). The value of k is between one and the maximum number of operations assigned to a resource. The matrix

$$\mathscr{C} = \begin{bmatrix} o_{2,2} & o_{3,2} & - & - \\ o_{2,1} & o_{1,2} & o_{2,3} & o_{3,3} \\ o_{3,1} & o_{1,1} & o_{1,3} & - \end{bmatrix},$$

represents the coding of a solution where three jobs, having linear routes, are sequenced as $(o_{2,2},\ o_{3,2})$ on resource M_1, $(o_{2,1},\ o_{1,2},\ o_{2,3},\ o_{3,3})$ on M_2 and $(o_{3,1},\ o_{1,1},\ o_{1,3})$ on M_3, as illustrated in Figure 2.16.

Step 3 *Crossover operation*

In the crossover operation, two individuals (a) and (b) (as seen in Figure 2.17) are selected according to a binary tournament selection operator. Thereafter, a matrix $\mathscr{B}$ of Booleans with the same profile as (a) is randomly generated, where each element has equal probability to be '0' or '1'. The encoding of (a), (b) and the random

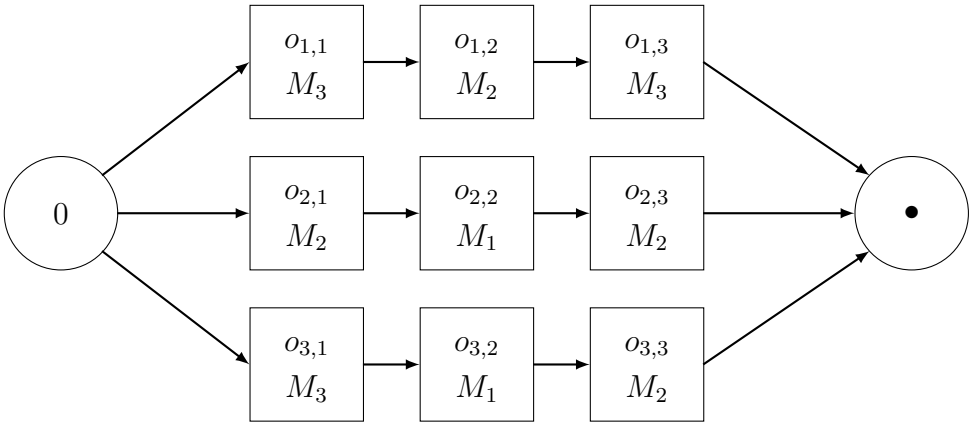


Figure 2.16: Encoding of a solution for the JSP (reproduced from Vilcot and Billaut (2008)).

Boolean matrices are

$$(a) = \begin{bmatrix} o_{2,2} & o_{3,2} & - & - \\ o_{2,1} & o_{1,2} & o_{2,3} & o_{3,3} \\ o_{3,1} & o_{1,1} & o_{1,3} & - \end{bmatrix}, \quad (b) = \begin{bmatrix} o_{1,1} & o_{1,2} & o_{2,3} \\ o_{3,1} & o_{1,3} & o_{3,3} \\ o_{2,1} & o_{2,2} & o_{3,2} \end{bmatrix},$$

$$\mathscr{B} = \begin{bmatrix} 0 & 1 & - & - \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & - \end{bmatrix}.$$

The '1' in $\mathscr{B}$, means that the corresponding operation in (a) will keep the same position in the child, which can be seen in the first step of the child matrix. The operations of (a) that are kept in the child, are then deleted in (b). The final child is then completed by filling the holes with the remaining operations, as seen in the final child matrix. If no slots are available to set an operation, the operation is put at the end of the sequence on the same resource as in (b). However, if there are slots remaining at a given resource, the sub-sequences are shifted to the left.

$$\text{Child first step} = \begin{bmatrix} - & o_{3,2} & - & - \\ o_{2,1} & o_{1,2} & - & - \\ o_{3,1} & - & o_{1,3} & - \end{bmatrix} \quad \begin{matrix}\text{Remaining} \\ \text{operations in } b\end{matrix} = \begin{bmatrix} o_{1,1} & \cancel{o_{1,2}} & o_{2,3} \\ \cancel{o_{3,1}} & \cancel{o_{1,3}} & o_{3,3} \\ \cancel{o_{2,1}} & o_{2,2} & \cancel{o_{3,2}} \end{bmatrix}$$

$$\text{Final child} = \begin{bmatrix} o_{1,1} & o_{3,2} & o_{2,3} \\ o_{2,1} & o_{1,2} & o_{3,3} \\ o_{3,1} & o_{2,2} & o_{1,3} \end{bmatrix}$$

After the crossover operation, the final child may not be feasible, due to the possible generation of a cycle. In this case, the operation belonging to the cycle and having a '0' in matrix $\mathscr{B}$, must be identified. Thereafter, the value of this element is changed to '1' in matrix $\mathscr{B}$ and a new child is generated. There is, however, a worst case, where all the values in the Boolean matrix $\mathscr{B}$ are equal to '1', which then generates a final child that is a clone of matrix (a) which is feasible.

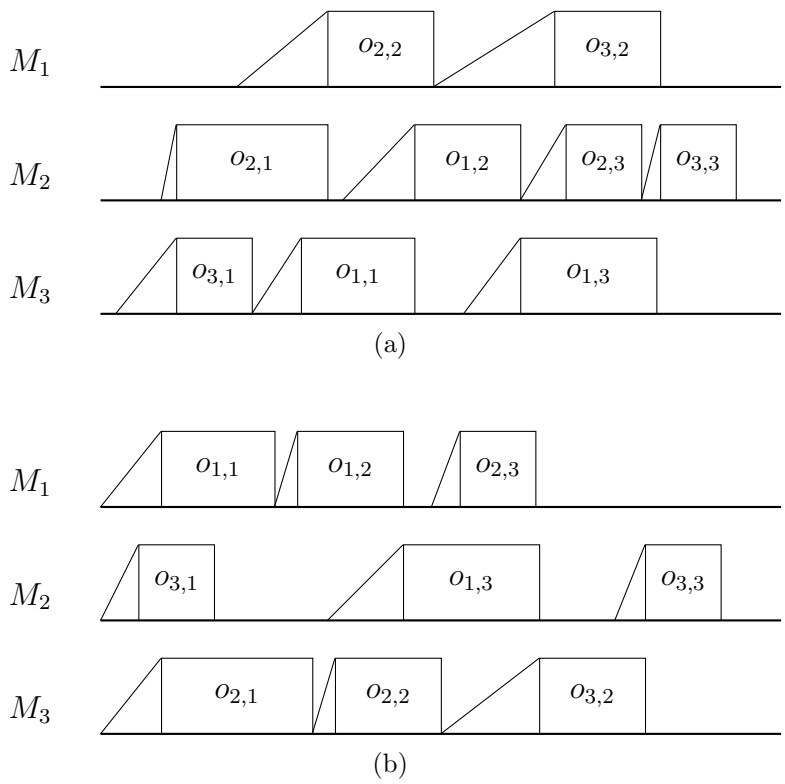


Figure 2.17: (a) Individual a, (b) Individual b (reproduced from Vilcot and Billaut (2008)).

Step 4 *Mutation operation* The mutation operator is used on an individual. The operator, used in the study, is based on a local search incorporating individual neighbours. A neighbour is obtained by moving a critical operation. In the conjunctive graph, moving an operation x between operations y and z, means deleting and adding arcs corresponding to resource constraints. This can be described by:

- deleting arc (v, x) and (x, w), where v is the predecessor and w the successor of operation x on the related resource,
- adding an arc (v, w),
- deleting the arc (y, z),
- adding arcs (y, x) and (x, z).

The assignment and sequence can therefore be modified when using this technique. A feasible move is one in which operation x is assigned to a resource where it can be processed and such that no circuit is generated in the graph. During the mutation operation, one of the individual neighbours is randomly chosen as a new individual.

Other studies regarding the application of GAs on the JSP include the development of a new GA for a flexible JSP, by Pezzella *et al.* (2008). This new algorithm integrates

different strategies for the generation of initial populations, as well as selecting individuals for reproduction. The results from this study proved that the integrating of more strategies in the generic framework leads to better results with respect to other GAs. Chen *et al.* (2008) applied the GA to a JSP to achieve lower tardiness, machine idle time, and makespan. In another study, Lei (2012) addressed the stochastic JSP that is subject to breakdown. In the study, Lei proposes an efficient GA with the objective of minimising the makespan. The GA was then tested on 22 benchmark problems and then compared with a simulated annealing and particle swarm optimisation algorithm. In another study, Rebai *et al.* (2012) developed a new metaheuristic which combines GA, local search, and the branch-and-bound algorithm, which resulted in a small deviation of the metaheuristic solution from the optimal solution. The effectiveness of metaheuristics as new approaches for solving hard scheduling problems, was therefore confirmed. There are at least 18 other implementations of GAs on the JSP in Çaliş and Bulkan (2015).

Simulated annealing (SA)

Simulated annealing was first introduced by Kirkpatrick *et al.* (1983), and is motivated by the annealing process of slowly cooling metals to increase strength. The annealing process provides a framework for optimisation of the properties of large and complex systems. Furthermore, Rardin (1998) states that SA algorithms control cycling by accepting non-improving moves according to probabilities tested with computer-generated random numbers.

The analogy when applying the SA to an optimisation problem, can be described as: the move selection at each iteration begins with a random choice of feasible move, ignoring the impact it has on the objective function. Thereafter, the objective function improvement (ΔObj) is computed for the chosen move. The ΔObj can be non-positive if the move is non-improving. The move is always accepted if it improves the objective function value ($\Delta Obj > 0$), otherwise the probability of accepting the move is $e^{\frac{-\Delta Obj}{q}}$. The parameter q is known as the *temperature*, which controls the randomness of the search. If the q-value is large, the exponent in the acceptance probability approaches zero, which implies that the probability of accepting a non-improving move is approximately $e^0 = 1$. On the other hand, if the q-value is small, the probability of accepting a non-improving move decreases dramatically. The SA algorithm normally starts with a larger q-value which decreases when a sufficient number of iterations has passed since the last temperature change. The algorithm, as with other searches allowing non-improving moves, maintains an *incumbent solution* $\hat{x}$, which keeps track of the best feasible solution found thus far. Lastly, the search terminates when the termination conditions of either a predetermined number of iterations

Algorithm 2 Simulated Annealing

1: Choose any starting feasible solution $x^{(0)}$, an iteration limit $t_{\max}$, and a large initial temperature $q > 0$
2: Set incumbent solution $\hat{x} \leftarrow x^{(0)}$, and $t \leftarrow 0$
3: **repeat**
4: Randomly choose a feasible move Δx in move set $\mathcal{M}$ as $\Delta x^{(t+1)}$
5: Calculate objective function improvement for the movement of $\Delta x^{(t+1)}$
6: **if** Objective function improves or *acceptance probability* is satisfied **then**
7: Accept the move $\Delta x^{(t+1)}$
8: Update $x^{(t+1)} \leftarrow x^{(t)} + \Delta x^{(t+1)}$
9: **if** Objective function of $x^{(t+1)}$ is superior to incumbent solution **then**
10: $\hat{x} \leftarrow x^{(t+1)}$
11: **end if**
12: **end if**
13: **if** Temperature change condition is satisfied **then**
14: Reduce temperature q
15: **end if**
16: $t \leftarrow t + 1$
17: **until** Termination condition
18: Output incumbent solution $\hat{x}$ as an approximate optimal solution

($t_{\max}$) is reached, or if some solution has no feasible solution in its neighbourhood. When the search terminates, the incumbent solution $\hat{x}$ is output as an approximately optimal solution. The SA algorithm is outlined by Rardin (1998) in Algorithm 2.

The implementation of Algorithm 2 will now be discussed with reference to a study by Aydin and Fogarty (2004). The study presents an implementation of the modular SA algorithm for classical job shop scheduling.

The algorithm used in the study evolves the population of solutions with a modular SA operator up to a predefined number of iterations. The algorithm starts with a population of solutions that are randomly initialised, which is followed by choosing the number of iterations. Thereafter, a solution is randomly selected to be operated by the modular SA operator. The starting temperature is chosen to be 100, which modifies the randomly selected solution by using the neighbouring function. It is also used to judge whether the yielded solution will be promoted to the next iteration. The algorithm then applies a cooling process to the temperature with the use of a cooling coefficient of 0.955, at the end of each iteration. When the process cools to a temperature of 0.01, the modular SA

Algorithm 3 SA for the JSP proposed by Aydin and Fogarty (2004)

Begin
Initialise the *population*
repeat
    Pick one *feasible solution (old)*
    Set the *temperature (100)*
    repeat
        Select a particular task
        Conduct a move by *neighbourhood function*
        Repair the *new solution (new)*
        if $(new - old) < 0$ **then**
            Replace *old* with *new*
        else
            Generate a *random number (r)*
            if $e^{\frac{-(new-old)}{temperature}} > r$ **then**
                Replace *old* with *new*
            end if
        end if
    until $temperature < 0.01$
    Put the solution back into *population* subject to *replacement rule*
until Pre-defined number of iterations
End

operator releases the solution. The solution obtained through this process is then put back into the population subject to a replacement rule. If the replacement rule is satisfied, the new solution is replaced with the old one. That is the end of one iteration, and the process repeats until the total number of iterations is completed. The algorithm used in the study is presented in Algorithm 3.

Other implementations of the SA algorithm include a study by Ogbu and Smith (1990), where the objective of minimising the makespan in a flow shop was addressed. At the time when the study was executed, scheduling problems have received less attention in the literature than other combinatorial problems. In the study, Ogbu and Smith propose a modification to the Metropolis scheme which is simpler to use and whose results provide near-optimal schedules in reasonable computation time. In another study, Suresh and Mohanasundaram (2006) investigated a multi-objective JSP with the objectives of minimising the makespan and the mean flow time of jobs. The study proposed the Pareto archived SA algorithm to discover non-dominated solution sets for the JSPs. A segment-random

insertion scheme was used to generate a set of neighbourhood solutions to the current solution. The performance of the proposed algorithm was also evaluated by solving benchmark JSPs.

Roshanaei *et al.* (2013) performed a study, where they proposed a hybrid metaheuristic for solving the flexible JSP. The study was motivated by the NP-hard nature of the flexible JSP, which renders mixed integer linear programming models inefficient in generating schedules with the desired quality for industrial-scale problems. The objective function that the metaheuristic solved was the minimisation of the makespan of jobs in the flexible JSP. Zhang *et al.* (2008) developed a heuristic search approach that combines the SA and Tabu search strategy. This approach makes use of SA to find the elite solutions, while using tabu search to reintensify the search from the promising solutions. The approach was also used to solve the objective of minimising the makespan of jobs. In another study Andresen *et al.* (2008) investigated the SA algorithm, where several neighbourhoods were suggested and tested together with the control parameters of the algorithm. The study also included extensive computational results for problems with up to 50 jobs and 50 machines. Lastly, a study by Zhang and Wu (2010) was performed, where a hybrid SA algorithm based on a novel immune mechanism is proposed for the JSP. The objective of this problem was to minimise to total weighted tardiness of jobs. The study is motivated by bottleneck jobs, which require more intensive optimisation. Therefore, the bottleneck level (also known as the criticality) for each job is evaluated, to quantitatively describe the bottleneck job distribution.

Tabu search (TS)

The TS algorithm was first proposed by Glover (1986), who describes the metaheuristic as an approach that strives to exceed local optimality by a strategy of forbidding certain moves. The purpose of forbidding certain moves, or making the moves "tabu", is mainly to prevent cycling. Moves that are classified tabu are generally a small fraction of those available, and a move loses its tabu classification after a relatively short time to become accessible again.

Rardin (1998) describes that the TS algorithm makes use of a *tabu list* that records forbidden moves. At each iteration, a non-tabu feasible move is chosen for the next iteration, while all the moves returning immediately to the previous point are added to the tabu list. The moves on the tabu list are not allowed for a few iterations, but are eventually removed from the tabu list. Since the moves may either improve or degrade the objective function value, an *incumbent solution* $\hat{x}$ is maintained throughout the search. The $\hat{x}$ tracks the best feasible solution found thus far, and is reported as an approximate optimum when

Algorithm 4 Tabu Search

Begin
Choose any starting feasible solution $x^{(0)}$, an iteration limit $t_{\max}$
Set incumbent solution $\hat{x} \leftarrow x^{(0)}$, and $t \leftarrow 0$
No moves are tabu
repeat
    Choose a non-tabu feasible move Δx in move set $\mathcal{M}$ as $\Delta x^{(t+1)}$
    Update $x^{(t+1)} \leftarrow x^{(t)} + \Delta x^{(t+1)}$
    Calculate objective function value of $x^{(t+1)}$
    if Objective function of $x^{(t+1)}$ is superior to incumbent solution **then**
        Replace $\hat{x} \leftarrow x^{(t+1)}$
    end if
    Remove forbidden moves from tabu list
    Add moves that immediately return to previous point to tabu list
    $t \leftarrow t + 1$
until Termination condition
Output incumbent solution $\hat{x}$ as an approximately optimal solution

the search stops when a termination condition is satisfied. The termination condition is satisfied when there are no non-tabu moves that lead to a feasible neighbour of the current solution, or at a predefined iteration limit ($t_{\max}$). This generic TS algorithm is presented by both Glover (1989) and Rardin (1998), as seen in Algorithm 4.

The implementation of Algorithm 4 will now be described with reference to a study conducted by Vilcot and Billaut (2008). The study investigated the general JSP with multiple constraints. As stated previously, the study aimed to optimise two conflicting objectives simultaneously, but the explanation that follows deals with the coding and the setup of a tabu list for the problem. The algorithm used in the study was structured as:

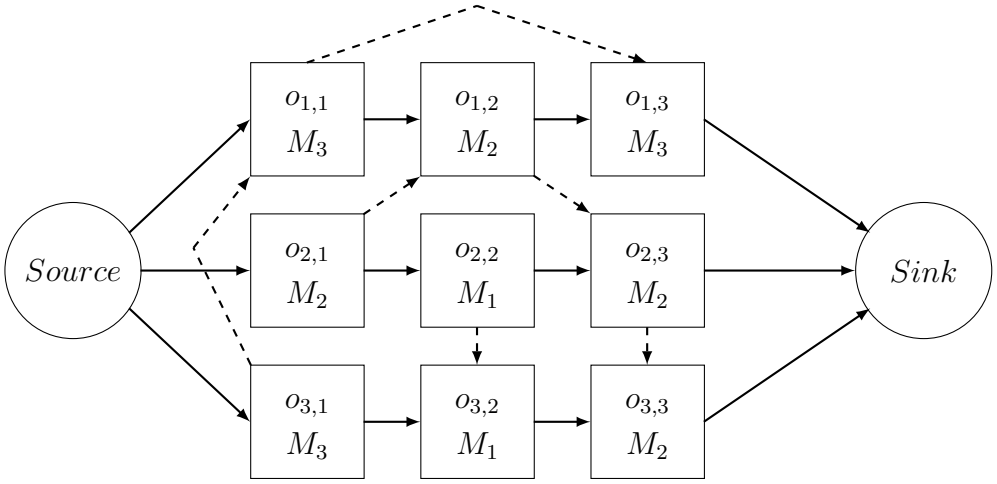


Figure 2.18: Coding of a solution for the TS algorithm (reproduced from Vilcot and Billaut (2008)).

Step 1. *Coding of a solution*

The coding of a solution for the JSP can be described with reference to Figure 2.18, where three jobs are processed on three machines with a sequence of $(o_{2,2},\ o_{3,2})$ on resource M_1, $(o_{2,1},\ o_{1,2},\ o_{2,3},\ o_{3,3})$ on M_2 and $(o_{3,1},\ o_{1,1},\ o_{1,3})$ on M_3. There is an arc between:

- the source and all the initial operations of each job;
- operation $o_{i,j}$ and operation $o_{i',j'}$, if a routing constraint imposes that $o_{i,j}$ precedes $o_{i',j'}$ on resource M_k;
- operation $o_{i,j}$ and operation $o_{i,j+1}$ that ensures that the operation of a job is completed in the correct sequence/order.

Step 2. *Initial solution*

The initial solution is obtained using the same two-step procedure as described in Step 1 of the GA implementation of Vilcot and Billaut (2008).

Step 3. *Neighbourhood*

A neighbour of the current solution is obtained by moving a critical operation. Therefore, moving an operation x between operations y and z means deleting and adding arcs corresponding to resource constraints. This can be described by:

- deleting arc (v, x) and (x, w), where v is the predecessor and w the successor of operation x on the related resource,
- adding an arc (v, w),
- deleting the arc (y, z),

- adding arcs (y, x) and (x, z).

The assignment and sequence can therefore be modified when using this technique. A feasible move is one in which operation x is assigned to a resource where it can be processed and such that no cycle is generated in the graph. For the TS algorithm, the best neighbour is the neighbour that is not in the tabu list and improves the objective function.

Step 4. *Tabu list*

The tabu list is used to prevent the search from cycling between solutions. In the study, the authors applied a fixed tabu list size. This means that when the list is full, the oldest tabu solution in the list is replaced with a new tabu solution. The oldest tabu solution then changes to a non-tabu solution and can be applied once more. The tabu list used in the study worked as follows: after moving x between y and z, (x, y) is added to the tabu list. That same move is therefore now forbidden, because it belongs to the tabu list.

Other studies that implemented the TS algorithm to solve the JSP include a study by Saidi-Mehrabad and Fattahi (2007). The proposed TS algorithm consisted of two parts: a procedure that searches for the best sequence of job operations, and a procedure that finds the best choice of machine alternatives. The initial feasible solution of jobs and operations sequence is first generated and then the algorithm searches for the best choice of the machine alternatives for these jobs and operations sequence. In another study by Li *et al.* (2010a), a hybrid TS algorithm was proposed to solve three minimisation objectives (*i.e.* makespan, total workload of machines, and the workload of critical machines) simultaneously. The hybrid TS algorithm was tested on sets of well-known benchmark instances, and the performance of the proposed algorithm was found to be superior to several existing algorithms.

Li *et al.* (2011), in another study, proposed a novel hybrid TS algorithm with a fast public critical block neighbourhood structure. This algorithm was used to solve the flexible JSP with the objective of minimising the makespan of jobs. The study concluded by comparing the performance of the proposed algorithm to several algorithms, where the proposed algorithm achieved superiority in terms of solution quality, convergence ability and efficiency. Pezzella and Merelli (2000) presents a computationally effective heuristic method for solving the minimum makespan problem of job shop scheduling. The proposed heuristic is based on the TS, as well as the shifting bottleneck procedure used to generate the initial solution and to refine the next solutions. There are more implementations of the TS algorithm to the JSP, and the reader is therefore referred to studies by Dauzère-Pérès

and Paulli (1997); Dell'Amico and Trubian (1993); Hurink *et al.* (1994); Zhang *et al.* (2008) and Barnes and Chambers (1995).

This concludes the detailed discussion of the most popular metaheuristics used in job shop scheduling. Other metaheuristics are briefly mentioned in the next subsection.

Other metaheuristics

Other metaheuristics used to solve the JSP, as described in a study by Çaliş and Bulkan (2015), include *Agent based systems* (ABS), *Ant colony optimisation* (ACO), *Neural networks* (NN), *Particle swarm optimisation* (PSO), *Variable neighbourhood search* (VNS) and *Fuzzy logic* (FL).

Agent-based systems (ABS)

The term "agent" was first used in a study by Holland and Miller (1991), which then evolved into the metaheuristic. An example where the ABS metaheuristic was used to solve a JSP, is in a study by Usher (2003). The objective of the study was to minimise the number of tardy jobs. The study included an experimental approach for performance analysis of a multi-agent system for job routing in the job shop environment. In another study by Aydin and Öztemel (2000), ABS were used to solve the objective of minimising the mean tardiness in the JSP. In the study, an intelligent agent-based dynamic scheduling system was proposed. From the results of these studies, conclusions could be drawn that ABS provide encouraging solutions and the performance of the agent will be improved by enriching the environment as well as the skills of the agent. Due to the success of ABS in scheduling, both studies also concluded that the use of ABS is not limited to manufacturing control, but can also be applied to manufacturing design and planning.

Ant colony optimisation (ACO)

The study of Goss *et al.* (1989) on the collective behaviour of Argentine ants, provided the idea of ACO algorithms, which was developed and introduced by Dorigo and Di Caro (1999). Examples where the ACO algorithm was used to solve the JSP include a study by Huang and Liao (2008), where a hybrid algorithm combining ACO and TS is proposed. This algorithm was used with the objective of minimising the makespan of jobs in a job shop. In another study by Surekha and Sumathi (2010), an ACO is presented for solving the minimisation of the makespan in a JSP. These studies concluded by testing the algorithms on benchmark instances and compared the results with other algorithms. The proposed algorithms yielded similar or slightly improved solutions to that of the best-performing algorithms for the JSP. The proposed algorithms did however obtain these results by using

higher computation times than recorded for the other algorithms.

Neural networks (NN)
The history of NN dates back to a study by McCulloch and Pitts (1943), when simple types of NN where shown to be able to learn logical functions. In a study by Weckman *et al.* (2008), a NN scheduler was successfully developed, which provided a close approximation to the performance of a GA scheduler for JSPs. The objective of the study was to minimise the makespan of jobs. In another study by Fonseca and Navaresse (2002), the use of NN as a valid alternative to the traditional job shop simulation approach, was explored. The study concluded that NN-based simulations were able to fairly capture the underlying relationship between the machine sequences of jobs and their resulting average flow times.

Particle swarm optimisation (PSO)
The PSO algorithm was first developed and used by Kennedy and Eberhart (1995). Studies that applied the PSO algorithm to the JSP include a study by Zhang *et al.* (2009), where an effective hybrid PSO algorithm was proposed to solve the multi-objective JSP. The objectives that were addressed in this study were the makespan, maximal machine workload, and the total workload of machines. In another study by Li *et al.* (2010b), a hybrid algorithm combining PSO and TS was proposed to solve the flexible JSP with the criterion of minimising the makespan. Both studies concluded that the proposed hybrid PSO algorithms are effective in solving the variations of the JSP, and that they provide better results than the GA. The higher computation time of these hybrid algorithms still poses problems, and therefore future work proposed in these studies refer to decreasing the computation times of the algorithms.

Variable neighbourhood search (VNS)
VNS was first proposed in a study by Mladenović and Hansen (1997), as a metaheuristic for solving combinatorial and global optimisation problems. Roshanaei *et al.* (2009) proposed a VNS algorithm to solve the JSP, where the set-up times were sequence dependent on each processor to minimise the makespan. In another study, Yazdani *et al.* (2010) proposed a parallel VNS algorithm to solve the flexible JSP, by minimising the makespan time. The studies concluded by testing the performance of the proposed algorithms on benchmark instances. The computational results showed the competent performance of the proposed algorithms, and verified that the VNS had lower computational time than other well-known algorithms.

Fuzzy logic (FL)
Zadeh (1965) conceived the concept of FL, which is a method that incorporates uncertainty into a decision model. Examples of the implementation of FL to the JSP, include a study by Bilkay *et al.* (2004). The study consists of two stages, *i.e.* assigning priorities to part types with the use of a FL-based algorithm, and an operation-machine allocation and scheduling stage. In another study, Sakawa and Kubota (2000) introduced job shop scheduling with fuzzy processing times and fuzzy due-dates. The fuzzy JSP is then solved with the help of a GA. The study concluded with simulations for both 6x6 and 10x10 multi-objective fuzzy job shop scheduling problems. The computational results illustrated that the combination of the fuzzy JSP and the GA provided better results in all trials compared to the SA algorithm.

2.4 Introduction to real-time stochastic scheduling

Vredeveld (2012) states that 'in standard deterministic scheduling all relevant data to the problem is known beforehand', but in a real-world problem, this assumption is not always realistic. In many scenarios, a good schedule needs to be found when data is incomplete and decisions with wide-ranging implications need to be made. An approach that can be used to cope with the uncertainty, is *stochastic scheduling*.

The attributes of the stochastic scheduling problem, including *processing times*, *due dates*, *unexpected releases of high-priority jobs*, and *machine up/downtimes*, may be considered as variables. Furthermore, *probability distributions* are used to describe the variables, and ultimately help to determine a schedule in a stochastic environment (Cai *et al.*, 2014; Pinedo, 2016). While the values of random variables for the problem attributes are unknown before they are observed, their probability distributions are assumed to be known. Hence a schedule can be established based on these probability distributions. A solution to a stochastic scheduling problem can be determined by using the probability distributions and then minimising the expected values of the original deterministic performance measures, according to Baker (2014).

A fundamental complication of a scheduling problem, is to determine an optimal strategy to complete a set of jobs by one or more machines. Cai *et al.* (2014) describes four aspects for a model of such a problem:

1. *Jobs*: A job in a scheduling problem can have a wide sense. It may be a simple task in manufacturing; a computing program; a journey from one place to another; or a set of tasks in a complex project.

2. *Machine*: A machine represents a facility to process the jobs, hence also referred to as a processor.
3. *Policy*: A policy is a strategy to determine how jobs are to be processed, such as the order in which jobs are processed, or which job is to be processed at a given point in time.
4. *Performance measures*: Each scheduling problem has a performance measure as a basis to compare the outcomes of different scheduling strategies and determine the optimal solution to the problem.

2.4.1 Mathematical representation of stochastic scheduling

In this section, a stochastic programming model, formulated by Gu *et al.* (2010), minimising the expected makespan is discussed, to solve a stochastic job shop scheduling problem. The probability distributions of the processing times are assumed to be known, while the actual outcome of the random processing time only becomes known at the completion of the processing. Furthermore, the assumptions of the mathematical model include:

(1) All machines are always available.

(2) A machine can process only one job at a time, and a job is not allowed to visit the same machine twice.

(3) Set-up and removal times are included in processing times.

(4) The transportation times between machines are negligible.

(5) There is no limit to a queue/buffer for a machine.

(6) There is no priority among jobs.

The notation used for the stochastic expected makespan model include:

H_0: A large positive number used for a penalty factor.
ξp_{ijk}: The stochastic processing time of operation $o_{i,j}$ on machine k, subjected to independent normal distributions.
ξs_{ij}: The stochastic starting time of operation $o_{i,j}$.
ξC_{ij}: The stochastic completion time of operation $o_{i,j}$.
$\xi C_{\max}$: The stochastic makespan.

$$Y_{iji'j'k} = \begin{cases} 1, & \text{if operation } o_{i,j} \text{ precedes operation } o_{i',j'} \text{ on machine } k \\ 0, & \text{otherwise} \end{cases}$$

$1 \leq i \leq n,\ 1 \leq i' \leq n,\ 1 \leq k \leq m.$

$$a_{ijk,h} = \begin{cases} 1, & \text{if operation } o_{i,j} \text{ processed on machine } k \text{ is preferential to that on machine } h \\ 0, & \text{otherwise} \end{cases}$$

$$1 \leq i \leq n,\ 1 \leq k \leq m,\ 1 \leq h \leq m.$$

$$X_{ijk} = \begin{cases} 1, & \text{if operation } o_{i,j} \text{ is processed on machine } k \\ 0, & \text{otherwise.} \end{cases}$$

The makespan is the maximum completion time of the jobs and the objective is therefore to minimise the expected makespan. The mathematical model for the stochastic job shop scheduling problem can finally be formulated as

$$\xi C_{max} = \max_{1 \leq i \leq m} \{ \max_{1 \leq j \leq n} \ \xi C_{ij} \}$$

$$\text{Minimise } E(\xi C_{\max})$$

subject to

$$E(\xi C_{ij} - \sum_{k} (X_{ijk} \xi p_{ijk}) + H_0(1 - a_{ijk,h})) \geq E(\xi C_{ij}),$$
$$i = 1, 2, \ldots, n;\ h, k = 1, 2, \ldots, m, \quad (2.19)$$

$$E(\xi C_{ij} - \sum_{k} (X_{ijk} \xi p_{ijk}) + H_0(1 - Y_{iji'j'k})) \geq E(\xi C_{ij}),$$
$$k = 1, 2, \ldots, m;\ i, i' = 1, 2, \ldots, n, \quad (2.20)$$

$$\xi C_{ij} \geq 0,\ \forall (i, j). \quad (2.21)$$

Constraint (2.19) is a machine sequence constraint which ensures that operations of job j follow its machine list. Constraint (2.20) on the other hand, ensures that each machine processes no more than one job at any given time. Finally, constraint (2.21) ensures that there are no negative completion times for any jobs.

2.4.2 Defining real-time systems

Laplante and Ovaska (2011) define a *real-time system* as "one whose logical correctness is based on both the correctness of the outputs and their timeliness". The traditional model of a real-time system can be decomposed into a set of subsystems, *i.e.* the controlled object, the real-time computer system and the human operator, as observed in Figure 2.19. The real-time computer system must react to stimuli from the controlled object or the operator, within time intervals dictated by its environment.

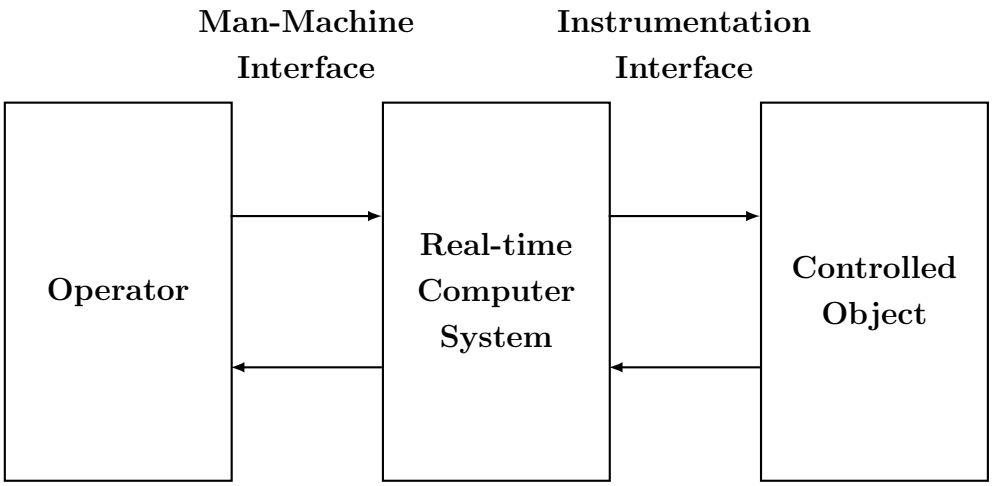


Figure 2.19: Representation of a traditional real-time system (Juvva, 1998).

Kopetz (2011) calls the instant at which a result must be produced a '*deadline*'. Buttazzo *et al.* (2005), Laplante and Ovaska (2011) and Kopetz (2011) refer to the different types of real-time systems as, *soft*, *firm*, and *hard*.

- *Soft real-time systems*: A soft real-time system is one in which the performance of a system is degraded but not destroyed by failure to meet response-time constraints.
- *Firm real-time systems*: A firm real-time system is one in which an arbitrary number of missed deadlines will not lead to total failure, but missing more than the permissible number of deadlines may lead to complete or catastrophic system failure.
- *Hard real-time systems*: A hard real-time system is one in which failure to meet even a single deadline may lead to complete or catastrophic system failure.

According to Laplante and Ovaska (2011), the study of *real-time systems* is a multi-dimensional subdiscipline of computer systems engineering that is strongly influenced by control theory, operations research, and software engineering. Figure 2.20 depicts several disciplines that affect the design and analysis of a *real-time system*.

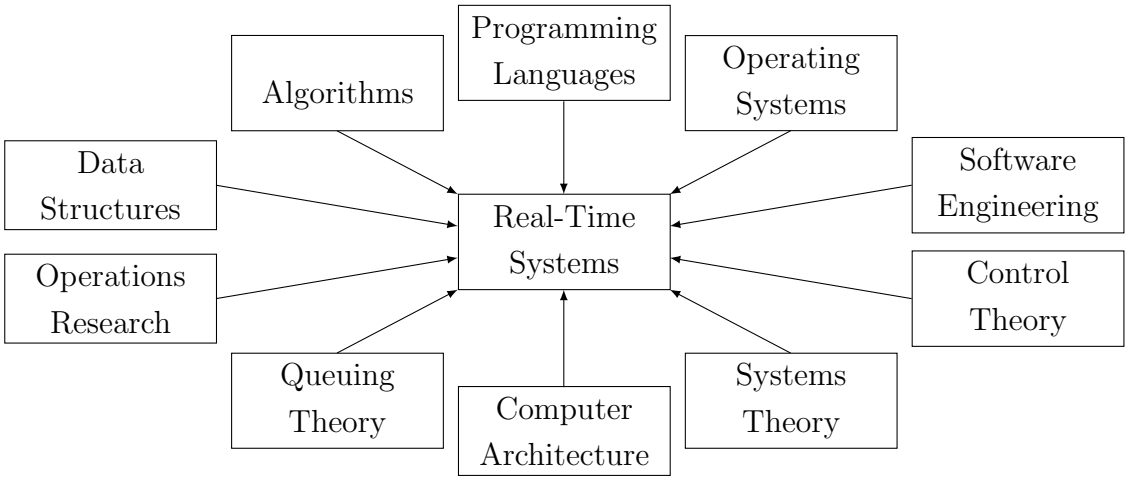


Figure 2.20: Disciplines that affect real-time systems engineering (Laplante and Ovaska, 2011).

2.4.3 Investigating real-time scheduling systems

Figure 2.21 presents the classification of real-time scheduling algorithms. Figure 2.21 illustrates that any real-time scheduling system can be classified as *dynamic* or *static*. Furthermore, Kopetz (2011) and Benítez-Pérez and Miguel (2005) refer to two types of schedulers, *i.e. dynamic* and *static*. A *dynamic* scheduler defines its scheduling decisions at run-time, selecting one out of the current set of *ready* tasks. Dynamic schedulers are flexible and adapt to an evolving task scenario. They consider only the *current* task requests. The run-time effort involved in finding a schedule can be substantial. A *static* scheduler defines its scheduling decisions during the off-line process. It generates a dispatching table for the run-time dispatcher off-line. For this purpose it needs complete prior knowledge about the task-set characteristics, *e.g.* maximum execution times, precedence constraints, mutual exclusion constraints, and deadlines. This dispatching table contains all information the dispatcher needs at run-time to decide at every point of a discrete time-base, which task is to be scheduled next. The run-time overhead of the dispatcher is small.

From Figure 2.21 it is observed that real-time scheduling algorithms can be further classified as *preemptive* and *non-preemptive* scheduling. Dhamdhere (2006) and Kopetz (2011) state that in *non-preemptive* scheduling, the currently executed task will not be interrupted until the task is completed. However, *preemptive* scheduling allows a higher priority task to replace a currently running task, even if the task is not completed (Ramesh, 2010).

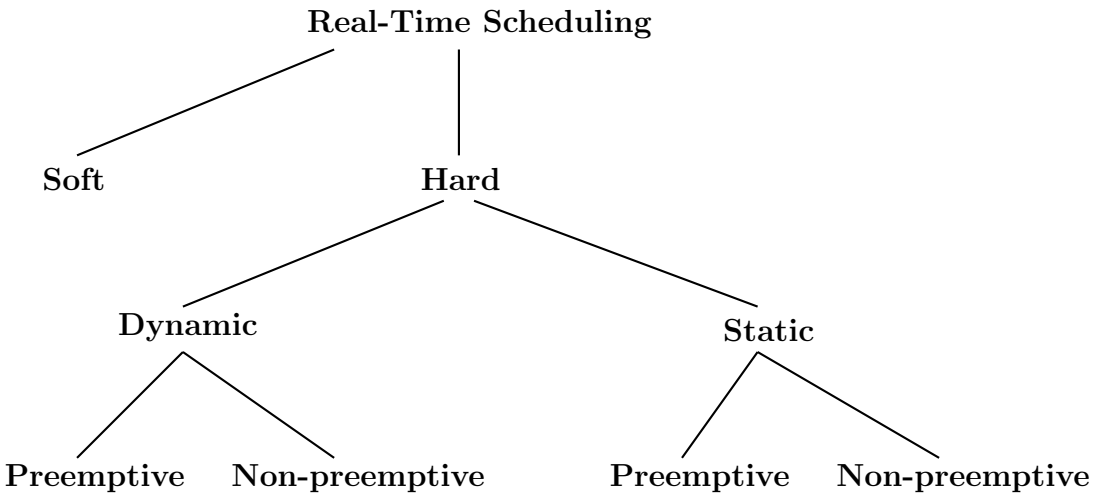


Figure 2.21: Classification of real-time scheduling algorithms (Kopetz, 2011).

2.4.4 Use of simulation as scheduling method

Kádár *et al.* (2004), Frantzén *et al.* (2011), and Banks *et al.* (2010) define simulation as a powerful tool used to design and analyse complex systems, where decisions are made about these systems. Furthermore, Banks *et al.* (2010) also refer to another purpose that simulation is appropriate for, *i.e.* the use of simulation for the observation of the effects that are caused due to informational, organisational, and environmental changes.

Simulation has many advantages and some drawbacks, as described by Bekker (2016a). The advantages include:

- System may be analysed before the implementation of a new procedure, avoiding and/or minimising cost.
- Simulation can provide optimal or near-optimal solutions to many problems.
- Simulation can be used for tactical and strategic planning.
- Changes to a system can be investigated without disrupting the operations of the system.
- Long processes are studied in a relatively short time.
- Critical parameters in a system can be identified and studied.
- Alternative solutions can be evaluated against each other.

The drawbacks of simulation include:

- The developer of a simulation model requires good training and experience.

- Simulation studies may be expensive in some cases.
- Simulation studies can be time-consuming if the simulation is properly executed.
- A simulation study could be too costly for a given problem.
- There are other analysis techniques that are more appropriate for certain problems.
- The interpretation of simulation results requires a sound statistical background.

Simulation modelling can be divided into two approaches due to the difference in how state changes occur. These state changes can either be 'discrete' or 'continuous'. According to Schriber *et al.* (2015), 'a discrete-event simulation is one in which the state of the model changes only at a discrete, but possibly random, set of simulated time points, called event times'. On the other hand, a continuous system simulation is applicable to systems where the variables are continuous in nature (Bandyopadhyay and Bhattacharya, 2014). The focus of the discussion will now change to simulation-based scheduling in reference to the topic of this thesis. Further information regarding the understanding of simulation can be found in Law (2007) and Banks (1998).

Herrmann (2006) and Sabuncuoglu and Bayız (2000) describe the use of simulation as part of the production control approach, where three uses of simulation are identified (*i.e.* simulation-based generation, refinement, and optimisation). Simulation-based schedule generation and refinement use simulation directly in order to determine initial schedules (generation) and to improve existing schedules (refinement). In addition, simulation-based optimisation applies simulation repeatedly in order to estimate a certain objective function value and improve it.

Simulation-based schedule generation refers to the use of simulation to generate an initial schedule, where dispatching rules, already part of the simulation engine, are used to allocate jobs to machines (Puente and Nemes, 2014). The sequencing of jobs observed in the simulation is then used to produce a schedule. According to Herrmann (2006), the components of a simulation based scheduling system include:

1. a simulation engine that contains the required dispatching rules for job selection,
2. a graphical user interface that produces Gantt charts based on the results from the simulation,
3. an interface to the information systems on the shop floor that develops a dispatch list from the Gantt charts.

The simulation model should however represent the base process and the base system in an appropriate manner.

Stamatis (2012) and Ward (2000) refer to schedule refinement as a 'necessary rework, redefinition, or modification of the schedule logic or data developed during planning'. During simulation-based schedule refinement, the schedule generated in the generation phase can be refined in order to achieve better estimates of system performance. This can be accomplished by evaluating the schedule with the performance measures of the system, as described in Section 2.1.1.

Finally, simulation-based schedule optimisation originates from the difficult situations faced to evaluate an objective function. Wang *et al.* (2011) states that optimisation of operation sequencing is typically performed using metaheuristics, *e.g.* genetic algorithms, tabu search or simulated annealing. These methods tend to be computationally costly, and therefore the level of detail for the simulation model is important in simulation-based optimisation (Herrmann, 2006). The simulation-based optimisation therefore also requires a simulation model.

There are many examples of simulation-based scheduling available in literature, over a long time span (Kim and Kim (1994), Bengü (1994), Calvet *et al.* (2016), Aurich *et al.* (2016)). Most recent work will be highlighted in this section.

Frazzon *et al.* (2018) proposed and applied a data-driven adaptive planning and control approach that uses simulation-based optimisation to determine the most suitable dispatching rules in real-time under varying conditions. The developed approach was evaluated considering the job shop production of a Brazilian producer of mechanical parts. Calvet *et al.* (2016) conducted a study that combined simulation with metaheuristics in distributed scheduling problems with stochastic processing times. The study focused on a scenario in which a company or a set of companies conforming to a supply network, had to deliver a complex product composed of several components to be processed on a set of parallel flow shops with a common deadline. The objectives that the study desired to minimise included the maximum completion time (makespan), as well as the accumulated deviations with respect to the deadline. A series of computational experiments were also performed in the study. In a study by Aurich *et al.* (2016), three approaches (*i.e.* an integrated simulation-based optimisation algorithm, SA and TS) are compared to solve a hybrid flow shop scheduling problem. The study concludes that all the approaches produce improvements in terms of producing more jobs on time while minimising the makespan, but the integrated simulation-based optimisation algorithm delivers the results much faster than the two metaheuristics. The two metaheuristics do however produce slightly better results than the integrated simulation-based optimisation algorithm in terms of total tardiness.

Thiesing and Pegden (2016) describe the application of the Simio simulation software package in scheduling. The study also refers to the powerful role simulation can play in

scheduling, by predicting and improving the short-term performance of a system. Birch *et al.* (2015) conducted a study where simulation and modelling were used for efficient and effective scheduling of offshore supply vessels. In a study by Walker *et al.* (2015), a simulation-based methodology was developed for planning the schedules of providers and the appointments of patients at outpatient clinics. The simulation model was used in the study to find the best balance between new and existing patients arriving at each appointment time period during the day.

Rose *et al.* (2015) performed a study, where a simulation-based segmentation procedure divides the flexible JSP into several small sub-problems, after which a branch and bound method is used to solve the sub-problems. In another study, Xu *et al.* (2015) considered a dynamic customer order scheduling problem in a stochastic setting. The objective of the study was to determine the optimal workload assignment policy that minimises the long-run expected order cycle time. A simulation-based GA was proposed to solve the problem. In a study by Bergmann *et al.* (2015), the suitability of various data mining and supervised machine learning methods were investigated, to emulate job scheduling decisions with data obtained from production data acquisition.

Kádár *et al.* (2004) indicated that features provided on the newer generation of simulation software facilitated the integration of simulation models with the production planning and scheduling systems. Furthermore, if the simulation system is combined with the production database of the enterprise, it is possible to instantly update the parameters in the model. This combination refers to a real-time system, that is able to update the simulation model in real time. The simulation model can be used parallel to the real manufacturing system to support and/or reinforce the decisions on the shop floor. Further literature regarding the use of simulation in scheduling systems, can be found on the website of the Winter Simulation Conference (`http://www.informs-sim.org/`).

2.5 Synthesis of literature review

This section provides a reflection of the literature that was discussed, as well as relating the literature to the thesis. Firstly, due to the popularity of the GA, SA, and TS algorithms for solving JSPs, the thesis will make use of one of these metaheuristics. The GA is a strong candidate, due to the literature that is converging to the use of the GA to solve JSPs, however this view might be distorted due to many researchers following each other and using the GA. Therefore, the desired metaheuristic will only be selected in the chapters to come. Furthermore, the thesis will first focus on single-objective optimisation, where each of the models minimising the performance indicators (as presented in Section 2.2) will

be considered. The models will however be adapted to use the stochastic variables ξp_{ijk}, ξs_{ij}, ξC_{ij}, and $\xi C_{\max}$. After the successful implementation of single-objective optimisation in the simulation scheduler, it will be expanded to the multi-objective domain, which is a more realistic representation of the real-world job shop environment.

Through the literature review, it was determined that there are three types of real-time systems, *i.e.* soft, firm and hard. The real-time system that will be created for this thesis, will be a hard real-time system. This was decided upon due to the hard real-time system's uncompromising nature, which ensures that the new schedule must be followed, otherwise the system will fail. As described in the environment of the JSP, the real-time system will use non-preemptive scheduling, where the currently executed task will not be interrupted until the task is completed. Lastly, discrete-event simulation will be used to model the system and to perform the rescheduling process. The reason for this can be attributed to the system interruptions, which change the state of the system, are only at discrete, and possibly random, simulated time points.

2.6 Chapter summary

This chapter started by investigating *job shop* scheduling, to identify appropriate scheduling methods as well as performance indicators for *job shops*. This was followed by the discussion of the mathematical models for minimising the performance indicators, as well as the methods for solving JSPs. Thereafter, stochastic scheduling was introduced and characteristics of such scheduling problems were defined. Finally, real-time scheduling systems and the use of simulation as a *real-time scheduling* method was investigated along with the basic principles. The succeeding chapter will describe the architectural design of the real-time scheduling system, and provide a description of the logical links between components of the system.

Chapter 3

Proposing the system with an architectural design

This chapter defines the architectural design process and concepts that are required for the successful development of an architecture. The chapter also makes reference to the framework required for the level of detail that should be considered by the developer in the development process.

3.1 Software architecture concepts

This section serves to introduce software architecture concepts essential for the successful development of large software systems. The main task of this thesis is the development of software to realise some of the objectives. Therefore, it is appropriate and essential that software architecture be studied and used prior to system development. The core information of this section is reproduced from Rozanski and Woods (2005), who provide the industry standards regarding architectural design of software systems. This section will discuss *stakeholders, viewpoints* and *perspectives*.

3.1.1 Stakeholders

The purpose for the development of software is dictated by the needs of the stakeholders of the software, and therefore it is necessary to clearly identify these stakeholders. Stakeholders are defined by Kessler and Sweitzer (2007) as "the people who affect the success of your software product, and are affected by it". Traditionally, software developers used similar definitions of stakeholders and identified end users and developers as the stakeholders of a software product. This interpretation of the definition is, however, too narrow if the software product is created to satisfy everyone affected by it. To ensure this, the software architect must clearly identify all stakeholders, understand their concerns, balance the potential conflicting priorities, and design an architecture that addresses their requirements as effectively as possible.

When selecting stakeholders, it is better to identify a wider stakeholder community, which will better the architect's chances of delivering a successful system. Stakeholder

attributes can assist the process of selecting stakeholders. Most literature regarding stakeholder attributes refers to the three attributes defined by Mitchell *et al.* (1997), which include:

- The stakeholder's *power* to influence the system.
- The *legitimacy* of the stakeholder's relationship to the system.
- The *urgency* of the stakeholder's claim on the system.

Furthermore, the software architect can also identify stakeholders by ensuring the selected stakeholders are *informed, committed, authorised* and *representative*. Informed stakeholders have all the required information, experience and understanding needed to make the correct decisions. Committed stakeholders are willing and able to participate in the process and they are prepared to make potentially difficult decisions. The decisions made by authorised stakeholders will not be reversed later on. Finally, representative stakeholders refers to stakeholders as a group rather than a person, and ensures that the chosen representatives are suitable.

Table 3.1 classifies stakeholders according to their roles and concerns. Most system development projects tend to include most if not all these types of stakeholders. There is, however, a trade-off when widening the set of stakeholders used in system development. The larger the stakeholder set, the more difficult it will be to reach a consensus.

Table 3.1: Stakeholder roles (Rozanski and Woods, 2005).

Stakeholder Class	Description
Acquirers	Oversee the procurement of the system or product
Assessors	Oversee the system's conformance to standards and legal regulation
Communicators	Explain the system to other stakeholders via its documentation and training materials
Developers	Construct and deploy the system from specifications (or lead the teams that do this)
Maintainers	Manage the evolution of the system once it is operational
Suppliers	Build and/or supply the hardware, software, or infrastructure on which the system will run
Support staff	Provide support to users for the product or system when it is running
System administrators	Run the system once it has been deployed
Testers	Test the system to ensure that it is suitable for use
Users	Define the system's functionality and ultimately make use of it

3.1.2 Viewpoints and views

A common temptation of software architect, when designing an architecture for their system, is to create a single, overloaded, comprehensive model. This, however, results in a model that is hard to understand and does not clearly identify the architecture's most important features. Therefore, according to Woods (2004), Hilliard *et al.* (2012) and Purhonen *et al.* (2004), a more appropriate method of designing an architecture is to separate the system into smaller subsets and represent the architecture via a number of related models (also known as *views*). Rozanski and Woods (2005) define the view as "a representation of one or more structural aspects of an architecture that illustrates how the architecture addresses one or more concerns held by one or more of its stakeholders".

The use of multiple views for describing an architecture was widely accepted when Krüchten (1995) published *Architectural Blueprints: The 4+1 View Model of Software Architecture*. The article provides a description of a software architecture making use of five views, each of which addresses a subset of concerns. The design decisions are captured in four views, while the fifth view is used to illustrate and validate the other four views.

The next challenge that software architects face, is to find the right level of detail to be designed into the views. If too much detail is provided, the audience may be overwhelmed, while too little detail may cause the audience to make assumptions that are not valid. A strategy to overcome this challenge, is to only include the details that further the objectives of the architectural description. This includes those details that help explain the architecture to stakeholders or demonstrate that stakeholder concerns are being met.

The development of views can however become tedious when the developer must use first principles each time a view is defined. To simplify and guide the creation of views, *architectural viewpoints* can be used. A viewpoint can be defined as a collection of patterns, templates, and conventions for constructing one type of view. This is done by defining the stakeholders, guidelines, principles, and template models for constructing its views.

3.1.2.1 Classification of viewpoints

Rozanski and Woods (2011) define seven core viewpoints for information systems, *i.e. context*, *functional*, *information*, *concurrency*, *development*, *deployment* and *operational* viewpoints. These core viewpoints can be grouped as shown in Figure 3.1.

The context, functional, information and concurrency viewpoints characterise the fundamental organisation of the system, after which the development viewpoint supports the construction of the system. Lastly, the deployment and operational viewpoints characterise the system once it enters its live environment. Each of the viewpoints can be described as follows:

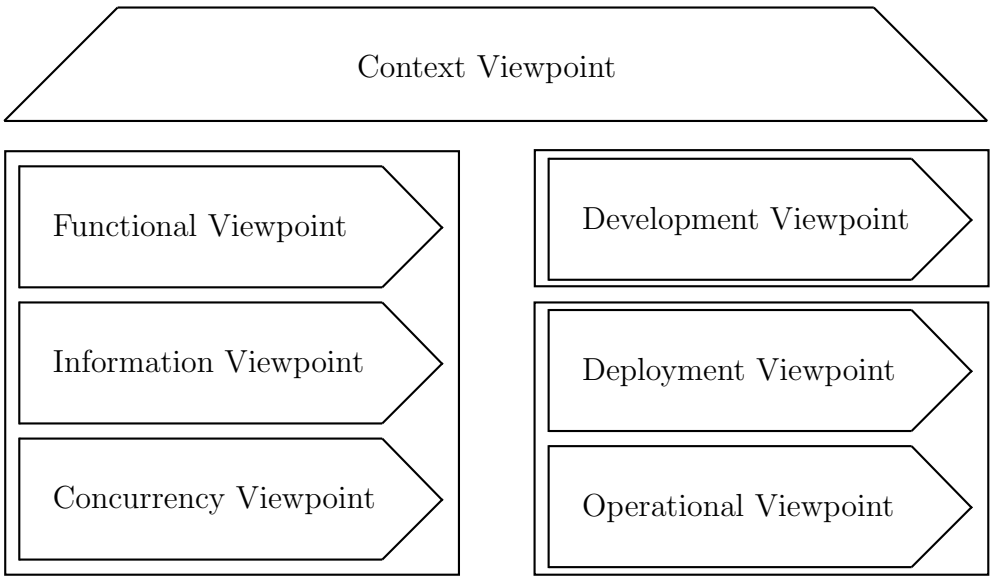


Figure 3.1: Classification of viewpoints (Rozanski and Woods, 2005).

Context viewpoint Describes the interaction, relationships and dependencies between the system and its environment. It includes the runtime context of the system, as well as its scope and requirements. The context viewpoint can be illustrated with the use of a context diagram.

Functional viewpoint Describes the functional elements and their responsibilities, interfaces and primary interactions of the system. This viewpoint is the cornerstone of the architectural description, which stakeholders tend to read first. It drives the other system structures (*i.e.* information structure, concurrency structure, *etc.*), while also having a significant impact on the quality properties of the system.

Information viewpoint Describes the way the architecture stores, manipulates, manages and distributes information. This viewpoint develops a high-level view of static data structure and information flow.

Concurrency viewpoint Describes the concurrency structure of the system, while also clearly identifying parts of the system that can execute concurrently and how this is coordinated and controlled.

Development viewpoint Describes the architecture of the system development process. The development views help to communicate the aspects of system development to the stakeholders involved in building, testing, maintaining and enhancing the system.

Deployment viewpoint Describes the environment into which the system will be deployed. This view captures the hardware environment that the system requires, the technical environment requirements for each element, and the mapping of software elements to the runtime environment that will execute them.

Operational viewpoint Describes how the system will be operated, administered, and supported once the system is in its live environment. The aim of the operational viewpoint is to identify system-wide strategies for addressing operational concerns of stakeholders, while also identifying solutions to address the concerns.

3.1.2.2 Benefits of views and viewpoints

Benefits of using views and viewpoints to describe the architecture of a system include:

- *Separation of concerns:* Describing many aspects of a system in a single overloaded model can cloud communication. Therefore, separating different models of a system into distinct descriptions helps the design, analysis, and communication processes by allowing one to focus on each aspect separately.

- *Communication with stakeholder groups:* The viewpoint-oriented approach can help guide stakeholder groups to different parts of the architectural description, based on their particular concerns. Each view can then be presented using language and notation appropriate to the knowledge, expertise and concerns of the group.
- *Management of complexity:* Creating a single overloaded model can lead to overwhelming complexity. Therefore, by separating each aspect of a system, the architect can focus on each aspect individually, resulting in less complexity.
- *Improved developer focus:* The architecture of a particular system is important for the developers of the system, because they use it as a foundation for the system design. By separating the system aspects into different views, the system developers can identify the important aspects and build the correct system.

3.1.2.3 Pitfalls when applying views and viewpoints

There are also pitfalls when using views and viewpoints, which do not allow an architect to solve all of the software architecture problems. These pitfalls include:

- *Inconsistency:* The use of multiple views to describe a system will inevitably lead to consistency problems. Achieving consistency throughout the architectural views can theoretically be made possible by using machine-checkable architecture description languages, but currently there are no such languages in widespread use. It is therefore a manual process that needs to be done by the software architect.
- *Selection of the wrong set of views:* It is not always evident which set of views is suitable for describing a particular system. Factors influencing this include the nature and complexity of the architecture, the skills and experience of the stakeholders and architect, and the available time to produce the architectural description.
- *Fragmentation:* When too many views are created, it can lead to the architecture being described by many independent models, each in a separate view, making it difficult to follow. To avoid fragmentation, all the views that do not address the important concerns of the system should be eliminated. Views can also be combined to form hybrid views.

3.1.3 Architectural perspectives

An architectural perspective can be defined as a collection of activities, tactics, and guidelines that are used to ensure that a system exhibits a particular set of quality properties. These properties may be required across several architectural views.

The perspectives are not used by themselves, but rather together with each architectural view to analyse and validate the qualities of the system's architecture and to drive further architectural decision-making. Table 3.2 contains quality properties that are addressed by the use of architectural perspectives.

Table 3.2: Quality properties addressed by architectural perspectives (Rozanski and Woods, 2005).

Quality properties	**Description**
Accessibility	The ability of the system to be used by people with disabilities
Availability	The ability of the system to be fully or partly operational when required, as well as to effectively handle failures affecting the system availability
Development resources	The ability of the system to be designed, build, deployed, and operated within the known constraints of people, budget, time, *etc.*
Evolution	The ability of the system to be flexible when change to the system eventually occur s
Internationalisation	The ability of the system to be independent from any language, country, or cultural group
Location	The ability of the system to overcome problems caused by the location of its elements and the distance between them
Performance	The ability of the system to predictably execute within the accepted performance profile and to handle increased processing volumes
Regulation	The ability of the system to conform to local and international laws, company policies, as well as other rules and standards
Security	The ability of the system to reliably control, monitor and audit who can perform actions on the system, as well as to detect and recover from failures in security mechanisms
Usability	The ability of the system to be easy to work with, and to allow people who interact with the system to work effectively

The key objective when applying the architectural perspectives to the architectural views, is to apply each relevant perspective to some or all of the views. This is done to ensure that each perspective's system-wide quality property concerns are addressed. In an

ideal scenario, each perspective can be applied to every view, but in reality, this scenario is not possible due to time constraints. Therefore, only some of the perspectives are usually applied to some of the views. Figure 3.2 illustrates how perspectives are applied to the views. Another illustration of the link between each perspective and the views is shown in Figure 3.3.

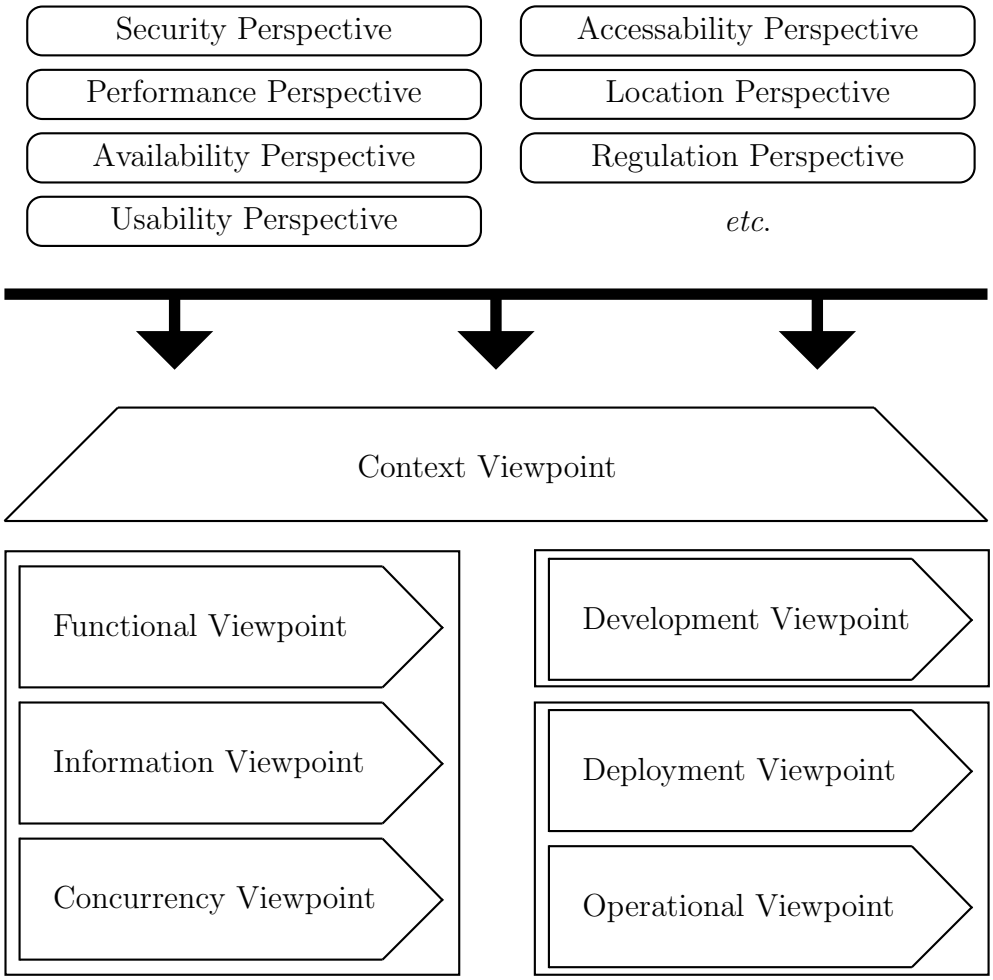


Figure 3.2: Applying architectural perspectives to architectural views (Rozanski and Woods, 2011).

Each rectangle in the grid of Figure 3.3 represents the application of a perspective to a view, and the contents of the rectangle define the important qualities and concerns of that intersection. The intersection between the security perspective and the information view, for example, guides the design of the architecture so that it includes appropriate data access control and data ownership. The intersection between the evolution perspective and the functional view guides the design of the architecture so that the types of changes that will be required are considered, and that the right level of flexibility is built in.

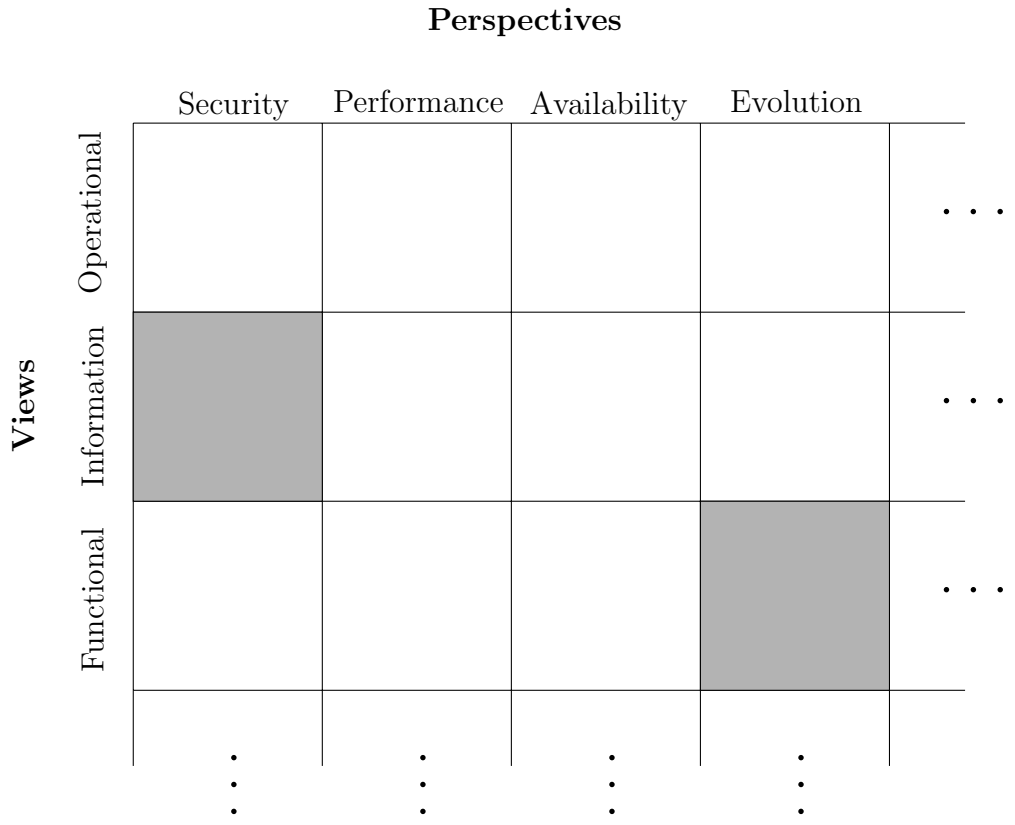


Figure 3.3: Examples of applying perspectives to views

3.1.3.1 Benefits of applying architectural perspectives

Benefits of applying architectural perspectives to views include:

- The perspectives provide common conventions, measurements, and a notation or language that describes the qualities of the system.
- The perspectives define the concerns that guide architectural decision making to help ensure that the resulting architecture will exhibit the quality properties considered by the specific perspective.
- The perspectives describe the method of validating the architecture to demonstrate that it meets the requirements across each of its views.
- The perspectives may offer solutions to common problems, thus helping to share knowledge between architects.
- The perspectives help the architect to work in a systematic way, to ensure that the concerns of the system are addressed.

3.1.3.2 Pitfalls when applying architectural perspectives

Potential pitfalls of applying architectural perspectives to views include:

- Each perspective addresses a single set of quality property concerns. This can therefore result in conflicting solutions suggested by different perspectives. It is the architect's responsibility to ensure that there is a balance between such competing needs.
- The stakeholder concerns and priorities are different for every system, so the degree to which the architect should consider each perspective varies considerably.
- The perspectives contain established and general advice for ensuring a system exhibits the correct quality properties. Every situation is different, and it is therefore important that the relevance of each perspective and its contents are considered and applied appropriately.

This concludes the introduction of software architecture concepts essential for the successful development of large software systems. The next section will describe the architecture definition process and highlight some development approaches used for software development.

3.2 Architecture definition in the software development methods

This section serves to describe how the architecture definition fits into the common approaches to designing and building systems. This section makes reference to three types of software development methods, *i.e. waterfall/linear* approaches, *iterative* approaches and *agile methods*.

3.2.1 Waterfall approaches

The waterfall approach views software development as a linear sequence of tasks, with each task using the outputs of the previous task as its inputs, as shown in Figure 3.4. An example would be the *requirement definition stage* provides the inputs to the *design stage*, which then provides the inputs to the build stage.

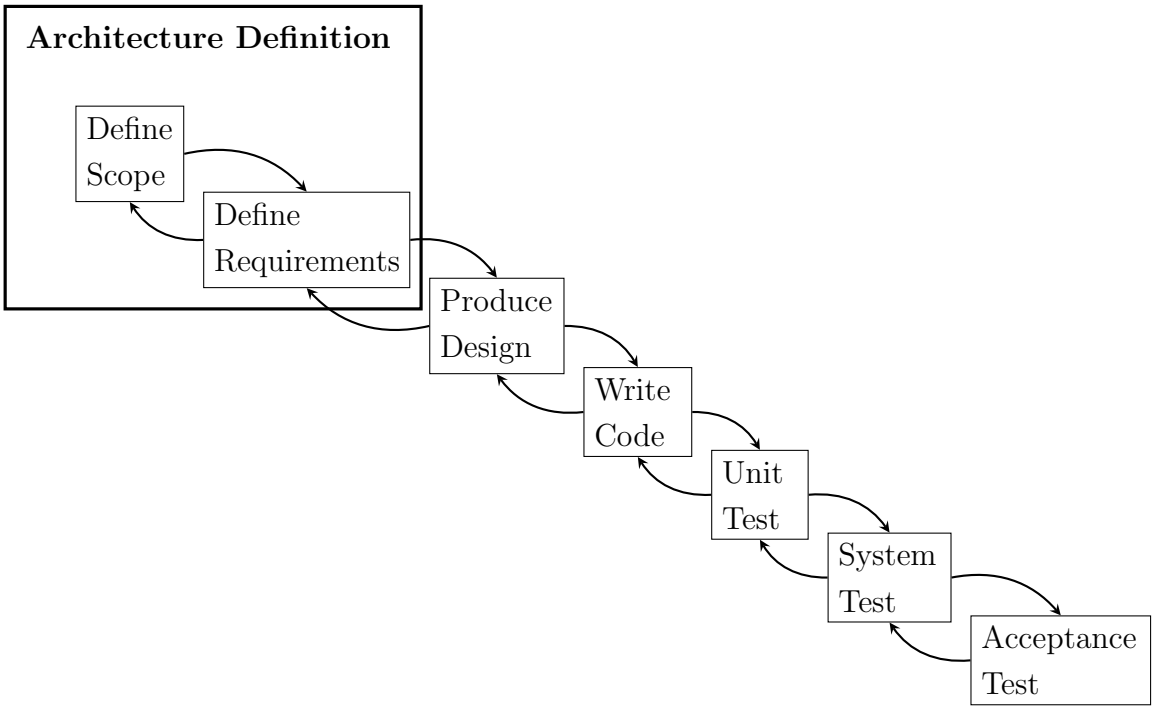


Figure 3.4: Waterfall Approach (Rozanski and Woods, 2005).

When changes are required, they feed back up to the preceding stage and possibly up the waterfall. This approach is however not ideal when designing and developing large systems. The architectural definition is integrated early in the linear approach, encompassing both the scope and requirement definitions, as shown in Figure 3.4.

3.2.2 Iterative approaches

Iterative approaches are implemented to reduce risk by means of early delivery of partial functionality, as shown in Figure 3.5. The classic iterative approach starts with an analysis phase, after which the design phase starts. After the design phase, it moves on to the build and test phase, and ends at the evaluation phase. After the evaluation phase, the current iteration is complete, and the next iteration starts at the analysis phase again.

Each iteration focuses on a single area that presents the highest risk due to its complex nature or unclear requirements. The architecture design would typically form part of the analysis phase of the approach, or it could alternatively run alongside other tasks as an ongoing activity.

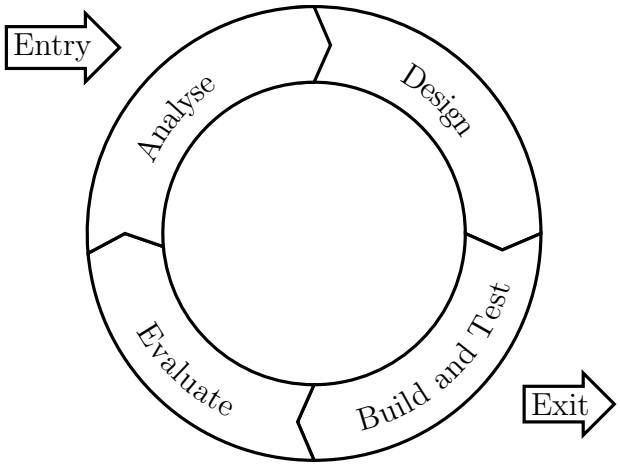


Figure 3.5: Iterative approach (Rozanski and Woods, 2005).

3.2.3 Agile methods

Agile methods are relatively recent developments in software engineering. These methods focus on rapid and continuous delivery of software to end users, encouraging constant interaction between customers and software developers, as well as attempting to minimise the management overheads of the development process. The technical practices of agile modelling help create simple and reliable software that can easily be changed. The architecture definition fits in above the construction increments, while setting the context for the iterations and helping to maintain the coherence and technical integrity across the system. Two of the best known agile methods are *Extreme Programming* and *Scrum*.

3.2.3.1 Extreme Programming

According to Lindstrom and Jeffries (2004) and Kendall and Kendall (2013), Extreme Programming is a discipline of software development based on four values, *i.e. simplicity*, *feedback*, *courage* and *communication*, as shown in Figure 3.6.

Simplicity: The first inclination of a developer, when working on a software development project, is to become overwhelmed by the complexity of the task. However, one cannot start with the development of complex tasks, without understanding the simple tasks first. Therefore, the simplicity value of software development requires the developer to start with the simplest thing that can be done.

Feedback: Feedback are test cases that compare the goal of the plan to the progress that has been made. It helps programmers make adjustments and lets the business

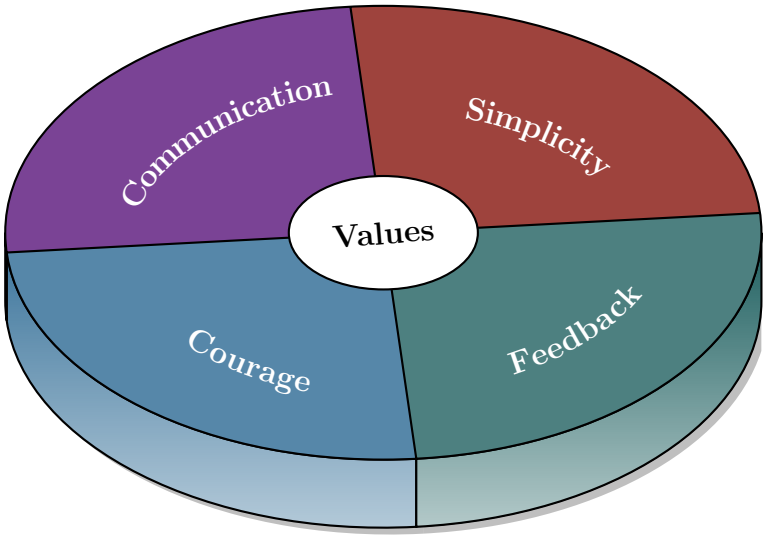


Figure 3.6: Values of Extreme Programming (Kendall and Kendall, 2013).

start experiencing what the functionality of the system will be like once it is fully functional.

Courage: Courage refers to the level of trust and comfort that must exist in a development team. This means that the whole team trusts each other and the customers enough to act in ways that will continuously improve what is being done on the project.

Communication: Communication is essential in a development team environment where the project requires constant updating and technical design. Practices like pair programming, task estimation, and unit testing rely heavily on good communication. When there is good communication within a team, problems are fixed rapidly and weak thinking is quickly strengthened.

Together with the values, there are certain principles that are essential to create the context for collaboration among developers and customers. The fundamental principles were first introduced by Beck (2000), and have evolved ever since. The principles are described as:

1. *Rapid feedback:*
   The time between an action and its feedback is critical to learning. Rapid feedback refers to getting feedback regarding the project, interpreting it, and implementing what was learned in the project as quickly as possible.

2. *Assume simplicity:*
   This principle states that each problem must be treated as if it can be solved with ridiculous simplicity. This is one of the hardest principles for developers to follow, due to the notion that a developer must plan for the future and design for reuse. However, Extreme Programming instructs developers to do a good job of solving today's problems and to trust their ability to add complexity in the future where it is needed.
3. *Incremental change:*
   It is difficult to solve a problem by making big changes all at once. This principle therefore instructs the developers to solve problems by making a series of small changes.
4. *Embracing change:*
   This principle states that the best strategy is the one that preserves the most options while solving the most pressing problem.
5. *Quality work:*
   Quality is one of the four project development variables (*i.e. scope*, *time*, *cost* and *quality*), and it is therefore essential to take it into consideration. To develop a high-quality product or system, the developing team is required to deliver quality work.

3.2.3.2 Scrum

Kendall and Kendall (2013) state that *scrum* is another agile approach, which originated from the starting position in rugby where rugby teams form a huddle and fight for possession of the ball. It is therefore centred around teamwork.

When applying the scrum approach, development teams start a project with a high-level plan that can be changed with ease as the development progresses. Each member of the team must realise that their individual success is secondary to the success of the project. Furthermore, the project leader has limited influence on the detail of the project, and the development team must work within a strict time frame. Some components of the scrum methodology can be described as:

1. Product backlog, in which a list is derived from product specifications.
2. Sprint backlog, a dynamically changing list of tasks to be completed in the next sprint.

3. Sprint, a 30-day period in which a development team transforms the backlog into software that can be demonstrated.
4. Daily scrum, a brief meeting in which communication has the highest priority. Team members must discuss what they completed since the last meeting, the obstacles they faced, and what they plan to do before the next scrum.
5. Demo, working software that can be demonstrated to the customer.

This concludes the discussion of common approaches for designing and building software systems, as well as how the architecture definition fits into these approaches. The next section will cover the scope, concerns, principles and constraints of the architecture definition.

3.3 Scope, concerns, principles, and constraints of the architecture definition

When the boundaries of a software development project are unknown, the development can become endless. A framework is therefore required to indicate through a high level of detail, what the developer should consider, which aspects to ignore and which aspects are irrelevant. Therefore, one of the earliest and most important tasks of the developer is to determine what the limits and constraints of the project are, and to clarify them with the stakeholders. To accomplish the stated task, the developer must incorporate:

- *Business goals and drivers:* the set of fundamental issues and problems that prompted the stakeholders to initiate the project.
- *Architectural scope:* a definition of what is included and what is excluded in the architecture.
- *Architectural concerns:* the requirements, objectives, intentions, and aspirations the stakeholders have for the architecture.
- *Architectural principles:* principles that shape, inform, or limit the architectural design choices.
- *Other architectural constraints:* standards, policies, and guidelines that also limit the project.

These components will now be discussed in more detail.

3.3.1 Business goals and drivers

The business goals and drivers set the context for the project and are the fundamental reason why the project exists. A business goal is a specific aim of the organisation, while a business driver is some force acting on the organisation that requires it to behave in a particular way, in order to protect and grow its business. It is essential to gain a good understanding of the business goals and drivers, otherwise if the software fails to meet these challenges, it would not have met the needs of its key stakeholders.

According to Clements and Bass (2010), business goal scenarios are required to ensure that business goals are expressed clearly, in a consistent fashion, and contain sufficient information. Furthermore, Clements & Bass define the six parts of a business goal scenario as:

- **Goal-subject:** This is the stakeholder that owns the goal. The stakeholder may be an individual, or an individual forming part of a specific organisation, or an organisation.
- **Goal-object:** This is the entity to which the goal applies or that will benefit from the successful completion of the goal.
- **Environment:** This is the context of the goal and acts as a rationale for the goal.
- **Goal:** This is the thing that needs to be accomplished.
- **Goal-measure:** This provides a measurement to determine whether the stated goal has been achieved.
- **Pedigree and value:** This indicates the degree of confidence in the goal, the goal's volatility, and the value of achieving the goal.

Figure 3.7 illustrates an example of a business goal scenario, and how each part fits into the scenario.

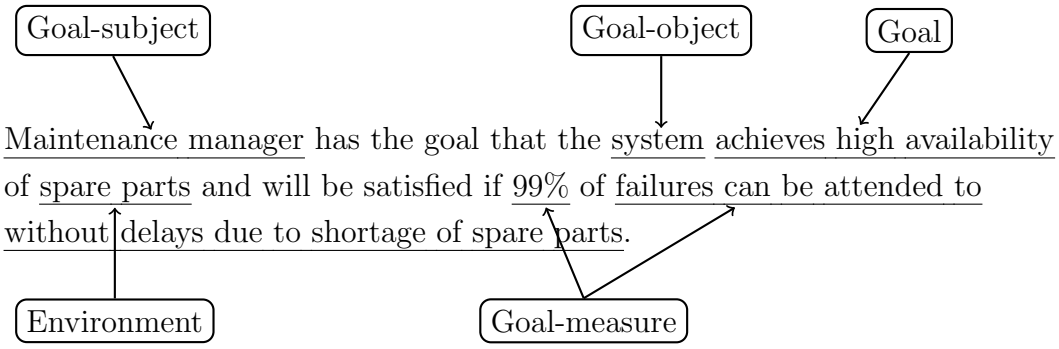


Figure 3.7: Business goal scenario example

3.3.2 Architectural scope

The definition of the system scope is an early milestone of any system development project. The scope must be defined based on the input from the stakeholders. Furthermore, the scope definition should contain information regarding the broad functional areas provided by the system, the external interfaces of the system and the external systems that communicate via these interfaces, decommissioning of a system, and any data to be migrated into the new system.

The scope definition forms the starting point of the architectural analysis and if the scope is not adequately defined, the project has a high probability of failure. An adequate tool for the scope definition process is the context diagram. The context diagram presents a high-level picture of the system boundaries and the adjacent external entities. Once the context diagram is created, it should be kept up-to-date with all the high-level changes.

3.3.3 Architectural concerns

As mentioned previously, the stakeholders of the software development project are the main source of information that shapes the architecture through their concerns. It is however not possible for an architect to be an expert in everything, and therefore the architect may be required to rely on specific business or technical expertise from subject matter experts, technology specialists or other architects. The architect must extract all the relevant information from these stakeholders, critically appraise it, consolidate it, and incorporate what is needed into the architectural definition.

The stakeholder concerns can be broadly classified as architectural goals, functional requirements, or architectural requirements. This information was already discussed in more detail in Sections 3.1.2 and 3.1.3, but will be briefly repeated. These classifications are described as:

1. **Architectural goals**

   Architectural goals refer to the vague concerns from the stakeholders. They tend to be expressed using imprecise language, due to stakeholders not being clear about what they mean. Architectural goals are unlikely to be measurable/quantifiable, and therefore there are no objective criteria for judging whether the goal has been met. Lastly, the goals have strong business focus, and it is often unclear how this might translate into an architectural solution.

   Architectural goals cannot be avoided, because they define the fundamental nature of the architecture and what must be achieved. Several tactics for dealing with these goals include trying to turn them into requirements by adding suitable objective

criteria, developing architectural principles that translate the goals into physical features and qualities of the architecture, or by managing the stakeholders' expectations of success, especially when goals are vague or unachievable.

2. **Functional requirements**
   These requirements help define what the system is required to do. It is unusual for architects to get involved in the specification of detailed functional requirements. This can be attributed to the scope statement provided to the architect, which should contain the main functional areas of the system. It is however uncommon for detailed functional requirements to significantly impact the architectural solution, therefore there is no need to capture any further detail for the purpose of the architectural definition.

3. **Architectural requirements**
   Architectural requirements, also referred to as non-functional requirements, do not directly mandate functionality but still have significant impact on the architecture. These requirements typically include system qualities such as performance, availability or scalability.

   There are several areas such as internationalisation, usability and accessibility, where there is little consensus as to how such requirements can be translated or expressed into system features. Determining the architectural requirements may require hard work and good communication between the architect and the stakeholders; however, it can be accomplished by means of architectural perspectives.

3.3.4 Architectural principles

An architectural principle can formally be defined as a fundamental statement of belief, approach, or intent that guides the definition of an architecture. It can refer to the current state or to a desired future state.

These principles can be used to establish a baseline or framework for the architecture definition. They also expose the underlying assumptions of the stakeholders and make them more explicit.

Characteristics of a good principle include:

- *Constructive:* Highlights issues, drives architectural decisions, and establishes the correct architectural framework.
- *Reasoned:* Strongly motivated by business drivers, goals, and other principles.

- *Well articulated:* Well understood by all stakeholders and not open to misinterpretation.
- *Testable:* Must be able to objectively test whether the principle is being adhered to.
- *Significant:* If the statement opposing the principle is still meaningful, then the principle is significant.

3.3.5 Other architectural constraints

Other architectural constraints include standards, guidelines, strategies and policies that may limit the architectural decisions.

Standards address either technology or business problems, and adopting standards eases the design and development process while also simplifying the integration process with other systems. It is essential that the compliance with standards can be tested in some way. Some standards are accompanied by test suites that will determine the compliance of a system, however if such tests are not available, the developers and testers need to be consulted to ensure that such tests are put in place.

Guidelines and strategies are less formal than standards and take the form of advice or recommended practice. It is not necessary to comply with the guidelines or strategies, and it is up to the architect and stakeholders to determine the need for compliance.

Policies start to define processes to be followed in order to meet stakeholder needs. It may be required to comply with pre-existing policies, or alternatively introduce new policies.

3.4 Chapter summary

This chapter defined the architectural design process and concepts that are required for the successful development of an architecture. The chapter also made reference to the framework required for the level of detail that should be considered by the developer in the development process. The succeeding chapter will describe the implementation and construction processes of the architecture of the real-time scheduling system.

Chapter 4

Implementation of architectural design

This chapter provides the implementation of the architectural design literature, where the appropriate stakeholders, viewpoints and perspective are selected. Furthermore, business goals and drivers, as well as the scope of the proposed system will be defined. Finally, a software development method will be chosen to guide the development of the proposed system.

4.1 Defining stakeholders

As mentioned in Section 3.1.1, stakeholders can be classified according to their roles and concerns. Table 3.1 refers to ten stakeholder classes that can be included in a development project. This section serves to identify and define the stakeholders that form part of the architectural definition.

First, the classes that are applicable to the architecture, as well as the stakeholders forming part of those classes will be discussed. Table 4.1 contains a list of applicable stakeholder classes.

As observed in Table 4.1, not all the stakeholder classes are incorporated into the proposed system's architectural definition process. The absent stakeholder classes include:

- Communicators: No training material will be created for the purpose of this research project, and therefore there is no need for a communicator stakeholder in the architectural definition process.
- Maintainers: The research will conclude when the proposed system is operational, and therefore the evolution of the system once it is operational does not form part of the research. Maintainers are not required for the purposes of this research.
- Suppliers: The proposed system will be created on the infrastructure of Stellenbosch University, but the research will conclude before the system is implemented. Therefore, no suppliers will be used for the purpose of the research.
- Support staff: The research will conclude before the proposed system is implemented, and therefore no support staff are required for the architectural definition process.

Table 4.1: Applicable stakeholders

Stakeholder class	Stakeholder	Description
Acquirers	Study leader	The study leader will oversee the procurement of the proposed system.
Assessors	Study leader	The study leader will oversee and assess the proposed systems conformance to standards.
Developers	The researcher of the study	The researcher is responsible for the development of the proposed system, from the specifications provided by the users.
System administrator	Study leader or a designated person	The study leader will serve as system administrator after the project is finished, and will keep the system active.
Testers	The researcher and study leader	Both the study leader and researcher are responsible for testing the proposed system, to ensure that it is suitable for use.
Users	Study leader and future students	The study leader will make use of the system, and serve as a proxy stakeholder. Future students will therefore gain access to the system through the study leader.

After the stakeholders are defined, the next step would be to describe each viewpoint applicable to the proposed system's architectural definition. The following section will include descriptions of the viewpoints, as well as the stakeholders and concerns linked to each viewpoint.

4.2 Describing viewpoints

A viewpoint is a collection of patterns, templates, and conventions for constructing one type of view. Of the seven core viewpoints defined by Rozanski and Woods (2011), only four are applicable to the proposed system's architectural definition. These four viewpoints include the context, functional, information and deployment viewpoints. The development

viewpoint is not required for the architecture of the proposed system due to there being only one developer, the researcher, while the operational viewpoint is not included because the research will conclude before the system is implemented. The concurrency viewpoint is also not necessary for the purposes of the research, due to there being only one user of the system, who will be the shop-floor manager.

As mentioned in Section 3.1.2.1, the context viewpoint describes the interaction, relationships, and dependencies between the system and its environment, and it can be illustrated by means of a context diagram. The context viewpoint will therefore be discussed in a section to follow, which describes the scope of the proposed system. Each of the three remaining viewpoints will now be discussed, referring to their concerns, pitfalls, models and required stakeholders.

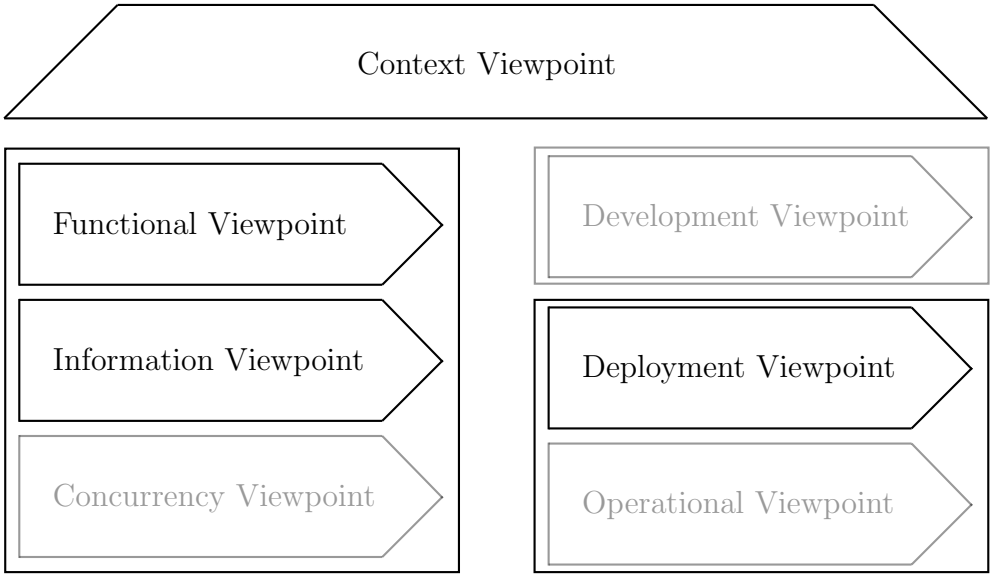


Figure 4.1: Applicable viewpoints

1. **Functional viewpoint:**

   Concerns: Stakeholder concerns regarding the functional viewpoint include the system's functional capabilities, external interfaces, internal structure, and design philosophy.

   Pitfalls: Pitfalls that frequently occur in the functional viewpoint include poorly defined interfaces, poorly understood responsibilities, an overloaded view, inappropriate level of detail, and too many dependencies.

   Models: A functional structure model will be used to clearly define the interfaces that are required for the proposed system, as well as to describe the functional capabilities of the system.

Stakeholders: The stakeholders that will possibly exhibit concerns regarding the functional viewpoint include the acquirers, assessors, developers, testers and users.

2. **Information viewpoint:**

   Concerns: Stakeholder concerns regarding the information viewpoint include the information structure and content, information flow, data ownership, timeliness, data quality, data volumes, as well as archives and data retention.

   Pitfalls: Problems and pitfalls that frequently occur in the information viewpoint include poor data quality, data incompatibilities, interface complexity, key matching deficiencies and inadequate volumetrics.

   Models: Static data structure models will be used to overcome the pitfalls of the viewpoint as well as the concerns of the stakeholders. The models will prevent poor data quality, data incompatibilities and key matching deficiencies by incorporating primary keys for each entry and by linking data entries through these keys. It is also assumed that the volume of data entering the system in a given time period will not exceed the capabilities of the system, which prevents inadequate volumetrics.

   Stakeholders: The stakeholders that will possibly exhibit concerns regarding the information viewpoint include the acquirers, assessors, developers and users.

3. **Deployment viewpoint:**

   Concerns: Types of hardware required, specification and quantity of hardware required, third-party software requirements, technology compatibility, network requirements, network capacity required and physical constraints.

   Pitfalls: Pitfalls of the deployment viewpoint include a lack of specialist technical knowledge and late consideration of the deployment environment.

   Models: Deployment structure models will be used to illustrate the structure of the prototype, as well as the technology capabilities and network requirements needed. While developing the prototype, the researcher will continuously consult experts in the applications that are used in the development process. Furthermore, the hardware required for the project is provided by the University.

   Stakeholders: The stakeholders that will possibly exhibit concerns regarding the deployment viewpoint include the assessors, developers and testers.

This concludes the discussion of the applicable viewpoints. The next section will identify the required perspectives and apply them to the selected viewpoints.

4.3 Applying perspectives to viewpoints

A perspective is a collection of activities, tactics, and guidelines that are used to ensure that a system exhibits a particular set of quality properties. Table 3.2 refers to ten different quality properties that can be addressed by using perspectives. Of these ten quality properties, only the availability, performance and usability properties are essential for the proposed system.

The quality properties that will not be addressed or incorporated into the architectural definition of the proposed system include:

- Accessibility: The proposed system will not be implemented as part of the research, and therefore no person with disabilities will access the system and the system will not be affected if the accessibility property is not met.
- Development resources: The proposed system will not reach the implementation or operational phases in this research, and there are no real constraints applicable to the designing and building phases of the proposed system.
- Evolution: The implementation of the proposed system does not form part of this research, and therefore no changes can occur. Flexibility to change is therefore not required by the system.
- Internationalisation: The initial design of the proposed system will be for the South African environment, and it is therefore assumed that English is sufficient for the language capabilities. English is considered to be a global language in business and education.
- Location: The proposed system will be designed to be on the cloud, and therefore there is no need to address the location property because there is no physical distance to overcome.
- Regulation: The proposed system will be designed to conform to company policies, but will not be severely affected by the inclusion of regulations, and therefore this property is not included in the architectural design.
- Security: The security concerns of the proposed system are limited, due to the lack of sensitive data being processed by the system.

When applying the chosen quality properties to the viewpoints, a grid resembling Figure 3.3, can be used. The new grid is illustrated by Figure 4.2, and the descriptions of the intersections between each viewpoint and perspective are given by the number at the specific intersection.

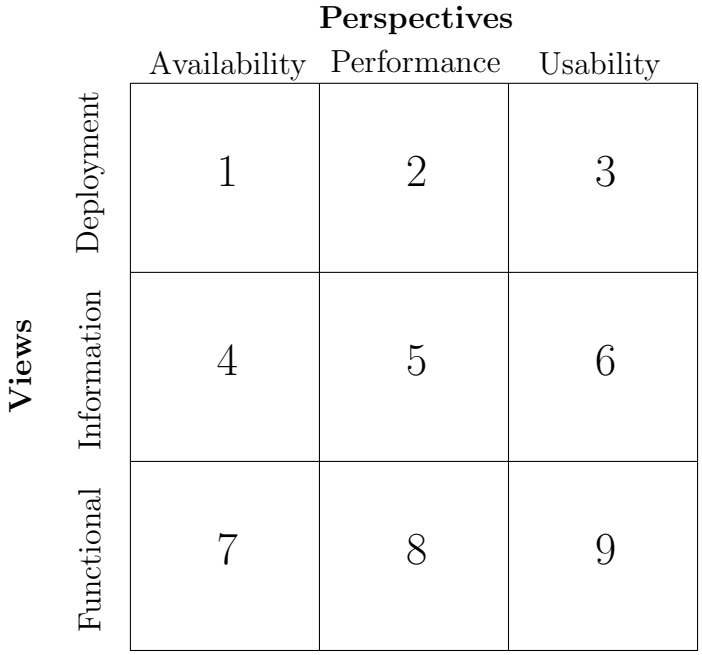


Figure 4.2: Applying the selected perspectives to viewpoints

Intersection 1: To ensure high availability of the proposed system, fault-tolerant hardware supplied by the University will be used. Sufficient network capacity will be used to ensure the availability of the system. The system will also make use of cloud-based servers that are maintained by the cloud-service provider, which will ensure that the system retains high availability.

Intersection 2: Proven simulation and database technologies will be used to ensure the system performs as intended. All these technologies must be compatible with one another to ensure the successful exchange of data as well as the performance of the system.

Intersection 3: N/A

Intersection 4: There will be processes in place that will ensure the backup of all of the system data. The data will be recorded on a database in the cloud for easy backup and recovery when required. It will also provide easy access for the user of the system.

Intersection 5: Shared resources, together with persistent storage will be used to ensure the retention of data, even when the connection and power is shut off from the storage device. The data will be retained in a database located in the cloud.

Intersection 6: Processes will be put in place to ensure the clear, accurate, complete and unbiased manner in which the information and results are presented to the user.

Furthermore, through discussion with the stakeholders, it will be determined what information is useful to the system's users, and only that information will be made available.

Intersection 7: The system will have an availability of 99%, which will be accomplished by using robust code that will mitigate the risk of the system failing, causing the system to be unavailable.

Intersection 8: The system runs a single code thread and execution is expected to be sufficiently fast. Scheduling and simulation may take longer, but code will be optimised where possible.

Intersection 9: Usability will be designed into the functional model, to ensure that the required interfaces are created for easy use of the system.

4.4 Identifying scenarios

Rozanski and Woods (2005) define a scenario as a crisp, concise description of a situation that a system is likely to face in its operational environment, along with a definition of the response required of the system. Furthermore, Rozanski and Woods identified the use of scenarios as one of the most powerful techniques that can be used to illustrate how a system will work in practice. The scenarios can be divided into two groups, *i.e. functional* scenarios and *system quality* scenarios.

- *Functional scenarios:* a sequence of external events that the system must respond to in a particular way.
- *System quality scenarios:* how the system should react to changes in its environment in order to exhibit one or more quality properties.

Next, expected functional and quality scenarios will be identified for the proposed system. Each functional scenario will include a scenario reference, overview, system state, system environment, external stimulus and a required system response; while every quality scenario will include a scenario reference, overview, system environment, environment changes and required system behaviour.

4.4.1 Functional scenarios

Following are functional scenarios identified for the proposed system.

Scenario Reference	Sc_1.1
Overview	The scenario refers to a new order entering the job shop.
System State	The job shop is running according to the current schedule, with the information system in an undisturbed state.
System Environment	System is in normal operation.
External Stimulus	The new order creates a disruption for the system and changes the state of the information system to *disturbed*.
Required System Response	When the information system enters a disturbed state, a notification must be triggered and sent to the floor manager. This notification should prompt a response from the manager to either initiate the rescheduling process or not.

Scenario Reference	Sc_1.2
Overview	The scenario refers to an operator calling in absent.
System State	All machines are operational and the job shop is capable of completing all the current jobs.
System Environment	System is in normal operation.
External Stimulus	An operator calls in absent.
Required System Response	The absent operator changes the state of the information system to *disturbed*, which triggers a notification that must be sent to the floor manager. This notification should prompt a response from the manager to either initiate the rescheduling process or not. Furthermore, the state of the machine used by this operator should change from *operational* to *operator absent*.

Scenario Reference	Sc_1.3
Overview	How the system deals with logging a failed machine.
System State	The job shop seems to be running according to the current schedule, with the information system in an undisturbed state.
System Environment	System is in normal operation.
External Stimulus	A machine breaks down which causes a disruption for the system and changes the state of the information system to *disturbed*.
Required System Response	When the information system enters a disturbed state, a notification must be triggered and sent to the floor manager. This notification should prompt a response from the manager to either initiate the rescheduling process or not. Furthermore, the state of the machine should change from *operational* to *broken*.

Scenario Reference	Sc_1.4
Overview	How the system deals with a new schedule that has been generated.
System State	The information system is in a disturbed state, and the manager must consider whether rescheduling is needed.
System Environment	System is in normal operation.
External Stimulus	The manager requests the system to generate a new schedule.
Required System Response	The cloud-based simulation model is prompted by the manager to reschedule, and consumes information from the information system. A new schedule is generated as the output of the rescheduling process, which can be viewed by the manager on the web application. The manager can then select a schedule to be implemented.

Scenario Reference	Sc_1.5
Overview	How the system deals with the acceptance of a new schedule.
System State	The information system is in a disturbed state, and the manager initiated the rescheduling process.
System Environment	System is in normal operation.
External Stimulus	The rescheduling process generated a new schedule, and the manager decided the new schedule should be implemented.
Required System Response	Firstly, the state of the information system will change from disturbed to undisturbed. Then the schedule that the sensored shop floor follows will be replaced by the newly generated schedule.

Scenario Reference	Sc_1.6
Overview	How the system deals with the rejection of a new schedule.
System State	The information system is in a disturbed state, and the manager initiated the rescheduling process.
System Environment	System is in normal operation.
External Stimulus	The rescheduling process generated a new schedule, and the manager decided the new schedule should not be implemented.
Required System Response	The state of the information system will change from disturbed to undisturbed. The newly generated schedule will have no effect on the current schedule that the sensored shop floor follows. The processing of jobs will continue as if no disruption occurred.

4.4.2 Quality scenarios

Following are quality scenarios identified for the proposed system.

Scenario Reference	Sc_2.1
Overview	The scenario where the connectivity of the system to the cloud is broken.
System Environment	The system is in normal operation.
Environment Changes	A disruption occurred in the factory; however, the system is not able to connect to the cloud-based information system to record the disruption.
Required System Behaviour	The system should continuously ping the cloud-based information system until the connection is reestablished.

Scenario Reference	Sc_2.2
Overview	The scenario where the optimisation of the schedule takes more than an hour.
System Environment	The system is in normal operation.
Environment Changes	The rescheduling process was initiated by the manager but the process takes more than an hour to complete.
Required System Behaviour	The production in the factory should continue, and therefore the current schedule should be used. After an hour, reject the new schedule due to it being too far behind real time.

Scenario Reference	Sc_2.3
Overview	The scenario where a sensor at a machine fails.
System Environment	The system is in normal operation.
Environment Changes	A sensor on one of the machines failed, and therefore the cloud-based information system will not receive any real-time data from the sensor.
Required System Behaviour	An information buffer should be put in place to ensure that data can be captured manually while the sensor is being replaced.

Scenario Reference	Sc_2.4
Overview	The scenario where the configuration of the factory changes, whereby resources are added or removed.
System Environment	The system is in normal operation.
Environment Changes	The configuration of the factory changes when resources are either added or removed.
Required System Behaviour	The new resources should be added to the resource list in the information system, while any removed resources should have their status changed to *inactive*.

Scenario Reference	Sc_2.5
Overview	The scenario where the new schedules that are generated are approximately tied in terms of the optimisation of different objectives.
System Environment	The system is in normal operation.
Environment Changes	During the rescheduling process, several different schedules are generated by optimising different objectives. Some of these schedules are approximately tied and it is difficult to determine which schedule to implement.
Required System Behaviour	To overcome the difficulty, a default objective is specified *i.e.* the minimisation of the average lateness.

This concludes the definition of functional and quality scenarios. The following section serves to identify and define business goals and drivers.

4.5 Defining business goals and drivers

Although the system developed in this study is primarily for research purposes, the researcher and study leader anticipated some business goals and drivers. These could serve as motivation for a commercial implementation. A business goal is *a specific aim the organisation has*, while a business driver is *some force acting on the organisation*. This driver *requires the organisation to respond or behave in certain way so that the organisation is protected and can grow* (Rozanski and Woods, 2011).

4.5.1 Business goals

System disturbances, *e.g.* a machine fails, or a new order is received, may result in inefficient use of available production time. Processing orders in a first-come-first-served manner may not be the best *modus operandi*, since it might be better to process a newly received order immediately. (This will be indicated by performance measures). Business goals are thus:

1. Minimise downtime due to system disturbances.
2. Maximise utilisation of available production time.
3. Acknowledge existence of suitable technologies and exploit these to increase profit.

4.5.2 Business drivers

The modern industrial world offers many technologies, including cloud services, mobile telecommunication, sophisticated simulation software and well-researched, proven scheduling techniques. A job shop business should therefore embrace these technologies and knowledge to keep up with its competition, and to improve and grow the business. Doing more work in a shorter time will raise the net profit. Business drivers are:

1. Technology developments beyond that which is primarily required for operations (machines). These developments include mobile communication technology, cloud services and simulation software.
2. Increasing computing capability and developments of scheduling techniques.
3. Real-time scheduling is almost a reality and must be utilised to continuously get the best performance from the system.

4.6 Proposed system scope

The scope of a new system should define the different components and functional areas within it, as well as the external systems that communicate via external interfaces. The logical links and relationship between each component should also be defined in the scope. According to Dori (2002), the task of engineering a new system has become more complicated in recent years, due to the increasing number of components involved. Another problem that organisations experience, is a lack of a common product development vocabulary or modelling framework.

Object Process Methodology (OPM) provides a modelling framework that contains a clear and concise set of symbols that translates into a language that expresses a system's building blocks and how they relate to each other. The set of symbols and links are given in "Object-Process Methodology: A Holistic Systems Paradigm" by Dori, as well as in Table A.1. The building blocks that are defined by the OPM are presented in Table 4.2.

Table 4.2: The building blocks of OPM (Dori, 2002).

	Visual Representation	Definition	Description
Entities	Object	It has the potential of stable, unconditional physical or mental existence.	Static things. Can only be changed by processes.
	Process(ing)	It is a pattern of transformation that an object undergoes.	Dynamic things. Are recognisable by the changes they cause to objects.
	Object state	It is a situation an object can be at.	They describe objects. They are attributes of objects. Processes can change an object's state.

OPM can be used to create a symbolic representation of the objects in a system, their structural relations as well as the functions they enable. Furthermore, the methodology has the ability to represent the proposed system simultaneously in a graphical representation and a natural language. Coupled with the capabilities of the OPM, it also forms part of the ISO standards. ISO 19450:2015 specifies the use of "the concepts, semantics, and syntax

of OPM as a modelling paradigm and language for producing conceptual models at various extents of detail" (ISO, 2015).

A scope/context diagram was developed by Snyman and Bekker (2017) which made use of the OPM to describe the proposed system. The diagram is illustrated graphically in Figure 4.3, followed by the corresponding semantic description.

The context diagram incorporates three uncertainties that initiate the rescheduling process, *i.e.* the logging of a new order, a machine that breaks down, and an absent operator. The process of logging a new order is handled by the user, which changes the state of the information system. The movement of jobs through the shop is detected by the sensor. When a job arrives at a machine or the machine finishes the processing of a job, the sensor detects it and registers the movement in the information system. Lastly, when the process that reports the absence of an operator is initiated, the state of the machine used by the specific operator, changes to *absent operator*. All three uncertainties change the state of the information system from *undisturbed* to *disturbed*.

When the disturbed state is entered, a notification of a disruption is triggered and sent to the floor manager. The manager then handles the notification by either selecting to reschedule, whereby a rescheduling process is initiated, or selecting not to reschedule. The manager will initiate the rescheduling process by logging on to a remote desktop control application that connects to the cloud server, after which the manager can instruct the simulation model to start. The rescheduling process consumes information from the integrated information system, and enables a cloud-based simulation model to generate proposed schedules. A record of the performance of each proposed schedule is kept, from which a detailed report is generated and saved. The new schedules generated by the simulation model can either be acceptable or non-acceptable. When acceptable, the schedules can be viewed by the manager in the form of the generated report on the web application where the manager can initiate the system updating process. The system updating process then changes the state of the information system to undisturbed and updates the sequence of the jobs to be processed.

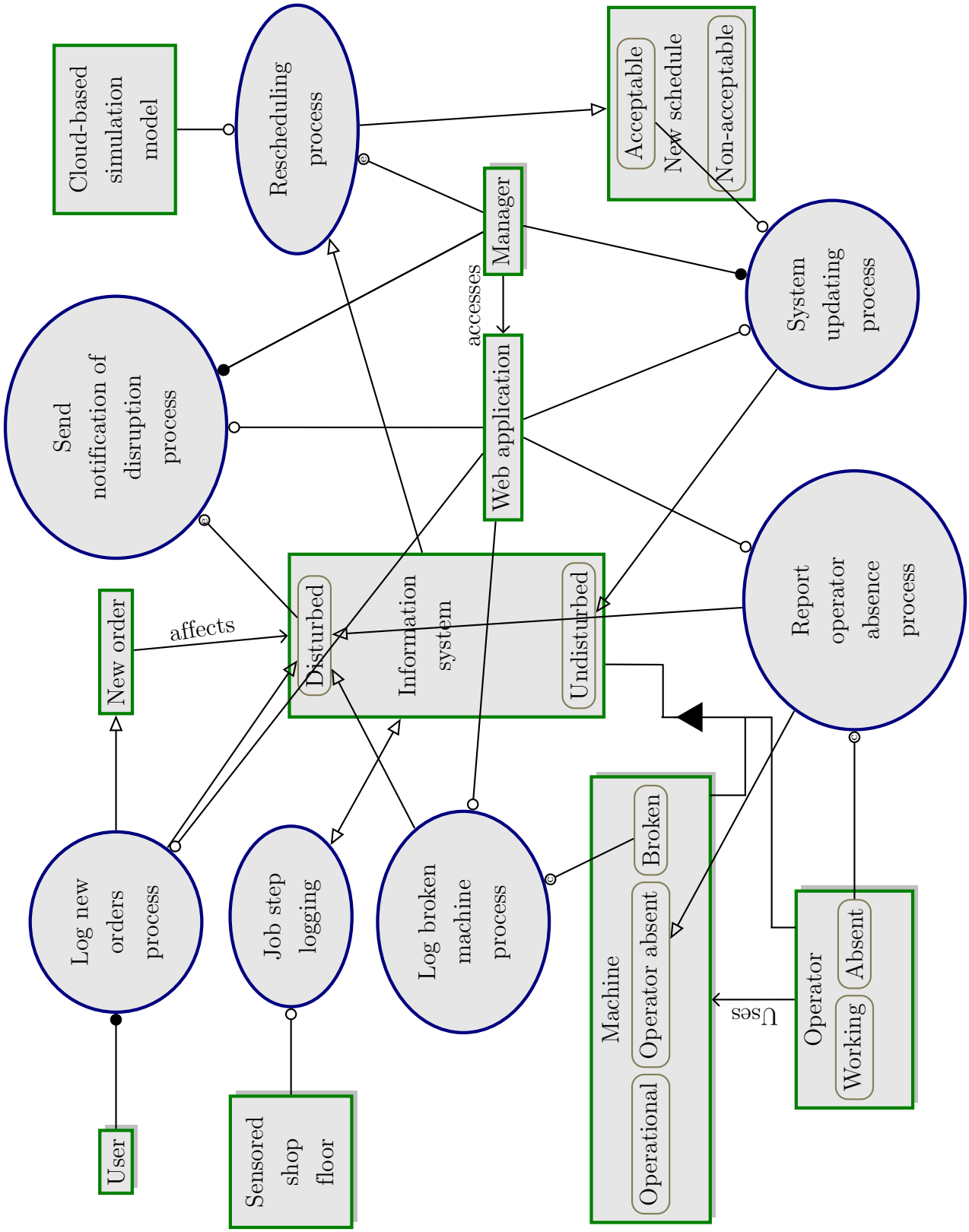


Figure 4.3: Top-level architecture of the proposed software solution (revised from Snyman and Bekker (2017)).

The semantic representation of the system diagram derived from Figure 4.3, according to the OPM, is as follows:

User is physical.
User handles Log new orders process.
New order affects Information system.
Information system can be Undisturbed by default or Disturbed.
Information system consists of many Machines and many Operators.
    Machine is physical.
    Machine can be Operational by default, Broken, or Operator absent.
    Operator is physical.
    Operator can be Working by default or Absent.
    Operator uses Machine.
Information system triggers Sending notification of disruption process when it enters Disturbed.
Manager is physical.
Manager accesses Web application.
Manager handles System updating process and Sending notification of disruption process.
Manager triggers Rescheduling process.
New schedule can be Acceptable or Non-acceptable.
Sensored shop-floor is physical.
Job step logging requires Sensored shop-floor.
Job step logging affects Information system.
Log new orders process requires Web application.
Log new orders process yields Disturbed Information system and New order.
Log broken machine process occurs if Machine is Broken.
Log broken machine process requires Web application.
Log broken machine process yields Disturbed Information system.
Report operator absence process occurs if Operator is Absent.
Report operator absence process requires Web application.
Report operator absence process yields Disturbed Information system and Operator absent Machine.
Rescheduling process requires Cloud-based simulation model and Manager.
Rescheduling process consumes Information system.
Rescheduling process yields New schedule.
Sending notification of disruption process requires Web application and Disturbed Information system.
System updating process requires Acceptable New schedule and Web application.
System updating process yields Undisturbed Information system.

The sensorised shop floor will contain eight machines (*i.e.* two turning, two grinding, two milling, one gear, and one induction hardening machine), and there is an operator for each individual machine. The system will also make use of a database, as well as a simulation model of the shop floor that are in the cloud. It is also assumed that all orders can be processed on the available resources and that there is no need for outsourcing of jobs. Only a moderate volume of orders is expected per time period, which means that the workload of the job shop will not be unmanageable. Lastly, in real-world environments, there is a difference between when an operator is absent and when an operator is on leave. For the purposes of this system, the scenario where an operator is on leave is equivalent to an operator being absent for more than one day.

4.7 Selection of software development method

For the purpose of the research, a combination of the iterative approach and the Extreme Programming agile method will be used as the software development method. These life cycles were selected because they can be used by a single developer that does not form part of a team, and the iterative nature will help the system to be improved upon continuously through the development phase of the research. For further detail regarding the chosen two methods of software development, the reader is referred to Section 3.2.

The proposed system will not be developed for commercial use after the research is completed, but will rather be used as a means of implementing, testing and learning within the field of software development. This purpose therefore also confirms the use of the iterative approach as a software development method.

4.8 Chapter summary

This chapter provided a description of the implementation of the architectural design literature, where the appropriate stakeholders, viewpoints and perspective were selected. Business goals and drivers were defined, as well as the scope of the proposed system. Finally, a software development method was chosen to guide the development of the proposed system. The subsequent chapter will discuss the development of the information system, simulation model and sensors.

Chapter 5

Development of proposed system

This chapter documents the development process of the proposed system. The components that were developed for the system include an information system where all required data will be stored, the web pages used for logging data changes, a simulation model that mimics the behaviour of a job shop, and finally, sensors for logging real-time data. Figure 5.1 illustrates all these components and how they interact with each other. The components will also be discussed in this chapter.

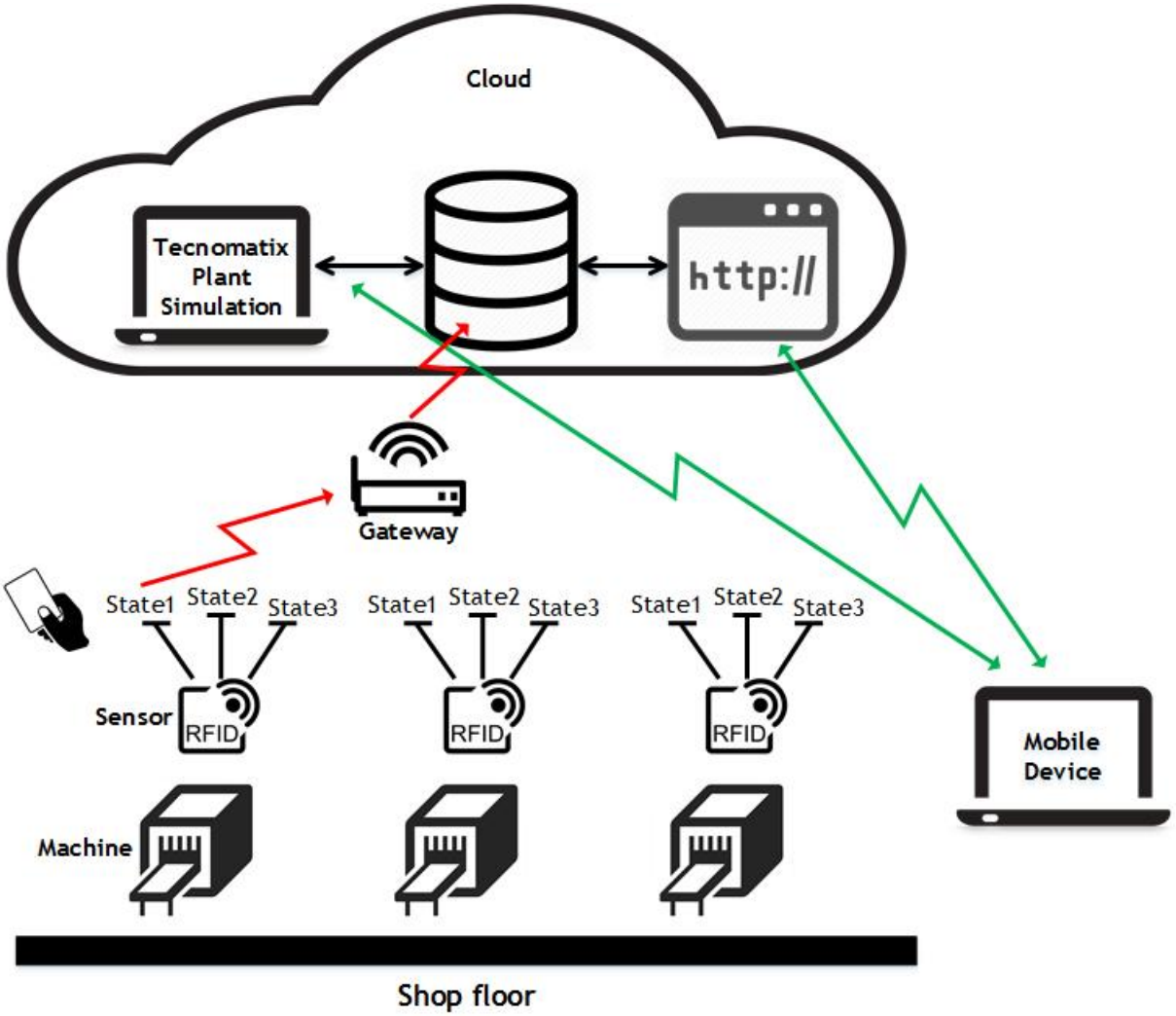


Figure 5.1: Components of the proposed system

5.1 Development of the information system and web pages

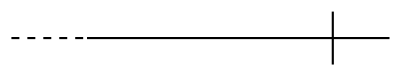
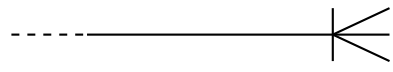
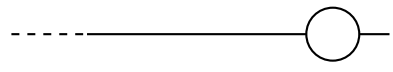
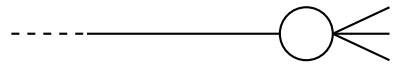
This section focuses on the development of the proposed information system, by constructing data dictionaries of all the data tables required as well as the EERD that illustrates the relationship between each table. The section also includes the development of the web pages that will be used to log data onto the information system.

5.1.1 Construction of information system

Due to the system requirement which prescribes the system to be cloud-based, the first step was to set up a cloud-computing account. To accomplish this, several cloud-computing products needed to be considered. These products include *Google Cloud Platform* (GCP), *Microsoft Azure*, *Amazon Elastic Compute Cloud* and *IBM Cloud*, which are some of the larger and better-known cloud-computing products. For the purposes of this project, the researcher decided to use GCP which provided the most functionality for a free account, compared to the other products.

After the selection of the cloud service provider, the second step was to set up an information system. Due to GCP making use of MySQL®[1] as the information system development tool, a MySQL server was created. An extended entity-relationship diagram (EERD) could then be constructed which illustrates the relationships between different entities within a system. These may be people, places, events or things. For the purposes of this thesis, the crow's foot notation, as defined by Martin (1990) and described in Table 5.1, was used to construct the EERD.

Table 5.1: Revised crow's foot notation

Symbol	**Meaning**
	One (Mandatory) :1
	Many (Mandatory) :1 . . . N
	One (Optional) :0 . . . 1
	Many (Optional) :0 . . . N

[1]The registered trademark for MySQL® will from now on be omitted.

For the construction of the EERD, all the tables that are required for storing data have to be defined, keeping in mind what data will be stored. For the proposed system, it is necessary to store data about:

People: Operators,

Things: Machines,
Jobs,
Operations,
Standard auxiliary data, *e.g.* Statuses.

The description and purpose of the tables that were included in the diagram will now be provided.

Table 5.2: Definition of `tblMachineStatus`

Purpose: `tblMachineStatus` is used to store the different states that machines can be in.

Attribute	**Description**	**Data type**
MachineStatus_ID	Unique primary key to identify a specific machine status.	Int(11)
MachineStatus	The description of the specific machine status.	Varchar(45)

Table 5.3: Definition of `tblMachines`

Purpose: `tblMachines` is used to store the different machines that are currently, or were previously, in the job shop.

Attribute	**Description**	**Data type**
Machine_ID	Unique primary key to identify a specific machine.	Int(11)
MachineName	The name of the specific machine.	Varchar(45)
MachineStatus_IDFK	Foreign key that specifies the status of the machine.	Int(11)

Table 5.4: Definition of `tblOpStatus`

Purpose: `tblOpStatus` is used to store the different states that operators can be in.

Attribute	Description	Data type
OpStatus_ID	Unique primary key to identify a specific operator status.	Int(11)
MachineStatus	The description of the specific operator status.	Varchar(45)

Table 5.5: Definition of `tblOperators`

Purpose: `tblOperators` is used to store the information of all the operators that are currently working, or have previously worked, at the job shop.

Attribute	Description	Data type
Operator_ID	Unique primary key to identify a specific operator.	Int(11)
OperatorName	The name of the specific operator.	Varchar(45)
OpStatus_IDFK	Foreign key that specifies the status of the operator.	Int(11)

Table 5.6: Definition of `tblStdOps`

Purpose: `tblStdOps` is used to store the different standard operations that can be performed in the job shop as well as their respective costs.

Attribute	Description	Data type
StdOps_ID	Unique primary key to identify a standard operation.	Int(11)
StdOpsName	The description of the specific standard operation.	Varchar(45)
StdOpsCost	The cost associated with the specific standard operation.	Decimal(5,2)

Table 5.7: Definition of `tblJobs`

Purpose: `tblJobs` is used to store the information of all the jobs that have entered the job shop.

Attribute	Description	Data type
Job_ID	Unique primary key to identify a specific job.	Int(11)
JobName	The name identifier which is used throughout the thesis as "J_{number}" (*e.g.* "J_1").	Varchar(45)
JobDate	Date that a job enters the system.	Datetime
JobDescription	The name of the specific part that needs to be processed (*e.g.*"Gear").	Varchar(45)
JobPriority	How important the job is. This attribute is used in the most-important-job-first dispatching rule.	Int(11)
JobQuantity	The total number of parts that need to be produced.	Int(11)
JobDueDate	When should the job be completed. This attribute is used in the earliest due date and critical ratio dispatching rules.	Date
Job_RFID	RFID card number that is linked to the specific job.	Int(11)

Table 5.8: Definition of `tblOperationStatus`

Purpose: `tblOperationStatus` is used to store the different states that the operations have to go through while in the job shop.

Attribute	Description	Data type
OperationStatus _ID	Unique primary key to identify a specific operation status.	Int(11)
OperationStatus	The description of the specific operation status.	Varchar(45)

Table 5.9: Definition of `tblOperations`

Purpose: `tblOperations` is used to store the information of all the operations that are linked to specific jobs that have entered the job shop.

Attribute	**Description**	**Data type**
Operation_ID	Unique primary key to identify a specific operation.	Int(11)
Job_IDFK	Foreign key that links a specific job to the operation.	Int(11)
OperationNumber	Indicates the operation number of the linked job.	Int(11)
Machine_IDFK	Foreign key that specifies a specific machine used to process the operation.	Int(11)
StdOps_IDFK	Foreign key that links a specific standard operation cost to the operation.	Int(11)
ExpectedProcessing Time	The expected time required to complete the processing of the operation.	Time
EarliestStartTime	The earliest time the operation can begin.	Datetime
LatestStartTime	The latest time the operation can begin while still ensuring that the job is completed on time.	Datetime
OperationStatus _IDFK	Foreign key that specifies the status of the operation.	Int(11)

Table 5.10: Definition of `tblMachineOperator`

Purpose: `tblMachineOperator` is used to store the link between an operator and a machine.

Attribute	**Description**	**Data type**
MO_ID	Unique primary key to identify a specific machine and operator connection.	Int(11)
Machine_IDFK	Foreign key that specifies a specific machine.	Int(11)
Operator_IDFK	Foreign key that specifies a specific operator.	Int(11)
MODate	Date that the connection between the machine and operator was made.	Datetime

After each table was defined, the final EERD could be constructed to illustrate the relationships between the different tables. Figure 5.2 illustrates the EERD for the proposed information system.

The tables that do not have any relationship with other tables are either reference tables or auxiliary tables. The auxiliary tables are used to store the schedules of all the dispatching rules, while the schedule selected by the manager is stored in the reference tables. The reference tables indicating what jobs to process next will be visible to the operators.

With the completion of the EERD, the data dictionaries need to be constructed on the MySQL server on the cloud. To accomplish this, MySQL Workbench was downloaded in order to access the server. When logged onto the server, each table could be created separately, after which all the foreign keys could be linked to the primary keys.

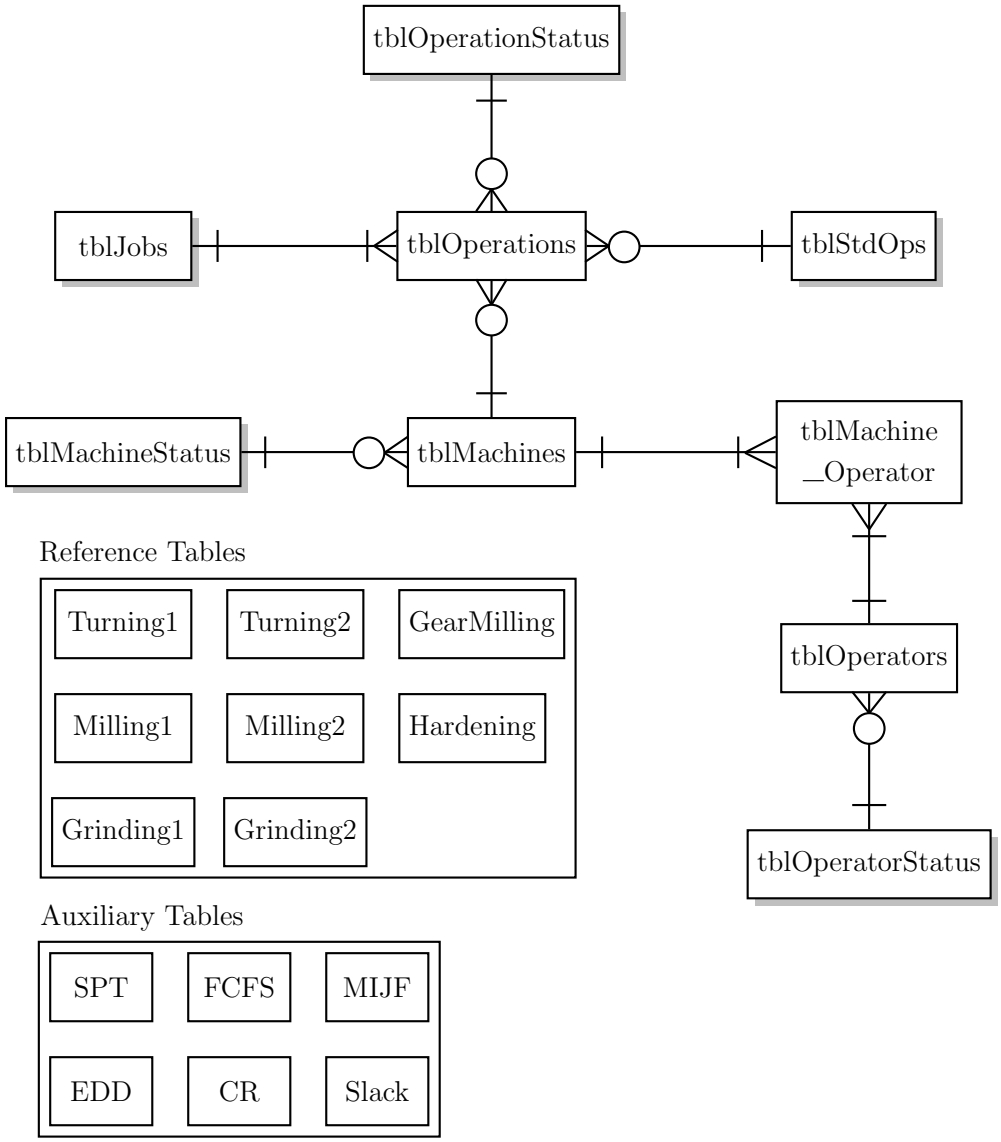


Figure 5.2: Entity relationship diagram of proposed information system

5.1.2 Construction of web pages

As with the information system, the web pages that need to be created should make use of a cloud server. To accomplish this, a virtual machine needed to be set up in such a way that it could be accessed from another device via the internet. This was accomplished by using a desktop and enabling the desktop to be accessed via a *Virtual Private Network*. This machine was then given authorisation to access the MySQL server, enabling the web pages to connect to the information system. Microsoft® Visual Studio®[1] was then installed on the machine, which will be used to create the different web pages.

[1]The registered trademark for Microsoft® Visual Studio® will from now on be omitted.

The web pages that are essential for logging disturbances in the system as well as processing data, include:

1. logging a broken machine,
2. logging a new job entering the system,
3. changing an operator status,
4. displaying html simulation report,
5. choosing new schedule,
6. manual capturing of sensor data, and
7. adding or removing a machine.

Each of these web pages will now be described, with reference to their layout and functionality.

Logging a broken machine

As illustrated in Figure 5.3, the layout of this page is quite simple. When the web page is loaded, a grid view displays each machine currently in the system and their status. There is also a drop-down list showing all the machine names, as well as a drop-down list that lists the different machine statuses. The user can therefore select a machine and status from the drop-down lists and then click the *"Change Machine Status"* button. This will log the status change in the information system, which will in turn change the data in the grid view to illustrate the change that occurred. An email will also be sent to inform the manager that a machine status has changed, and the manager can then decide whether or not to reschedule.

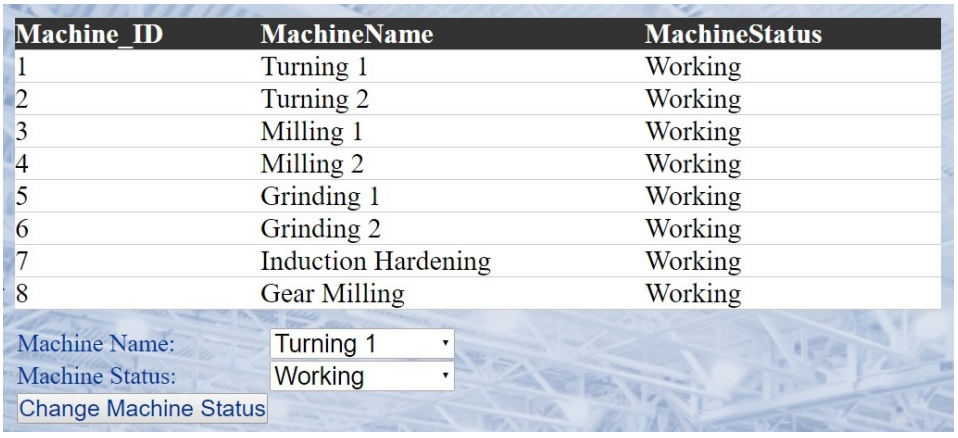


Figure 5.3: Layout of logging a broken machine web page

Logging a new job

This web page is illustrated in Figure 5.4. When the web page is loaded, a grid view appears that displays all the jobs that have already entered the system. There are also text boxes where the user can enter the name, description, quantity and RFID of the new job, as well as a drop-down list that lists the priorities that the user can choose. Lastly, there is a calendar which is used to select the due date of the job. When the user completes all these inputs and clicks the *"Log new job"* button, the data is stored in the information system and the grid view updates with the new job included. An email will also be sent to inform the manager that a new job has entered the shop, and the manager can then decide whether or not to reschedule. When the grid view is updated, another drop-down list appears which lists the machines currently in the shop, together with text boxes where the user can define the processing time, earliest start time and latest start time of the operation. The format of the processing time is *"Hours:Minutes:Seconds"*, while the format of the earliest start time and latest possible start time is *"Year/Month/Day Hours:Minutes:Seconds"*. When the user completes these new inputs and clicks the *"Add operation"* button, the data is inserted into the information system and a new grid view appears with the operations that are linked to the new job. This step can be repeated until all the operations of the new job are stored in the information system. Typically, this web page will be used by an

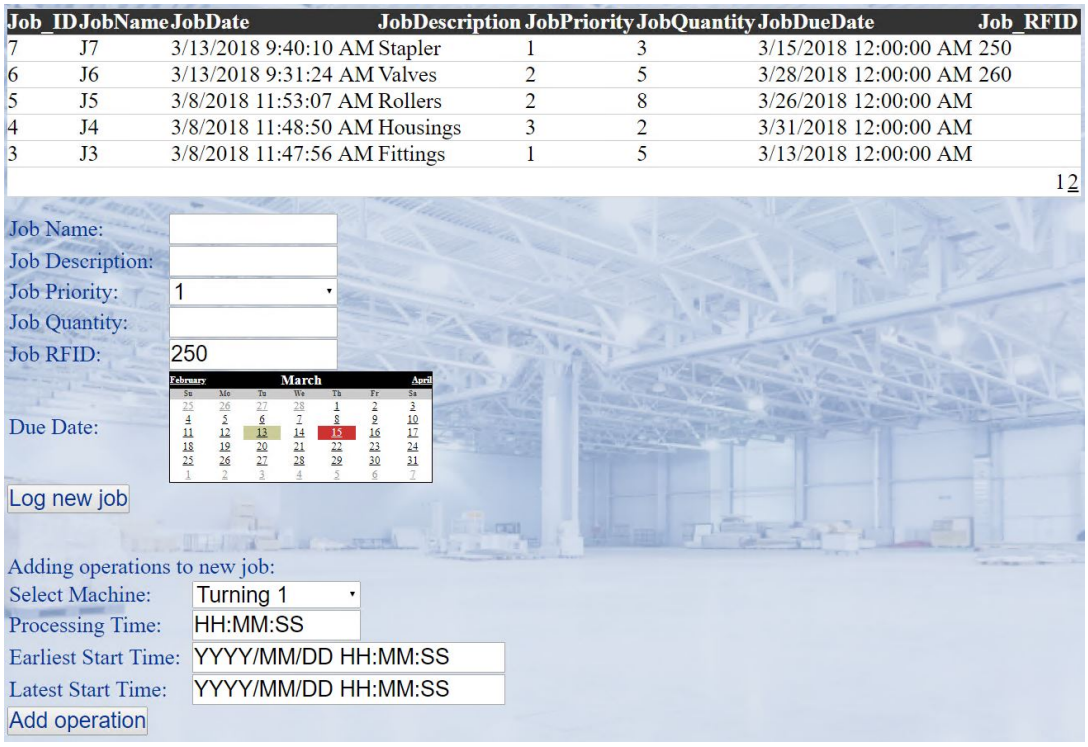


Figure 5.4: Layout of logging a new job web page

experienced planner which will link several operations to a specific job.

Change operator status

The web page which allows the user to change the operator status is illustrated in Figure 5.5. When the web page is loaded, a grid view appears that displays each operator's name with their current status in the system. Two buttons also appear; the *"Add New Operator"* button and the *"Change Operator Status"* button. When the *"Add New Operator"* button is selected, a text box appears, where the user can insert the new operator's name, and a drop-down list that displays the different operator statuses that can be chosen. If the inputs are completed and the *"Add Operator"* button is selected, the data is stored in the information system and the grid view is updated with the new operator added. The choice of machines to which the new operator can be linked is limited to a single machine.

Otherwise, if the *"Change Operator Status"* buttonis selected, two drop-down lists appear which list the operators currently in the system and the different operator statuses respectively. When the user selects an operator and status and selects the *"Update Status"* button, the data is updated in the information system and the grid view is updated to display the updated information. An email will also be sent to inform the manager that an

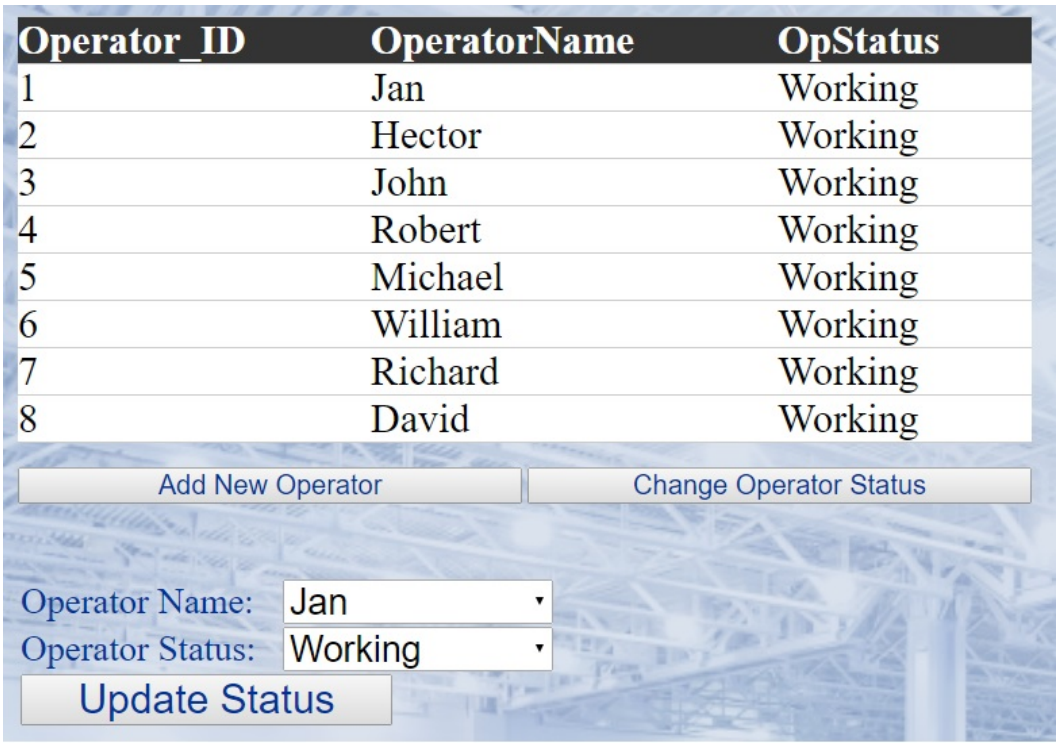


Figure 5.5: Layout of the change operator status web page

operator status has changed, and the manager can then decide whether or not to reschedule.

Display HTML simulation report

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. An HTML report is created on test completion and is a convenient way of displaying test results. This web page has a simple layout with two buttons. If the *"Display Report"* button is clicked, the HTML report that was generated by the simulation model will be displayed in a frame on the web page. Alternatively, if the *"Download Report"* button is clicked, the HTML report is downloaded.

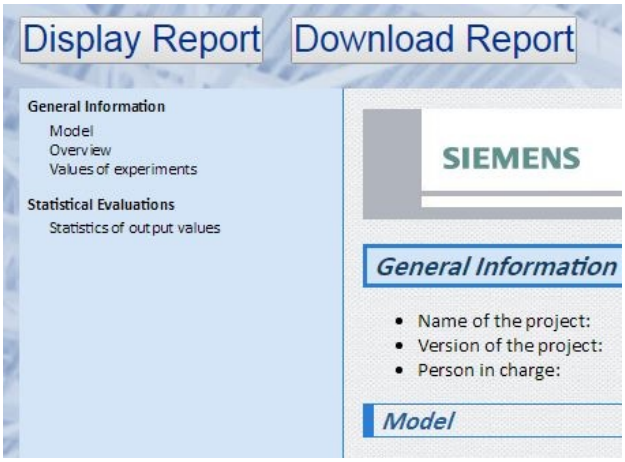


Figure 5.6: A snapshot of the layout of the display HTML simulation report web page

Choose new schedule

This web page contains a list of check boxes, each associated with a specific dispatching rule. The dispatching rule that the manager wants to implement can therefore be selected and the *"Generate Schedule"* button can be clicked. The reference tables that display the machine processing sequences will then be updated from the auxiliary table associated with the selected dispatching rule.

Figure 5.7: Layout of new schedule selection web page

Manual capturing of sensor data

This web page will only be used when a sensor fails or is broken and the tracking of jobs through the shop has to be logged manually. This page will ensure that quality scenario *Sc_2.3* is adhered to. The layout of the page is illustrated in Figure 5.8. There are two drop-down lists present on the page. The first list displays the machines that are in the system and when a machine is selected, the operations that are scheduled on it are displayed in a grid view. The operation that requires a status change can then be selected from the grid view after which the new status can be selected from the second drop-down list. Finally, when the *"Update operation status"* button is selected, the status of the selected operation is updated to the new status and the grid view is updated. This web page will typically be used by an experienced planner, who allocated the operations of the jobs to the machines.

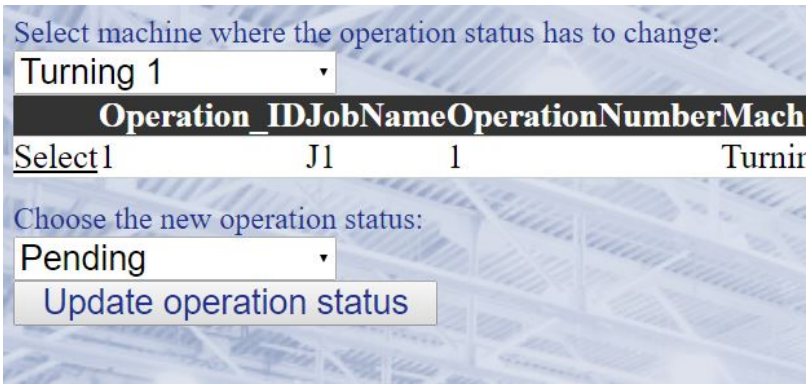


Figure 5.8: Layout of manual data capture web page

Adding or removing a machine

This web page will only be used when a machine is either bought or sold and the information system needs to be updated accordingly. This page will ensure that quality scenario *Sc_2.4*

is adhered to. The layout of the page is illustrated in Figure 5.9. There is firstly a grid view that lists all the current machines in the system. Below the grid view, there are two buttons that can be selected to either add or remove a machine. To remove a machine from the list, the machine record must first be selected in the grid view and then the *"Remove machine"* button can be selected which will delete the machine's record in the database and update the grid view.

On the other hand, if the *"Add machine"* button is selected, a text box appears where the new machine name can be entered. The *"Insert new machine"* button can then be selected which will insert the new machine record in the database and update the grid view with the new record.

By adding or removing machines in the shop, the layout of the shop changes, which will require the simulation model to also change. The system administrators are therefore responsible for updating the simulation model when these changes occur.

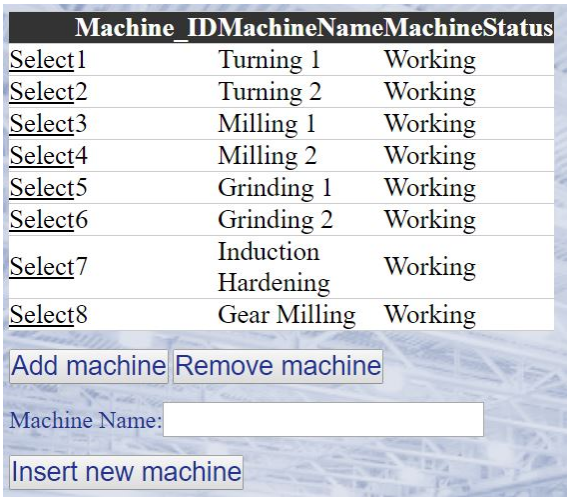


Figure 5.9: Layout of adding or removing a machine web page

5.2 Development of the simulation scheduling model

This section focuses on the development of the simulation model that performs the rescheduling process. This model was built in Tecnomatix Plant Simulation®[1] which is a simulation software package. The researcher chose this simulation software package because the University of Stellenbosch has a licence for it and also the researcher has experience in using this package from undergraduate studies. The model will create schedules according to the different dispatching rules and then calculate the KPI values.

[1]The registered trademark for Tecnomatix Plant Simulation® will from now on be omitted.

The different components that need to be addressed in the simulation model include:

1. MySQL data import,
2. dispatching rules,
3. machine schedules,
4. machine behaviour,
5. experiment inputs, and
6. experiment outputs.

The incorporation of these components into the simulation model will now be discussed.

MySQL data import

Before data could be imported from the MySQL server, a connection had to be established between the simulation model and the server. This connection was established by the use of an *Open Database Connectivity* (ODBC) object which logs onto the server. A method could then be created (*"ImportMySQL"*) which used the ODBC to log onto the MySQL server. The method also contained SQL statements that selected data from tables in the server and then imported the data to the tables in the simulation model. The tables that were accessed are shown in Figure 5.10.

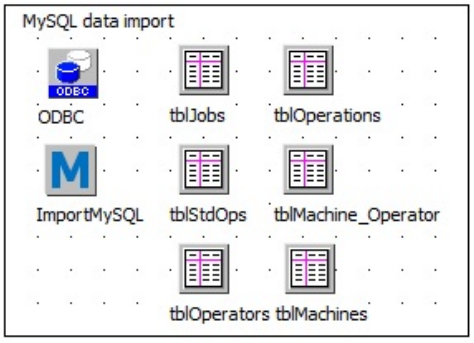


Figure 5.10: MySQL data import component

Dispatching rules

After the MySQL data has been imported into the simulation model, the data can be used to create the machine schedules according to the dispatching rules described in Table 2.1. Figure 5.11 illustrates how the dispatching rules component has been incorporated into the simulation model. Each dispatching rule is written into its own method which calls the

data from the imported tables, applies the dispatching rule and then generates the machine schedules according to the outcome of the dispatching rule. When a machine is broken or the operator located at that machine is absent, the dispatching rule will allocate all the operations of that machine to the duplicate machine, if applicable. When there is only one of a specific type of machine, then the dispatching rule will allocate the operations to the machine, even if the machine is broken or the operator is absent. These methods are called by the simulation model before the model starts.

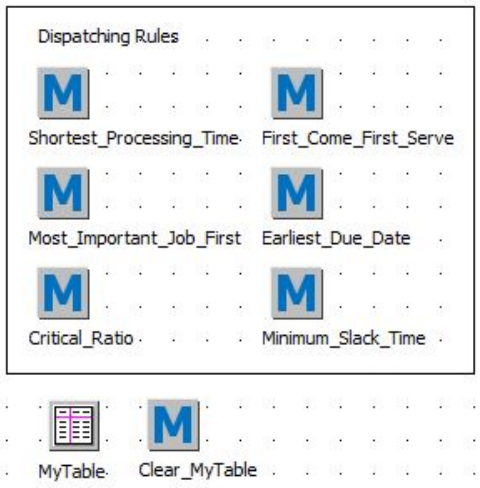


Figure 5.11: Dispatching rules

Due to the stochastic nature of processing times, the operation processing times are altered in each dispatching rule by means of a lognormal distribution. The experienced planner estimates the expected processing time of an operation; however, there can be variation. The variation is built into the dispatching rule through the lognormal distribution. It was decided that the distribution will have a mean that is equal to the expected processing time the user entered into the information system, and a standard deviation of 20 percent of the expected processing time. The processing time of the operation in each machine schedule is then set to the value generated from the distribution.

While setting up the dispatching rules, a problem was observed when the rules are applied on operation-level. A simple example of this problem will now be discussed. The example consists of two jobs that have two operations each, together with two different machines as illustrated in Figure 5.12. When the *shortest processing time* (SPT) dispatching rule is applied on operation-level, the schedule on each machine will be generated as shown in Figure 5.13. With this schedule, the second operation of both jobs is scheduled to be processed first on each machine, but this will violate the predetermined sequence of

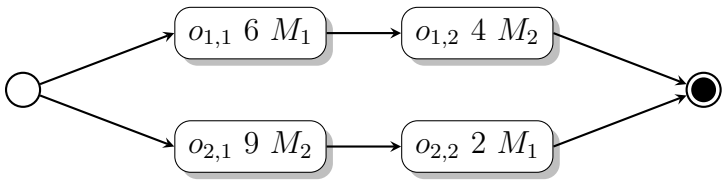


Figure 5.12: Simple example of two jobs and two different machines

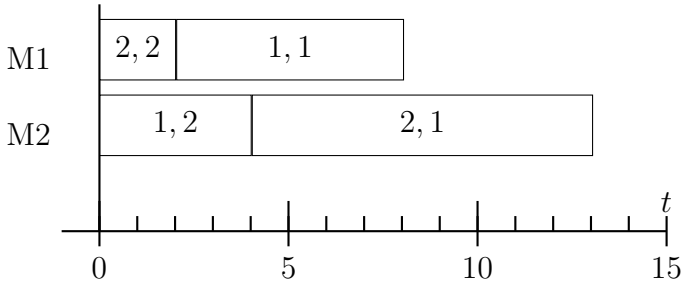


Figure 5.13: Schedule generated after shortest processing time rule was applied on operation-level

operations and can therefore not occur. This problem then causes the system to deadlock, and processing cannot commence. This example only takes the SPT rule into account, but there are similar scenarios that can cause the same problem when applying the other dispatching rules. This problem will, however, not occur if the machines are identical.

Montazeri and Van Wassenhove (1990) discuss all the types of dispatching rule that can be used in a manufacturing environment, and it is evident from their explanations that the six dispatching rules incorporated into the model are indeed created for job-level scheduling. Operation-level scheduling rules were also discussed in the paper, which include but are not limited to *shortest imminent operation time* (SIO) and *longest imminent operation time* (LIO). It was therefore decided to implement the dispatching rules on job-level. The new schedule for this example is illustrated in Figure 5.14, where the SPT was implemented on job-level. From the example it can be determined the J_1 has a processing time of ten minutes, which is shorter than the 11 minute processing time of J_2. Therefore, according to the SPT rule, the operations of J_1 have to be scheduled first, followed by the operations of J_2.

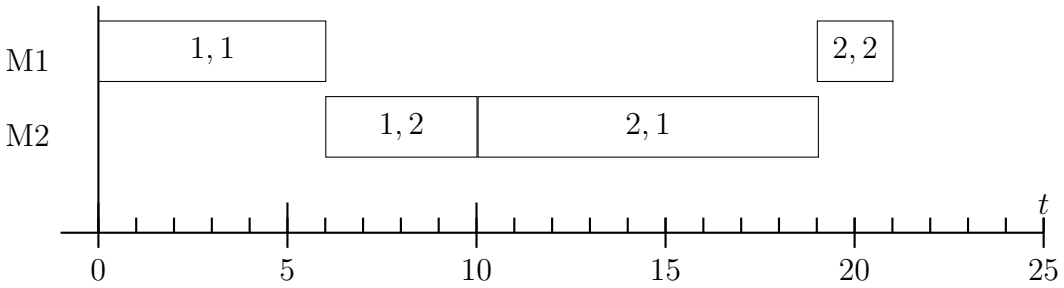
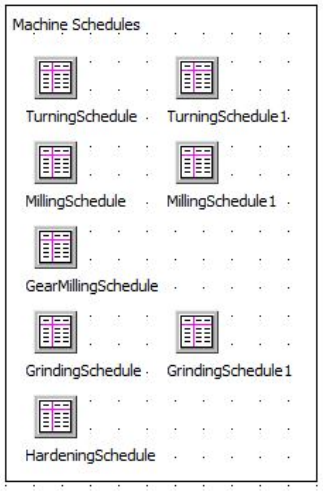


Figure 5.14: Schedule generated after shortest processing time rule was applied on job-level

Machine schedules

The machine schedules that are generated by the dispatching rule methods are stored in the tables illustrated in Figure 5.15a. Each machine in the system has its own table that it can refer to for the operation sequence it is required to follow. Each table contains the job ID, operation number, processing time and earliest start time of the jobs that have to be processed on the machine. An example of the contents of the schedule table for a machine is illustrated in Figure 5.15b.



(a) Machine schedule tables

	integer 1	integer 2	time 3	boolean 4	datetime 5	st 6
string	Job	OpNumber	ProcTime	Finished	EarliestStart	
1	34	1	1:51:52.0000	true	2018/06/05 13:00:00.0000	
2	1	1	1:33:13.0000	true	2018/06/05 13:00:00.0000	
3	10	1	1:51:50.0000	true	2018/06/05 13:00:00.0000	
4	8	1	5:26:16.0000	true	2018/06/05 13:00:00.0000	
5	26	1	7:46:05.0000	true	2018/06/05 13:00:00.0000	
6	28	3	46:36.0000	true	2018/06/05 16:06:00.0000	
7	44	3	46:36.0000	true	2018/06/05 16:06:00.0000	
8	42	3	46:36.0000	true	2018/06/05 16:06:00.0000	
9	7	2	2:54:45.0000	true	2018/06/05 15:08:00.0000	
10	4	3	37:18.0000	true	2018/06/05 17:02:00.0000	

(b) Schedule table content

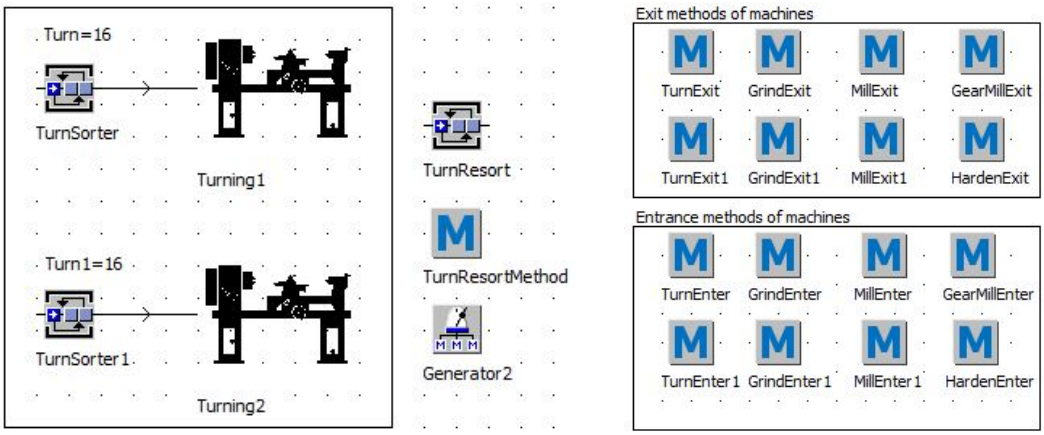
Figure 5.15: Machine schedules

Machine behaviour

Figure 5.16a illustrates an example of how the machines were incorporated into the simulation model. In this example, there are two turning machines. Each machine has a sorter in front of it. The sorter is guided by a method that checks its contents and if the method finds the job that needs to be processed next, according to the machine schedule, the job

is moved from the sorter to the machine. This method ensures that the sequence specified in the machine schedule is adhered to.

There is also another sorter and method together with a generator added in the model, as illustrated in Figure 5.16a. When a machine is either broken or the operator located at that machine is absent, the model will start with that machine unavailable. This extra method is then used to constantly check the status of the machine and when the machine is operational again, all the jobs that were allocated to the other machine should move to the extra sorter. A new sequence is then created for both machines, and the jobs are moved back to the applicable sorter. When both machines are broken, the model will start with both machines unavailable. These components were only added where there are identical machines. When a one of a kind machine is broken, the model will also start with that machine unavailable. The jobs that were allocated to that machine will queue at the sorter until the machine becomes operational again, after which processing will commence.



(a) Example of machines component.

(b) Entrance and exit methods used by machines.

Figure 5.16: Machines, entrance and exit methods in the simulation model

When the job then enters the machine, the machine calls an entrance method (illustrated in Figure 5.16b). This method identifies the job and determines the processing time of the job by locating the processing time of the job in the machine schedule. The processing time of the machine is then changed to the relevant value.

After the machine finishes the processing of the job, the machine calls an exit method that determines the destination where the job must be sent next. If the next destination is another machine, the method will send the job to the sorter of the specific machine, otherwise the job will be sent to the drain of the simulation model and exit the system.

Table 5.11: Types of machines included in simulation model

Machine types	**Number of machines**
Turning	2
Milling	2
Grinding	2
Gear Milling	1
Induction Hardening	1

For the construction of the simulation model it was decided that there should be eight machines included in the model. Table 5.11 lists the different types of machines as well as the number of machines per type, that are included in the model. The types of machines that were selected are based on the machines that were included in a similar job shop model that was created by Bangsow (2015).

Another factor that needed to be taken into account is the operating hours of the shop. For this, a shift was included in the model that defines the operating hours of the shop from 08:00 to 17:00. The machines will therefore not accept any operation for processing outside the shift time, but if there is an operation being processed when the time reaches 17:00, the machine will continue processing until the operation is finished. Figure 5.17 displays a Gantt chart that illustrates the time that falls outside the shift time, which indicates that no new operation is accepted by the machines. However, there was already an operation that started at *M8* before it reached 17:00, which will first be finished before the machine is stopped.

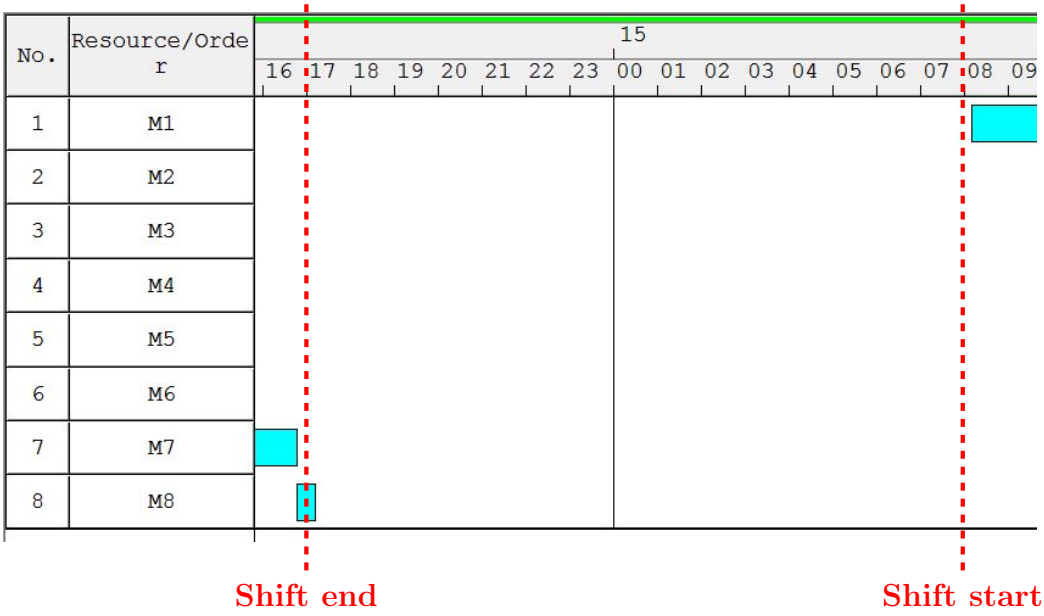
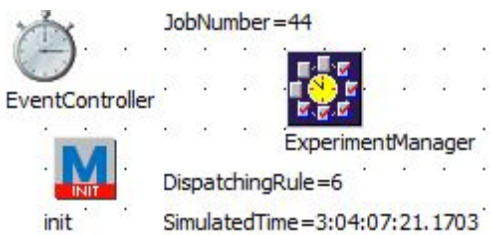


Figure 5.17: Gantt chart of machine processing outside shift hours

Experiment inputs

Now that all the components required for running once through the simulation model are incorporated, the experiments need to be defined. There are six scenarios that need to be addressed, one for each of the dispatching rules. Therefore, an experiment manager was included into the simulation model, as seen in Figure 5.18a, and experiment inputs were defined. The inputs change the value of a variable called *"DispatchingRules"*. The values the variable is changed to are illustrated in Figure 5.18b, where each number refers to a specific dispatching rule. Therefore, when the model runs, only one dispatching rule can be used. The experiment manager was also set up to run 25 replications per scenario.



(a) Input components

	Active	root.DispatchingRule
1	true	1
2	true	2
3	true	3
4	true	4
5	true	5
6	true	6

(b) Input values

Figure 5.18: Input components and values of the simulation model

Experiment outputs

After the experiment inputs are defined, the outputs can be addressed, which can be used to compare the different scenarios with each other. The outputs of the simulation model are the KPIs that were defined in Table 2.2 and are illustrated in Figure 5.19. The outputs are shown in *"Days:Hours:Minutes:Seconds"*. When the final outputs are calculated, a confidence interval graph is generated for each output, where the different scenarios are compared. The best solution can therefore be selected and implemented after analysing these graphs.

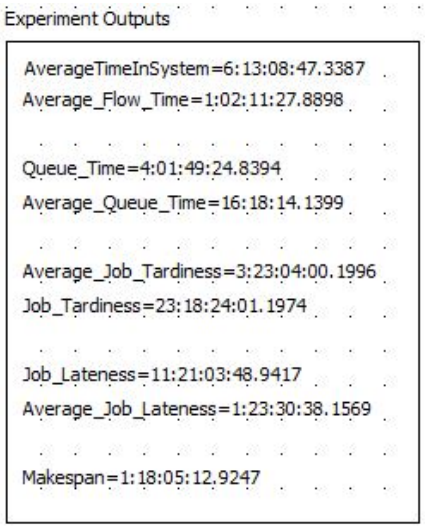


Figure 5.19: Output component of the simulation model

While the *Experiment Manager* runs the experiments, the schedules generated for each dispatching rule are stored in the auxiliary tables in the information system. These tables are used to retain the schedules for each dispatching rule, and when the new schedule is chosen by the manager, the reference tables are populated from these auxiliary tables.

This concludes the description of the software that was developed for the proposed system. Next the hardware developed for the system will be discussed.

5.3 Development of the sensors

When referring back to the architectural design of the proposed system, a sensorised shop floor forms part of the system. Therefore, this section serves to describe the construction and implementation of the sensors that will be used in the sensorised job shop. The description will make reference to the components used and functionality of the devices.

5.3.1 Components used

This subsection will be used to describe the technical information of all the components that were used in the development of the sensors.

Raspberry Pi

The Raspberry Pi is a well-known and well-supported compact computer that runs on the *Debian* platform which forms part of the *Linux* operating system (OS) (Raspberry Pi Foundation, 2014). The *Linux* OS environment integrates seamlessly with the hardware of the Raspberry Pi through Python®[1]. The *serial peripheral interface* (SPI) bus of the Raspberry Pi is used to communicate with the radio frequency transceiver, RFM98. The RFM98 is discussed later in this section. Figure 5.20 shows an illustration of a Raspberry Pi. The Raspberry Pi will act as a gateway that logs data changes received from the sensors on the cloud-based information system. The Raspberry Pi will be able to log onto the cloud-based information system through either *WiFi* or a *Local Area Network*. Only one Raspberry Pi is required to log data changes, because it is capable of receiving data from several different devices.

Figure 5.20: Illustration of a Raspberry Pi (The Pi Hut, 2017).

Arduino Pro Mini

The Arduino Pro Mini consists mainly of the Atmel Atmega328P micro-controller (Arduino, 2017). The operating voltage is 3.3V, which is ideal for this low power sensor due to the use of only 3.3V components. The low cost of this micro-controller together with its functionality that supports the sensor outcomes, made it the ideal choice (Scott Tattersall, 2018). The Arduino serves as the controller of the sensor, which captures the data from the RFID module and buttons and sends this data to the RFM98 module. The RFM98 and

[1]The registered trademark for Python® will from now on be omitted.

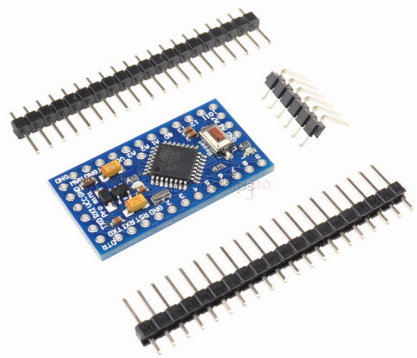


Figure 5.21: Illustration of an Arduino Pro Mini (Makerfabs, 2016).

RFID modules communicate via the SPI bus lines with the Arduino. The buttons trigger general purpose input output (GPIO) pins on the Arduino for user interface. Figure 5.21 provides an illustration of an Arduino Pro Mini.

LoRa (RFM98)

The RFM98 module is a breakout board that holds the Semtech LoRa *integrated circuit* (IC). LoRa is a long-range, low-power radio frequency communication platform which is at the forefront of *Internet of Things* (IoT) network applications worldwide (Semtech, 2018). It makes use of the low frequency sub-gigahertz bandwidth that is part of the free communication spectrum in South Africa. The disadvantage of using low frequency communication is that the information size is limited to small quantities (RF Wireless World, 2017). However, this application makes use of low data rates which makes the RFM98 module the ideal choice. LoRa also has the functionality to both transmit and receive information over a confirmed line of sight distance of between 15 and 20km (Murata, 2018).

The RFM98 module is used because of its compact design and online support. Figure 5.22 shows an illustration of the Semtech LoRa module. There are $n+1$ LoRa modules present in the system, where n is the number of sensors present in the system. The extra module is located at the gateway. The LoRa module on each sensor transmits the data package created by the sensor to the LoRa module located at the gateway, which then receives the package. The package can then be used by the Raspberry Pi to log the data changes in the cloud-based information system.

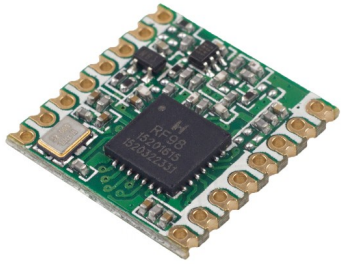


Figure 5.22: Illustration of a Semtech LoRa chip (Seeed Studio, 2017).

RFID (RC522)

The RFID RC522 module enables the user to identify and communicate a card or tag ID to the Arduino via the SPI bus. In this application passive cards and tags are used as it is the cheapest and does not require a battery. The cards and tags utilise the radio energy that is transmitted by the reader to energise its circuit and to provide its ID (NXP, 2016). Figure 5.23 shows an illustration of a RFID reader.

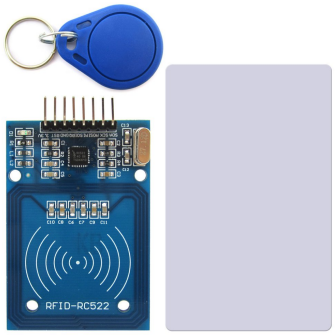


Figure 5.23: Illustration of a RFID reader (Lazanda, 2018).

Programmer

The programmer is a breakout board for the FTDI FT232 USB to serial integrated circuit. This component is used to program and develop the Arduino Pro Mini with the functionality required for the specific sensor. The programmer makes use of the serial connection on the Arduino to write the developed code onto the Atmel micro-controller. Figure 5.24 shows an illustration of the programmer. In this application an external programmer is used as it reduces the cost of each sensor (Arduino, 2017).

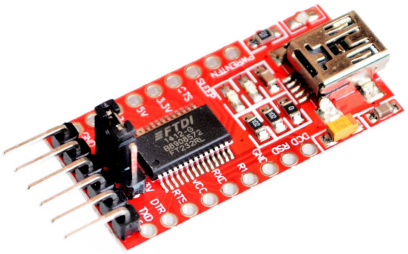


Figure 5.24: Illustration of a Programmer for Arduino Pro Mini (Ardu Shop, 2017).

Power supply

The sensors were developed to be seamlessly integrated into the proposed system by means of battery power. To make this possible and user-friendly, a charging module and 18650 Lithium-Ion battery were added to the sensors. The battery has a capacity of 2600mAh and is widely available (AA Portable Power Corp, 2004). A typical power bank charging module is used with the battery. It enables the user to easily charge the sensor through a USB. Figure 5.25 shows an illustration of the charging module.

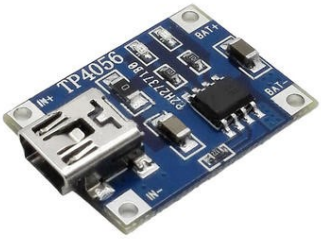


Figure 5.25: Illustration of a charging module (Bang Good, 2016).

5.3.2 Integration of components into sensor nodes

This subsection will describe how all the sensor components are integrated, as well as the interaction between these components.

Figures 5.26 and 5.27 display the layout of the printed circuit board (PCB) used for the sensor. There are five parts that are incorporated into the PCB and they are listed as follows (the numbers in the list refer to the numbers shown in Figures 5.26 and 5.27):

1. The RFID reader that reads the ID from a card and processes the information to the SPI lines which are used by the Ardunio Pro Mini.

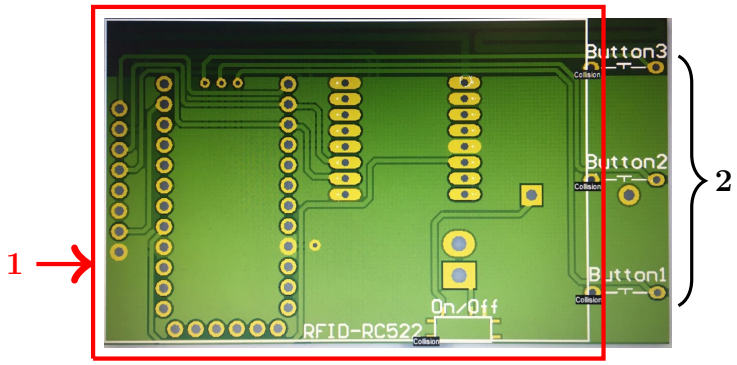


Figure 5.26: Top view of sensor PCB

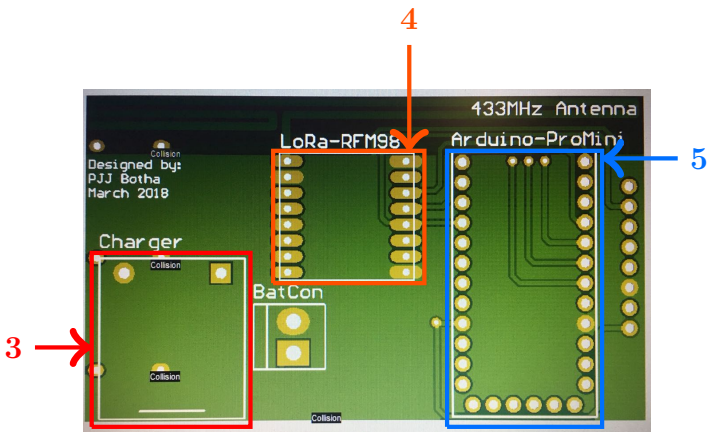


Figure 5.27: Bottom view of sensor PCB

2. The three buttons that define the status (*i.e. Waiting*, *Processing*, *Completed*) of an operation.
3. The charger and external battery that supplies power to the sensor.
4. The Semtech LoRa module, which acts as a transmitter and makes use of low-frequency communication to transmit data from the sensor to a receiver.
5. The Arduino Pro Mini that communicates information from one external peripheral to another. The information that is communicated includes the ID from the RFID reader to the LoRa chip, as well as the button number from the button to the LoRa chip.

Figure 5.28 illustrates the interaction and information flow through the sensor. The information flow starts off when the operator of a machine swipes a RFID card linked to a job at the sensor. The RFID reader detects the ID of the card and processes the information to the Arduino. The operator must select a button, which will process the ID of the button

to the Arduino. Only when the Arduino receives both the card ID and the button ID will it be able to build the data package. The data package is then communicated to the LoRa chip, which will then transmit the package to the gateway, which is the Raspberry Pi. The data package will consist of a string of numbers that include the sensor ID, card ID and the button ID. The sensor ID in this application will be the same as the machine ID where it is located. There are some exceptions that need to be taken into consideration, which include:

1. An operator accidentally presses a button twice.
2. An operator forgets to swipe a card and/or press a button.

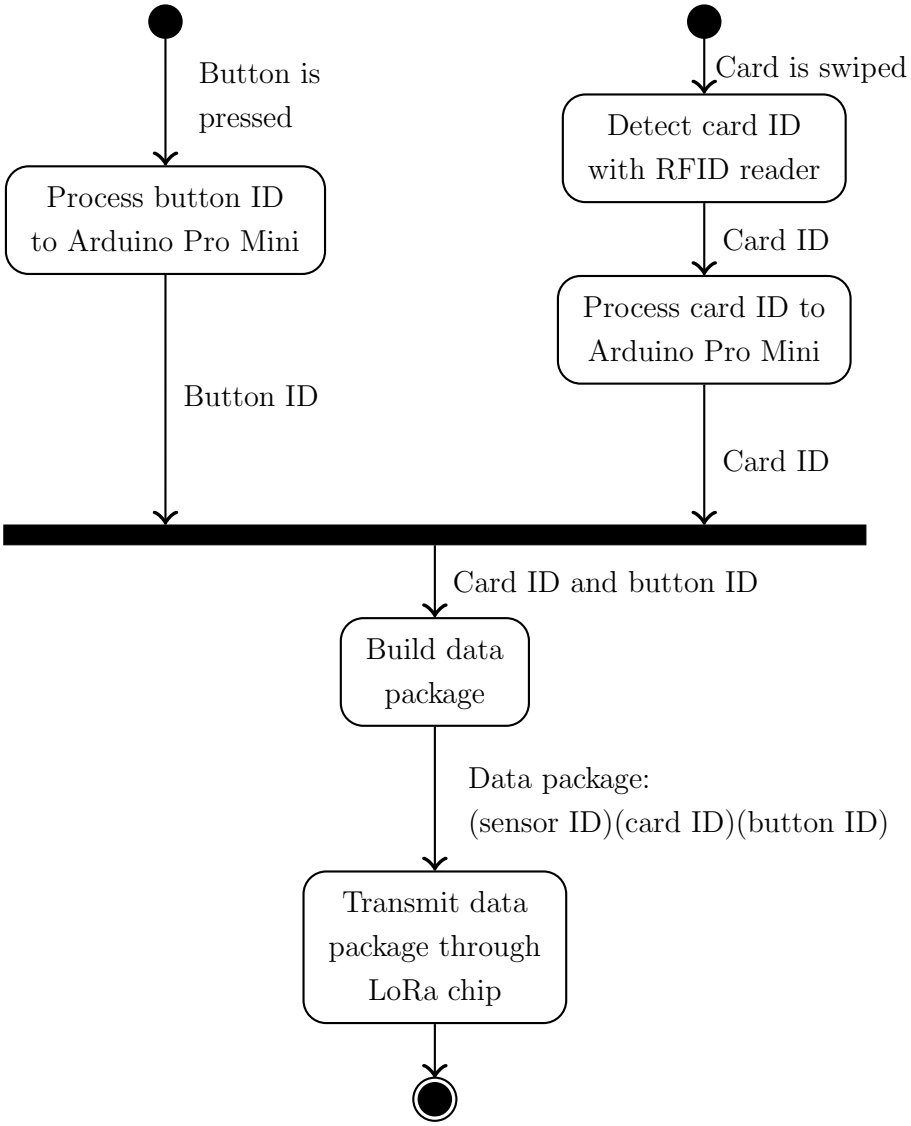


Figure 5.28: Activity diagram of information flow through sensor

To overcome both these exceptions, we assume that the operators will be trained accordingly and will work meticulously. Exception 1 does however not have an effect on the outcome, because the sensors were developed in such a way that the ID of the last button that was pressed will be used, even if multiple button presses occurred.

5.3.3 Integration of components into gateway

The gateway refers to the components that receive data from the sensors and log the data changes in the cloud-based information system. Figure 5.29 illustrates the layout of the PCB used for the gateway. The following components are incorporated into the gateway (the numbers in the list refer to the numbers shown in Figure 5.29):

1. The Raspberry Pi that connects to the cloud-based information system to log the data changes.
2. The LoRa module, which acts as a receiver of information that was transmitted to it.

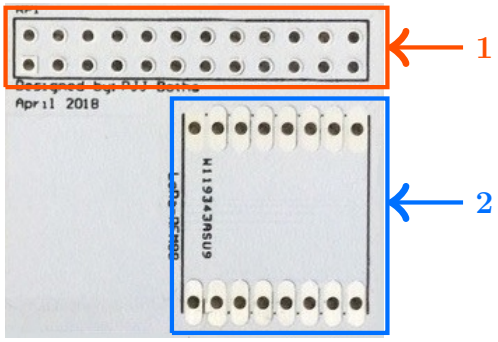


Figure 5.29: PCB design for the gateway

Figure 5.30 illustrates the information flow through the gateway. The information flow starts off with the LoRa module that receives the data package that was transmitted to it, after which it processes it to the Raspberry Pi. The Raspberry Pi then logs onto the cloud-based information system and determines which button was pressed by inspecting the button ID that forms part of the package. Thereafter, decisions are made to determine which update must be performed on the information system. Due to the link between every `Job_ID` and a unique card ID (as seen in Table 5.7), the gateway can determine which job is currently at the machine. The gateway can then determine the operation number at the machine of that job by searching in `tblOperations`. Therefore, if the button ID = "1", then the Raspberry Pi updates the status of the operation associated with the machine

ID and card ID, to *Waiting*; otherwise, if the button ID = "2", the Raspberry Pi updates the status of the operation associated with the machine ID and card ID, to *Processing*. If neither of the above-mentioned decisions are true, then the button ID must be "3" and the Raspberry Pi will update the status of the operation associated with the machine ID and card ID to *Completed*.

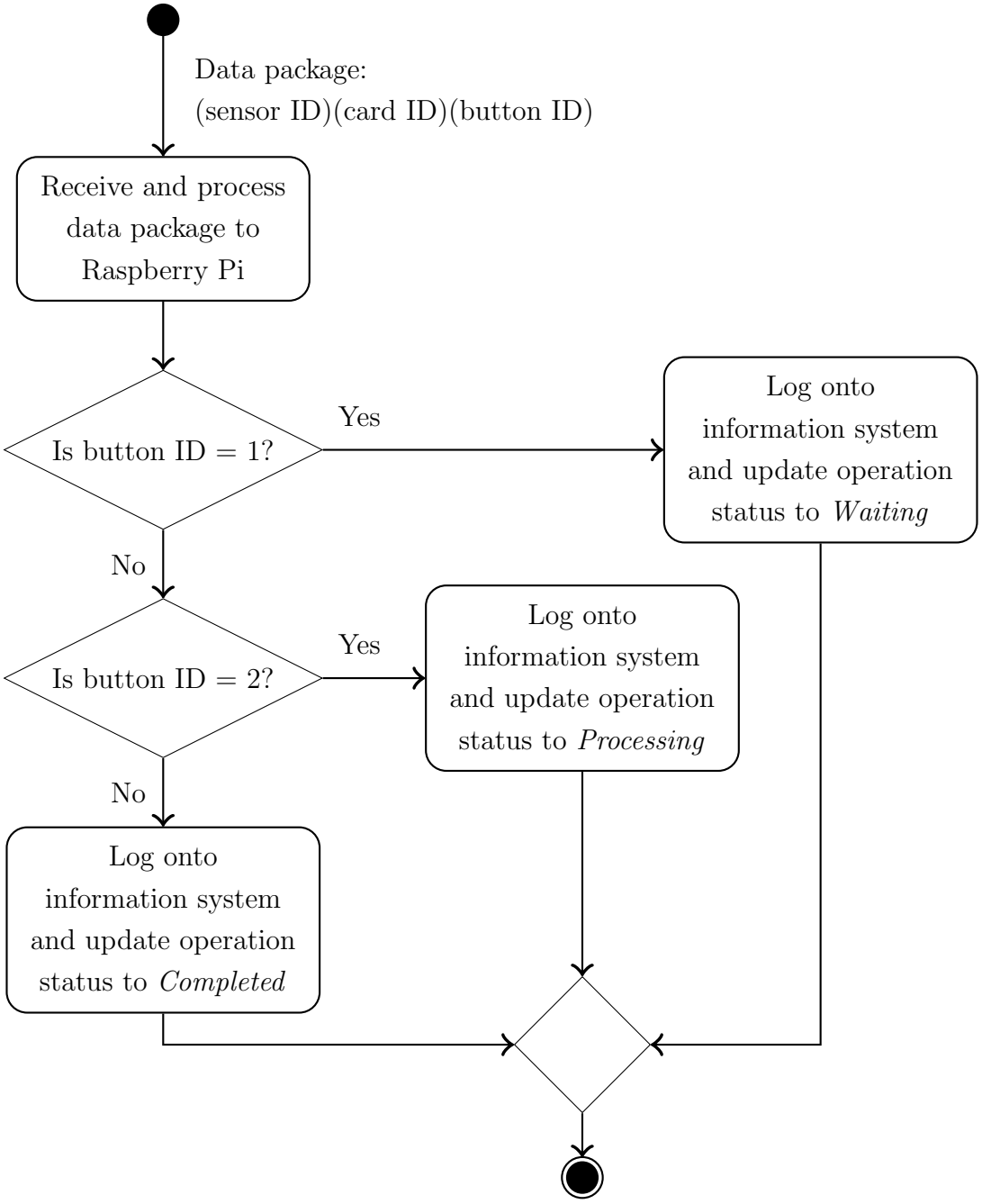


Figure 5.30: Activity diagram of information flow in the gateway

5.3.4 Sensor casing design

A casing was designed in *Autodesk Inventor* to enclose the sensor and all its components, as illustrated in Figure 5.31. The casing holds the battery safely underneath the sensor, and push buttons were designed to enable the easy use of the input buttons on the sensor within the case. There are also two openings on the one side of the casing, one for the recharging port of the battery-charging module and the other for the switch that turns the sensor on and off. After the design of the casing, it was 3D printed.

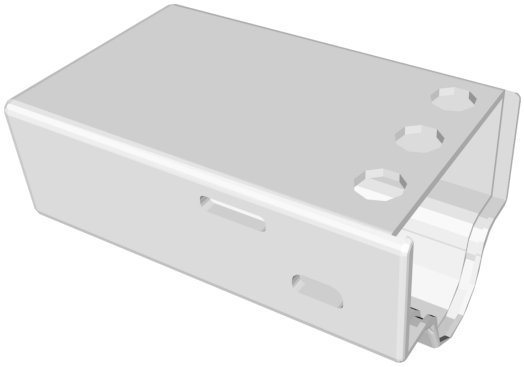


Figure 5.31: Sensor casing 3D design

5.4 Chapter summary

This chapter served the purpose of documenting the development process of the proposed system. The components that were developed for the system included an information system where all required data is stored, the web pages used for logging data changes, a simulation model that mimics the behaviour of a job shop, and finally, sensors for logging real-time data. Validation and testing of both the simulation model and the sensors was also conducted and discussed in Appendix B and C. After the validation and testing, it could be concluded that the system was successfully developed and each component work correctly according to its design. The following chapter will address the incorporation of metaheuristics into the simulation scheduler.

Chapter 6

Incorporation of metaheuristics in simulation scheduler

This chapter will describe the incorporation of metaheuristics in the simulation scheduler. The metaheuristics that were implemented are the *simulated annealing* and a variation of the *2-opt* algorithm. Detailed analysis and comparison of metaheuristics and dispatching rules is also provided.

6.1 Implementation of metaheuristics

From the analysis of the schedules generated by the simulation scheduler according to each dispatching rule, as presented in Appendix B, it was apparent that there is still excessive machine idle time. This indicates that the generated schedules can be improved upon. Cinar *et al.* (2015) state that dispatching rules are easy to implement; however, the quality of results decreases as the problem size increases. Due to this drawback and the increasing demand for better solutions, metaheuristics have been widely used for job shop scheduling problems. Sels *et al.* (2012) also state that the "performance of dispatching rules for the job shop problem has little value since other more advanced and computationally more demanding procedures, such as metaheuristics, lead to much better results".

Several studies have been done to solve the job shop scheduling problem with a combination of dispatching rules and metaheuristics. These studies include a paper by Moradi *et al.* (2011) that combined the genetic algorithm with priority dispatching rules for the job shop scheduling problem with non-fixed preventative maintenance activities. Scrich *et al.* (2004) developed a hierarchical and multi-start tabu search where the initial solution is obtained by priority dispatching rules. Lastly, Wang *et al.* (2008) developed a filtered beam search-based heuristic algorithm to solve a multi-objective job shop scheduling problem. To avoid useless paths and decrease the computational time, the authors used heuristics based on dispatching rules as local and global evaluation functions. From these studies and the literature, it can be deduced that the use of dispatching rules for the job shop scheduling problem is appropriate to assist and guide the implementation of metaheuristics or when the problem size is small. For these reasons, the researcher decided to identify metaheuristics that can be implemented to increase the quality of the generated schedules.

6.1.1 Selection of metaheuristics

While searching for possible metaheuristics that can be used to solve the job shop scheduling problem, the researcher found that scheduling problems have previously been modelled as *travelling salesman problems* (TSP). According to Hoffman *et al.* (2013), the problem can be simply stated as: "if a travelling salesman wishes to visit exactly once each of a list of m cities (where the cost of travelling from city i to city j is c_{ij}) and then return to the home city, what is the least costly route the travelling salesman can take"?

Kosiba *et al.* (1992) provided a novel perspective on the use of the TSP methodology to the problem of steel product sequencing through the milling process of a large manufacturing facility. The paper showed how more traditional heuristic scheduling procedures can be restructured as TSP with significant improvement in attainment of production objectives. Whitley *et al.* (1989) stated that by generating potential solutions to the TSP is interesting due to it not using any information about the distance between cities, but rather the cost or performance of the overall tour. This is important, because it means the method can be used for scheduling problems where there are no distances to measure, but rather the overall evaluation of the total sequence. Solutions to such problems are encoded into binary or integer strings and are referred to in literature as *encodings*. The sequence of these encodings can then be interpreted as the sequence in which cities must be visited.

Whitley *et al.* (1989) also alluded to the problem that is faced when creating a binary encoding for the TSP. When binary encodings are crossed over to form alternative solutions, it can result in duplication of *cities* or the omission of *cities*. Another problem with binary encodings is the large volume of computational resources that is required to decode the binary encoding to an integer encoding, applying the changes and then encoding it back to binary. Due to these difficulties, it was decided to make use of an integer encoding. Local search metaheuristics will also be used to avoid the duplication or omission of positions within a solution, caused by crossover operations.

Two common local search metaheuristics that make use of integer encodings and that have been applied to the TSP include the *simulated annealing* and *2-opt* algorithms. Both these algorithms will now be discussed, after which different applications where these algorithms were applied to the job shop scheduling problem will be discussed.

Simulated annealing (SA)

As described in Section 2.3.2, SA was first introduced by Kirkpatrick *et al.* (1983), and is motivated by the annealing process of slowly cooling metals to increase strength. The annealing process provides a framework for optimisation of the properties of large and complex systems. Furthermore, Rardin (1998) states that SA algorithms control cycling by

Algorithm 5 General Simulated Annealing

1: Choose any starting feasible solution $x^{(0)}$, an iteration limit $t_{\max}$, a large initial temperature $q > 0$, random number r and temperature reduction percentage k
2: Set incumbent solution $\hat{x} \leftarrow x^{(0)}$, and $t \leftarrow 0$
3: **repeat**
4: Randomly choose a feasible move Δx, where two positions in the encoding are exchanged as $\Delta x^{(t+1)}$
5: Calculate objective function improvement for the movement of $\Delta x^{(t+1)}$
6: **if** Objective function improves or $r < e^{\frac{-\Delta Obj}{q}}$ **then**
7: Accept the move $\Delta x^{(t+1)}$
8: Update $x^{(t+1)} \leftarrow x^{(t)} + \Delta x^{(t+1)}$
9: **if** Objective function of $x^{(t+1)}$ is superior to incumbent solution **then**
10: $\hat{x} \leftarrow x^{(t+1)}$
11: **end if**
12: **end if**
13: **if** Temperature change condition is satisfied **then**
14: Reduce temperature $q * k$
15: **end if**
16: $t \leftarrow t + 1$
17: **until** Termination condition
18: **output** Incumbent solution $\hat{x}$ as an approximate optimal solution

accepting non-improving moves according to probabilities tested with computer-generated random numbers. The SA algorithm is outlined in Algorithm 5.

Examples of the use of the SA algorithm on the job shop scheduling problem include a paper by Yazdani *et al.* (2009) in which they addressed the flexible job shop scheduling problem by minimising makespan. They proposed a simulated annealing algorithm to solve this NP-hard problem. To evaluate the performance of their algorithm, 20 benchmark problems from literature were used. Van Laarhoven *et al.* (1992) used a generalisation of the simulated annealing algorithm to find the minimum makespan in a job shop scheduling problem. The authors found that the simulated annealing algorithm produced shorter makespans than the tailored heuristic of Adams *et al.* (1988), at the cost of large computational time.

Other implementations of the SA algorithm include a study by Ogbu and Smith (1990), where the objective was to minimise the makespan in a flow shop. At the time when the study was executed, scheduling problems had received less attention in the literature

than other combinatorial problems. In the study, Ogbu and Smith propose a modification to the Metropolis scheme which is simpler to use and whose results provide near-optimal schedules in reasonable computation time. In another study, Suresh and Mohanasundaram (2006) investigated a multi-objective job shop scheduling problem with the objectives of minimising the makespan and the mean flow time of jobs. The study proposed the Pareto archived SA algorithm to discover non-dominated solution sets for the job shop scheduling problems. A segment-random insertion scheme was used to generate a set of neighbourhood solutions to the current solution. The performance of the proposed algorithm was also evaluated by solving benchmark job shop scheduling problems.

Zhang *et al.* (2008) developed a heuristic search approach that combines the SA and tabu search strategy. This approach makes use of SA to find the elite solutions, while using tabu search to reintensify the search from the promising solutions. The approach was also used to solve the objective of minimising the makespan of jobs. In another study Andresen *et al.* (2008) investigated the SA algorithm, where several neighbourhoods were suggested and tested together with the control parameters of the algorithm. The study also included extensive computational results for problems with up to 50 jobs and 50 machines. Lastly, a study by Zhang and Wu (2010) was performed, where a hybrid SA algorithm based on a novel immune mechanism is proposed for the job shop scheduling problem. The objective of this problem was to minimise the total weighted tardiness of jobs. The study is motivated by bottleneck jobs, which require more intensive optimisation. Therefore, the bottleneck level (also known as the criticality) for each job is evaluated, to quantitatively describe the bottleneck job distribution. From the applications provided above, it can be deduced that the use of the SA algorithm to solve the job shop scheduling problem, has been extensively researched and has proved to provide a solution for the problem.

2-opt

The 2-opt is a simple local search algorithm that was first proposed by Croes (1958) to solve the TSP, where two edges of a tour are deleted, thus breaking the tour into two paths, and then reconnects those paths in the other possible way. Figure 6.1 illustrates how a 2-opt move is applied to a fictional tour. In this tour, branches (a,b) and (c,d) were deleted, and replaced by (a,d) and (b,c). This figure is only a schematic; if distances were used as shown in the figure, this particular 2-change would be counterproductive and would not be performed. The findings of Tarantilis and Kiranoudis (2002) also confirm that the 2-opt move eliminates two branches and replaced them with the newly formed branches. In their example, the branches $(i,i+1)$ and $(j,j+1)$ were eliminated, and replaced with (i,j) and $(i+1,j+1)$. This means that all the branches between the eliminated branches are

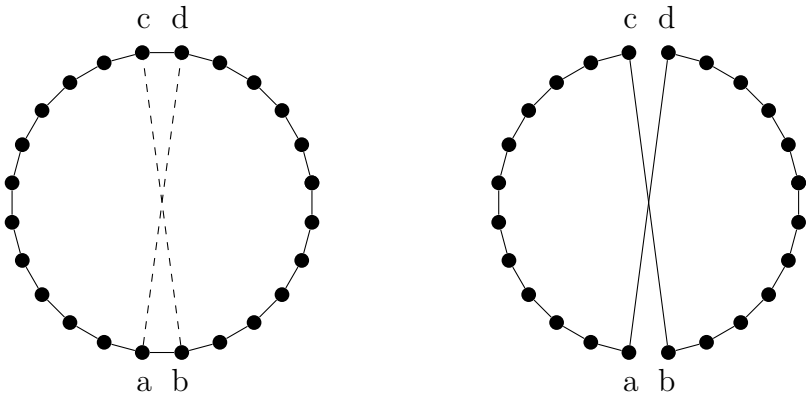


Figure 6.1: A 2-opt move (Johnson and McGeoch, 1995).

reversed. From the formulation of the 2-opt algorithm by Heragu and Alfa (1992) and by incorporating the definitions provided above, the 2-opt algorithm is outlined in Algorithm 6.

Algorithm 6 2-opt

Set $S \leftarrow$ initial solution; $z \leftarrow$ corresponding objective function value; $i \leftarrow 1$, $j \leftarrow i+1$
repeat
    Determine the positions of facilities i and j
    Perform move by reversing the positions of the facilities from i to j
    Calculate the new objective function value z'
    if $z' \leq z$ **then**
        Set $S \leftarrow S'$, $z \leftarrow z'$, $i \leftarrow 1$ and $j \leftarrow i+1$
    else
        $j \leftarrow j+1$
        if $j \geq n$ **then**
            Set $i \leftarrow i+1$ and $j \leftarrow i+1$
        end if
    end if
until $i \geq n-1$
output S

Examples where the 2-opt algorithm was applied to scheduling problems include a paper by Tarantilis and Kiranoudis (2002) which described a new metaheuristic for solving the job shop scheduling problem of process plants, which they termed as *list-based threshold accepting* (LBTA) method. This metaheuristic used the 2-opt local search heuristic to find neighbours of the current solution, which could convert one feasible schedule into another.

In another paper by Bauer *et al.* (1999), the authors used an ant colony optimisation approach to solve a single machine scheduling problem. They also applied a 2-opt-like strategy to the solutions generated by the artificial ants before updating the pheromone trail. Merkle *et al.* (2002) used an ant colony optimisation algorithm to solve for a resource-constrained project scheduling problem. The authors also investigated the use of the 2-opt move as a local search strategy, to determine if it produced a better sequence. In another study, Zobolas *et al.* (2009) proposed a hybrid metaheuristic for the job shop scheduling problem. The proposed hybrid metaheuristic combines three other metaheuristics, *i.e.* Differential Evolution, Genetic Algorithm and Variable Neighbourhood Search (VNS). The authors incorporated the 2-opt move as part of the VNS algorithm to explore the neighbourhood of the population.

When reflecting on the application of the 2-opt algorithm on scheduling problems, it is evident that the 2-opt move is used in conjunction with other metaheuristics to solve these problems. It is also apparent that there are very few implementations of the 2-opt algorithm to solve scheduling problems, let alone job shop scheduling problems. This indicates a possible area where further research is required.

The researchers therefore decided to adapt and test these algorithms to be able to solve the job shop scheduling problem.

6.2 Formulating the metaheuristics

This section will serve to describe the formulation of the SA and 2-opt metaheuristics to solve the job shop scheduling problem defined in this project. First, the encoding required for these algorithms will be defined, after which the algorithms implemented in the simulation scheduler will be outlined.

6.2.1 Defining the encoding

As stated previously, it was decided to make use of an integer encoding to minimise the computational requirements and prevent the duplication or omission of numbers within the encoding. The *random key encoding scheme* proposed by Lin *et al.* (2010) as well as the integer encoding proposed by Liu *et al.* (2014), served as inspiration for the development of a new encoding for this project.

In the study by Liu *et al.* (2014), an example was presented which contains five jobs. The number of operations allocated to each job varies and is shown in Table 6.1. According to the authors, "the gene arrangement represents the sequence of job operations and it indicates that all the operations of a given job are represented by the same real-parameter,

Table 6.1: Operations allocated to each job in example by Liu *et al.* (2014).

Job number	Allocated operations	Encoding
J_1	$O_{1,1}$, $O_{1,2}$, $O_{1,3}$	111
J_2	$O_{2,1}$, $O_{2,2}$	22
J_3	$O_{3,1}$, $O_{3,2}$, $O_{3,3}$	333
J_4	$O_{4,1}$	4
J_5	$O_{5,1}$, $O_{5,2}$	55

which depends on a randomly generated probability; the gene arrangement is then interpreted according to the sequence of occurrence of job operations in a given chromosome". This means that the encoding consists of only job numbers, which may appear several times if the specific job has more than one operation. The first appearance of a job number represents the first operation of that job and so on. For example, when referring to J_1 in Table 6.1, it can be observed that the job consists of three operations, *i.e.* $O_{1,1}$, $O_{1,2}$ and $O_{1,3}$. The encoding for this job can therefore be represented by (111), where the first number refers to $O_{1,1}$, the second number refers to $O_{1,2}$ and the last number refers to $O_{1,3}$.

When this same arrangement is used for all jobs to create a single encoding, it can be written as (11122333455). This sequence of operations of this encoding can then interpreted as $O_{1,1}$, $O_{1,2}$, $O_{1,3}$, $O_{2,1}$, $O_{2,2}$, $O_{3,1}$, $O_{3,2}$, $O_{3,3}$, $O_{4,1}$, $O_{5,1}$, $O_{5,2}$, which is a feasible sequence due to all the precedence constraints that are adhered to.

Lin *et al.* (2010) outlined a similar encoding structure and used an example with three jobs, each with two operations. They refer to an encoding written as (1, 2, 2, 3, 3, 1), which is then scanned from left to right. They state that "the first 1 means the first operation of job 1, corresponding to o_{11}, the second 1 means the second operation of job 1, corresponding to o_{12}". The sequence of the encoding containing all the operations can therefore be interpreted as ($o_{1,1}$, $o_{1,2}$, $o_{2,1}$, $o_{2,2}$, $o_{3,1}$, $o_{3,2}$). Furthermore, the authors state that "the operation sequence represented by this encoding scheme is always a feasible solution of job shop scheduling problems".

Taking the encoding structures provided in these studies into account, it was finally decided to use the same structure as proposed by Liu *et al.* (2014). The generation of the encoding will, however, be done randomly. The pseudo code used for generating an encoding is provided in Algorithm 7. First, the number of operations per job is counted, after which the operation selection probability is determined for each job. Thereafter, a random variable (*i.e.* r) is generated, which will be used to determine the job number that needs to be added to the encoding. Then a for loop starts which will run from one to the

Algorithm 7 Generating an encoding

repeat
  Count the number of operations per job
  Determine operation selection probability with $\frac{number\ of\ operations\ of\ job\ i}{total\ number\ of\ operations}$ for each job
  Generate a random variable r, where $0 \leq r \leq 1$
  for $i = 1$ *to number of jobs* **do**
    if $r <$ *selection probability of job* i **then**
      Add job number to encoding
      exit loop
    end if
  next i
until All operations are assigned
output Encoding

number of jobs there are in the system. For each iteration, the random variable will be compared to the selection probability. If the random variable is smaller than the selection probability of a job number, then that job number is added to the encoding and the count of operations for that job decreases by one. Once the selection is made, the for loop must be exited. These steps will then repeat until all the operations are assigned in the encoding.

Next the formulation of the SA and 2-opt algorithms to solve the job shop scheduling problem in this project will be discussed.

6.2.2 Formulating the 2-opt algorithm

Englert *et al.* (2014) state that the "2-Opt can take an exponential number of steps before it finds a locally optimal solution", which could pose a problem in the job shop environment if it is expected that a new schedule should be generated quickly. Therefore, the researcher proposed an Algorithm that generates a population of 50 solutions, and where each iteration of the algorithm only performs one 2-opt move per solution. This will inherently mean that the computation time will decrease, while still searching vast areas of the solution space. The 2-opt algorithm formulated to solve the job shop scheduling problem in this project is outlined in Algorithm 8.

The algorithm starts off with the creation of the initial population which has a size of 50 solutions. Algorithm 7 is used to create the solutions. Next, the stopping condition value is set to 10 minutes or "10:00", while the *IterationCount* variable is set to one. Thereafter, the objective function value is calculated for each solution. The best performing solution

Algorithm 8 2-opt move algorithm

1: Generated initial population of 50 solutions with encoding algorithm
2: Set $d \leftarrow$ stopping condition value, $IterationCount \leftarrow 1$
3: **repeat**
4: Calculate the objective function value of each solution
5: Set $z \leftarrow$ incumbent best objective function value of population
6: Set $S \leftarrow$ incumbent best solution
7: **if** $\frac{\Delta z}{10} \leq d$ **then**
8: Stop algorithm
9: **end if**
10: **for** $k = 1$ *to population size* **do**
11: Generate random variables i and j, where $j > i$
12: Perform 2-opt move by reversing numbers of encoding from i to j
13: **next** k
14: $IterationCount \leftarrow IterationCount + 1$
15: **until** $IterationCount = 100$
16: **output** z

will be stored in S, while the objective function value of the solution will be stored in z. Next, a test will be performed to determine if the average change in z over the past ten iterations is less than d. If this occurs, then the algorithm is stopped and the incumbent best solution z is presented. Otherwise, a *for* loop is initiated which will loop through each solution and generate two random variables (*i.e.* i and j). The 2-opt move will then be performed on each solution by reversing the positions of numbers in the encoding from i to j. The random variables i and j will be different for each solution, which will enable the algorithm to search a larger area in the solution space. The $IterationCount$ variable will then also be incremented by one. If the stopping criteria is never met, then the algorithm will run until it reaches 100 iterations, after which the incumbent best solution z is presented.

6.2.3 Formulating the Simulated Annealing algorithm

Due to several studies using the 2-opt move as part of the SA algorithm to solve both travelling salesman and scheduling problems, the researcher decided to incorporate the traditional SA as well as a hybrid SA which uses the 2-opt move as local search, to solve the scheduling problem presented in this project. The traditional SA will first be formulated

and described.

The traditional SA algorithm that was formulated to solve the scheduling problem defined in this project, is outlined in Algorithm 9. The algorithm starts off with the generation of a feasible solution $x^{(0)}$, an iteration limit $t_{\max}$ which is set to 100 and an initial temperature q set to 500 000. The reason why such a large initial temperature was chosen, was to encourage the acceptance of a weaker solution at earlier iterations. If any smaller temperature was chosen, then weaker solutions would not have been accepted. Thereafter, the incumbent solution $\hat{x}$ is set to $x^{(0)}$, incumbent best objective function value $\hat{z}$ is set to $z^{(0)}$ and the iteration count t is set to 0. Random variable i and j will then be generated and used to perform the swap of numbers in the encoding. When the swap is completed, the new solution is stored in $x^{(t+1)}$ and objective function value of this solution is calculated and stored in $z^{(t+1)}$. A test will then be performed to determine if $z^{(t+1)}$ is less than the incumbent best objective function value $\hat{z}$, or if the random variable r is

Algorithm 9 Formulated Simulated Annealing algorithm

1: Generate an initial feasible solution $x^{(0)}$, an iteration limit $t_{\max} \leftarrow 100$, an initial temperature $q \leftarrow 500\,000$
2: Set incumbent solution $\hat{x} \leftarrow x^{(0)}$, incumbent best objective function value $\hat{z} \leftarrow z^{(0)}$ and $t \leftarrow 0$
3: **repeat**
4: Generate random variables i, j and r
5: Perform the swap of numbers i and j in encoding
6: Set $x^{(t+1)} \leftarrow$ newly created solution
7: Calculate objective function value of $x^{(t+1)}$ as $z^{(t+1)}$
8: **if** $z^{(t+1)} < \hat{z}$ *or* $r < e^{\frac{-\Delta Obj}{q}}$ **then**
9: Accept the move
10: Set $\hat{x} \leftarrow x^{(t+1)}$ and $\hat{z} \leftarrow z^{(t+1)}$
11: **end if**
12: $t \leftarrow t + 1$
13: **if** $\dfrac{\Delta z}{10} \leq d$ **then**
14: Stop algorithm
15: **else if** *Temperature change condition is satisfied* **then**
16: Set $q \leftarrow q * 0.7$
17: **end if**
18: **until** $t = t_{\max}$
19: **output** Incumbent solution $\hat{x}$ as an approximate optimal solution

Algorithm 10 Hybrid Simulated Annealing algorithm

1: Generate an initial feasible solution $x^{(0)}$, an iteration limit $t_{\max} \leftarrow 100$, and an initial temperature $q \leftarrow 500\,000$
2: Set incumbent solution $\hat{x} \leftarrow x^{(0)}$, incumbent best objective function value $\hat{z} \leftarrow z^{(0)}$ and $t \leftarrow 0$
3: **repeat**
4: Generate random variables i and j, where $j > i$
5: Perform 2-opt move by reversing numbers of encoding from i to j
6: Set $x^{(t+1)} \leftarrow$ newly created solution
7: Calculate objective function value of $x^{(t+1)}$ as $z^{(t+1)}$
8: **if** $z^{(t+1)} < \hat{z}$ *or* $r < e^{\frac{-\Delta Obj}{q}}$ **then**
9: Accept the move
10: Set $\hat{x} \leftarrow x^{(t+1)}$ and $\hat{z} \leftarrow z^{(t+1)}$
11: **end if**
12: $t \leftarrow t + 1$
13: **if** $\dfrac{\Delta z}{10} \leq d$ **then**
14: Stop algorithm
15: **else if** *Temperature change condition is satisfied* **then**
16: Set $q \leftarrow q * 0.7$
17: **end if**
18: **until** $t = t_{\max}$
19: **output** Incumbent solution $\hat{x}$ as an approximate optimal solution

less than $e^{\frac{-\Delta Obj}{q}}$. When these tests are satisfied, the move is accepted and the incumbent best solution $\hat{x}$ is set to $x^{(t+1)}$ and the incumbent best objective function value $\hat{z}$ is set to $z^{(t+1)}$. The iteration number t is then incremented by one. The same stopping criteria as in Algorithm 8 is tested and if it is satisfied, the algorithm will stop and present the incumbent best solution $\hat{x}$; however, if it is not satisfied the temperature change condition is tested. The temperature change condition was chosen to be that the temperature should decrease if the number of iterations t is a multiple of ten. When this condition is met, the temperature is decreased to 70 percent of the current temperature. If the stopping criteria is never reached, the algorithm will keep on iterating until the iteration count t is equal to $t_{\max}$, after which the incumbent best solution $\hat{x}$ will be presented. This concludes the formulation of the traditional SA algorithm. Next, the hybrid SA algorithm incorporating the 2-opt move as local search scheme will be discussed.

The hybrid SA algorithm incorporating the 2-opt move as local search scheme, is out-

lined in Algorithm 10, where the only difference is in step 5. The algorithm was constructed exactly the same way that the traditional SA algorithm was formulated; however, the move that is applied does not only change two numbers in the entire encoding. The 2-opt move is applied instead, which reverses all the numbers in the encoding from i to j, where i and j are randomly generated variables. The same acceptance, stopping and temperature change tests are also applied to this algorithm as in the traditional SA algorithm. The 2-opt move is a more aggressive move, which will enable this algorithm to search a wider area in the solution space; however, it can also produce greater variance in the objective function values. This concludes the formulation of the metaheuristics that will be used to solve the job shop scheduling problem presented in this project. The following section will be used to describe the test scenarios as well as the results of the tests.

6.3 Metaheuristic testing

In this section, the performance of the metaheuristics will be compared to each other as well as the dispatching rules. Only a single objective optimisation problem will be considered, where the objective function will be to minimise the makespan of the schedule. Before comparing the outcomes of the tests, the test scenarios have to be defined. It was decided to make use of three test scenarios, and they are defined as:

- **Test scenario 6:** 50 jobs are entered into the system, each with different expected processing times which can vary from short to long.
- **Test scenario 7:** 100 jobs are entered into the system, each with different expected processing times which can vary from short to long.
- **Test scenario 8:** 200 jobs are entered into the system, each with different expected processing times which can vary from short to long.

In Appendix B, the test scenarios were defined in such a way that a job cannot be processed by the same machine more than once. This restriction was however lifted for the three new test scenarios. For that reason, if a job has to be milled followed by grinding, it will be able to go back to the milling machine if another milling operation is required. This is a more realistic assumption, which is why the researcher removed the constraint. The reason why the size of the test scenarios are 50, 100 and 200, is to be able to generate feasible schedules which is not possible by hand. It is not worth the computational costs to generate schedules for smaller problems, because relatively good schedules can be generated without the use of metaheuristics. Therefore, larger problems were defined which will require a

metaheuristic to find a good solution. The data that was entered into the system for each test scenario will not be provided, but the comparison of the results will be described in detail. Furthermore, each experiment that will be run is linked to a specific dispatching rule or metaheuristic, as seen in Table 6.2.

Table 6.2: Description of experiments

Experiment number	**Description**
1	Shortest processing time
2	First-come-first-serve
3	Most-important-job-first
4	Earliest due date
5	Critical ratio
6	Minimum slack time
7	2-opt move
8	Traditional Simulated Annealing
9	Simulated Annealing with 2-opt

6.3.1 Test scenario 6 results

The results that were obtained by simulating Test scenario 6 for each of the dispatching rules as well as the metaheuristics will now be discussed. When referring to the confidence intervals and the ANOVA table, as illustrated in Figures 6.2 and 6.3, of the *makespan* KPI, it can be deduced that the dispatching rules, represented by experiments 1 to 6, were outperformed by all three proposed metaheuristics. The 2-opt move algorithm performed the best of all the rules and metaheuristics. The confidence intervals and the ANOVA table suggests that the hybrid SA performed better than the traditional SA, and that there is statistical difference between them. Table 6.3 shows the results of each dispatching rule and metaheuristic for the *makespan* KPI in terms of "Days:Hours:Minutes:Seconds".

Table 6.3: Results of Test scenario 6

Experiment number	Makespan	Standard deviation	Minimum	Maximum
1: Shortest processing time	25:15:32:55	1:10:06:46	21:19:08:12	28:19:26:28
2: First-come-first-serve	28:12:30:45	22:44:33	26:16:09:44	31:16:43:44
3: Most-important-job-first	28:11:14:58	17:50:35	26:20:13:05	30:20:47:12
4: Earliest due date	27:22:49:36	1:00:49:44	26:02:34:00	30:03:55:04
5: Critical ratio	27:09:37:36	1:18:46:12	24:01:58:15	33:02:24:36
6: Minimum slack time	32:13:13:36	1:06:18:51	29:02:50:01	35:03:47:50
7: 2-opt move	11:13:01:36	21:50:59	10:21:11:39	15:16:18:56
8: Traditional SA	14:10:24:22	17:39:08	12:16:21:21	16:16:41:04
9: Hybrid SA	13:10:35:27	1:01:03:25	11:16:12:57	17:17:55:25

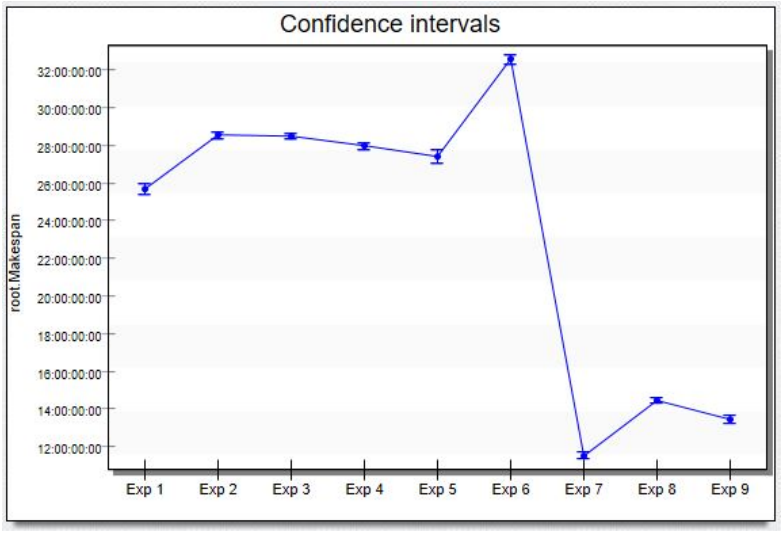


Figure 6.2: Confidence intervals for the makespan KPI of Test scenario 6

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6	Exp 7	Exp 8	Exp 9
Exp 1	0	0	0	0	0	0	0	0
Exp 2		0.663	0	0	0	0	0	0
Exp 3			0	0	0	0	0	0
Exp 4				0.008	0	0	0	0
Exp 5					0	0	0	0
Exp 6						0	0	0
Exp 7							0	0
Exp 8								0

Figure 6.3: ANOVA table for the makespan KPI of Test scenario 6

The metaheuristics could then also be compared when referring to the number of iterations and amount of time it took to find the incumbent best solution. When the time to find the incumbent best solution is compared, the 2-opt move algorithm took between one and two minutes to complete, while both the SA algorithms took less than ten seconds to complete. This is understandable, due to the large population size of the 2-opt move algorithm, while both SA algorithms use only one solution at a time. Next, when the number of iterations required to find the incumbent best solution for each metaheuristic is compared, it was found that the 2-opt move algorithm only required 30 iterations, while it took the traditional and hybrid SA algorithms 100 iterations each. Figure 6.4 illustrates how each metaheuristic changed the objective function value over the 100 iterations. The 2-opt move only accepted improved solutions, while both SA algorithms accepted weaker solutions at the start, but converged to a solution by the end.

The results that were obtained by the one replication per metaheuristic, only indicates which metaheuristic performed best for that replication. To get a better indication of which metaheuristic performs best, more replications need to be completed. The results from these replications can then be compared by using paired t-test, as explained in Montgomery and Runger (2013) and Law (2007). Law goes on to explain that the paired t-test is used to compare the performance measure of two systems by forming a confidence interval for the difference in the two expectations, rather than performing a hypothesis test. The interval will not only provide the "reject" or "fail-to-reject" conclusion, but will also quantify how much the expectations differ. When applying the paired t-test, normality is assumed, because it is simple, familiar and should be quite robust in this context (Law, 2007).

It was decided to perform 10 replications per metaheuristic, after which the paired t-tests will be done. The t-tests that were performed are 2-opt move algorithm versus traditional SA, 2-opt move algorithm versus hybrid SA, and traditional SA versus hybrid SA.

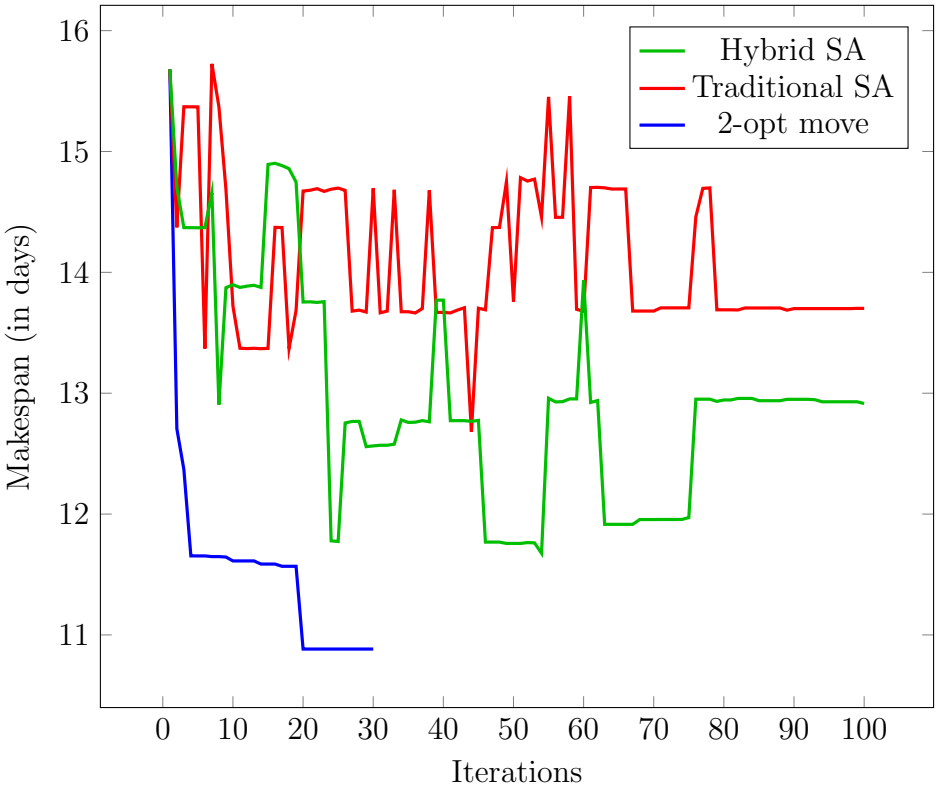


Figure 6.4: Iterations of each metaheuristic for Test scenario 6

Paired *t*-test of 2-opt move algorithm *vs* traditional SA

The mean difference ($\bar{d}$) of the 2-opt move algorithm and traditional SA makespan values, was calculated to be -1.373 and the standard deviation (s_d) was calculated as 0.703. With these values, the 95% confidence interval was calculated to be

$$-0.871 \leq \mu_d \leq -1.876,$$

where $\mu_d = \mu_{\text{2-opt move}} - \mu_{\text{traditional SA}}$. The confidence interval on μ_d does not include zero, which implies that at a 95% level of confidence, the data supports the claim that the 2-opt move algorithm and the traditional SA have different mean makespan values. The confidence interval also implies that the 2-opt move algorithm performed better than the traditional SA.

Paired *t*-test of 2-opt move algorithm *vs* hybrid SA

The mean difference ($\bar{d}$) of the 2-opt move algorithm and hybrid SA makespan values, was calculated to be -1.107 and the standard deviation (s_d) was calculated as 0.725. With these values, the 95% confidence interval was calculated to be

$$-1.626 \leq \mu_d \leq -0.589,$$

where $\mu_d = \mu_{\text{2-opt move}} - \mu_{\text{hybrid SA}}$. The confidence interval on μ_d does not include zero, which implies that at a 95% level of confidence, the data supports the claim that the 2-opt move algorithm and the hybrid SA have different mean makespan values. The confidence interval also implies that the 2-opt move algorithm performed better than the hybrid SA.

Paired *t*-test of traditional SA *vs* hybrid SA

The mean difference ($\bar{d}$) of the traditional SA and hybrid SA makespan values, was calculated to be 0.266 and the standard deviation (s_d) was calculated as 0.425. With these values, the 95% confidence interval was calculated to be

$$-0.038 \leq \mu_d \leq 0.570,$$

where $\mu_d = \mu_{\text{traditional SA}} - \mu_{\text{hybrid SA}}$. The confidence interval on μ_d does include zero, which implies that at a 95% level of confidence, the data does not support the claim that the traditional SA and the hybrid SA have different mean makespan values. The confidence interval also implies that the traditional SA and the hybrid SA performed similarly.

6.3.2 Test scenario 7 results

The results that were obtained by simulating Test scenario 7 for each of the dispatching rules as well as the metaheuristics will now be discussed. When referring to the confidence intervals and the ANOVA table, as illustrated in Figures 6.5 and 6.6, of the *makespan* KPI, it can be deduced that the dispatching rules, represented by experiments 1 to 6, were outperformed by all three proposed metaheuristics. The 2-opt move algorithm performed the best of all the rules and metaheuristics. For this test scenario, the hybrid SA performed better than the traditional SA and the ANOVA table indicates that there is statistical difference between the two results. Table 6.4 shows the results of each dispatching rule and metaheuristic for the *makespan* KPI in terms of "Days:Hours:Minutes:Seconds".

Table 6.4: Results of Test scenario 7

Experiment number	Makespan	Standard deviation	Minimum	Maximum
1: Shortest processing time	54:10:59:19	3:11:05:07	46:20:21:22	61:20:06:46
2: First-come-first-serve	52:18:41:17	1:09:38:21	48:17:32:17	55:18:18:01
3: Most-important-job-first	51:06:38:52	1:09:23:55	47:16:50:27	55:22:05:46
4: Earliest due date	55:13:06:56	1:21:24:14	51:16:50:00	61:17:34:31
5: Critical ratio	54:14:24:05	2:07:59:01	49:22:06:24	60:21:47:42
6: Minimum slack time	58:09:04:28	1:13:52:24	55:17:03:36	61:17:40:55
7: 2-opt move	19:05:33:11	21:19:05	18:09:16:23	22:08:49:47
8: Traditional SA	22:14:22:54	1:05:03:03	20:12:06:37	25:13:18:24
9: Hybrid SA	21:07:15:28	1:18:19:18	17:18:44:09	25:08:52:19

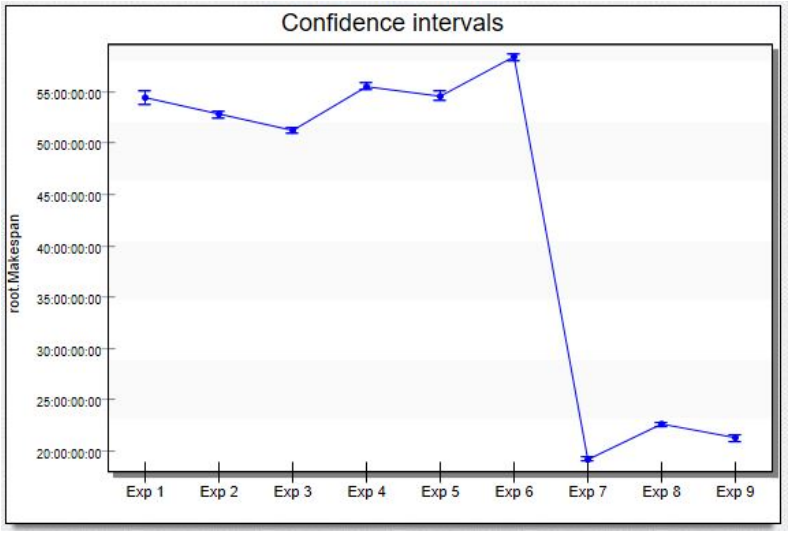


Figure 6.5: Confidence intervals for the makespan KPI of Test scenario 7

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6	Exp 7	Exp 8	Exp 9
Exp 1	0	0	0.006	0.734	0	0	0	0
Exp 2		0	0	0	0	0	0	0
Exp 3			0	0	0	0	0	0
Exp 4				0.002	0	0	0	0
Exp 5					0	0	0	0
Exp 6						0	0	0
Exp 7							0	0
Exp 8								0

Figure 6.6: ANOVA table for the makespan KPI of Test scenario 7

The metaheuristics could then also be compared when referring to the number of iterations and amount of time it took to find the incumbent best solution. When the time to find the incumbent best solution is compared, the 2-opt move algorithm took less than four minutes to complete, while both the SA algorithms took less than 20 seconds to complete. This is understandable, due to the large population size of the 2-opt move algorithm, while both SA algorithms use only one solution at a time. Next, when the number of iterations required to find the incumbent best solution for each metaheuristic is compared, it was found that the 2-opt move only required 30 iterations, while it took the traditional and hybrid SA

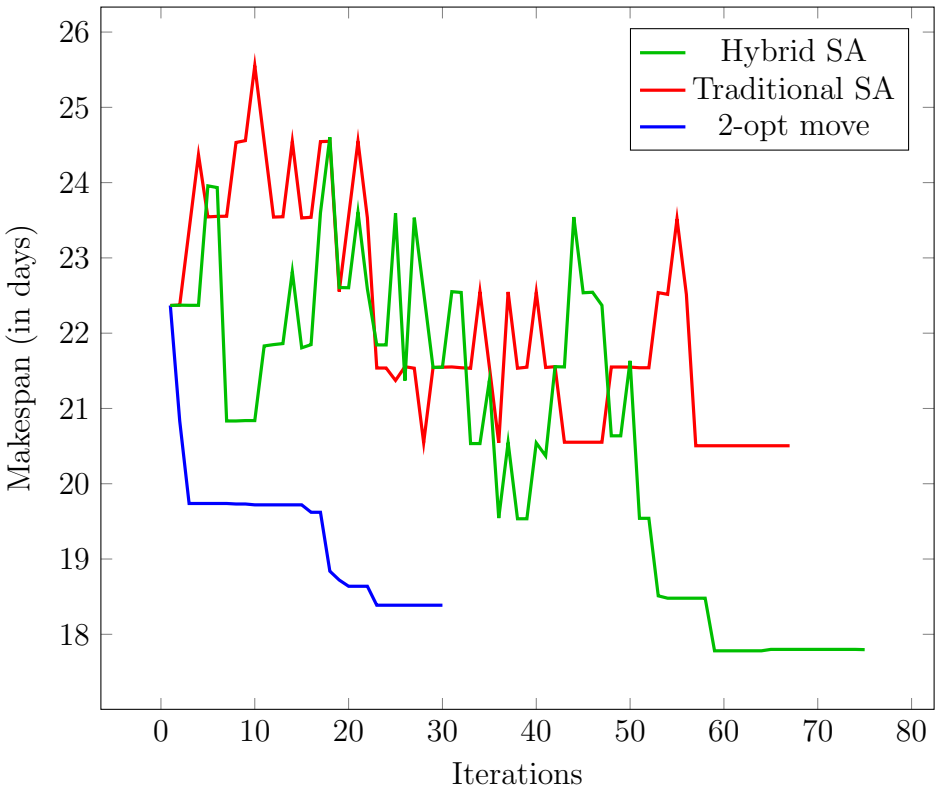


Figure 6.7: Iterations of each metaheuristic for Test scenario 7

algorithms 67 and 81 iterations respectively. Figure 6.7 illustrates how each metaheuristic changed the objective function value over the maximum of 100 iterations. The 2-opt move only accepted improved solutions, while both SA algorithms accepted weaker solutions at the start, but converged to a solution by the end.

It was again decided to perform 10 replications per metaheuristic, after which the paired t-tests will be done. The t-tests that were performed are 2-opt move algorithm versus traditional SA, 2-opt move algorithm versus hybrid SA, and traditional SA versus hybrid SA.

Paired t-test of 2-opt move algorithm *vs* traditional SA

The mean difference ($\bar{d}$) of the 2-opt move algorithm and traditional SA makespan values, was calculated to be -1.702 and the standard deviation (s_d) was calculated as 1.097. With these values, the 95% confidence interval was calculated to be

$$-2.487 \leq \mu_d \leq -0.918,$$

where $\mu_d = \mu_{\text{2-opt move}} - \mu_{\text{traditional SA}}$. The confidence interval on μ_d does not include zero, which implies that at a 95% level of confidence, the data supports the claim that the 2-opt move algorithm and the traditional SA have different mean makespan values. The confidence interval also implies that the 2-opt move algorithm performed better than the traditional SA.

Paired t-test of 2-opt move algorithm *vs* hybrid SA

The mean difference ($\bar{d}$) of the 2-opt move algorithm and hybrid SA makespan values, was calculated to be -1.759 and the standard deviation (s_d) was calculated as 0.825. With these values, the 95% confidence interval was calculated to be

$$-2.349 \leq \mu_d \leq -1.169,$$

where $\mu_d = \mu_{\text{2-opt move}} - \mu_{\text{hybrid SA}}$. The confidence interval on μ_d does not include zero, which implies that at a 95% level of confidence, the data supports the claim that the 2-opt move algorithm and the hybrid SA have different mean makespan values. The confidence interval also implies that the 2-opt move algorithm performed better than the hybrid SA.

Paired t-test of traditional SA *vs* hybrid SA

The mean difference ($\bar{d}$) of the traditional SA and hybrid SA makespan values, was calculated to be -0.056 and the standard deviation (s_d) was calculated as 0.618. With these values, the 95% confidence interval was calculated to be

$$-0.499 \leq \mu_d \leq 0.386,$$

where $\mu_d = \mu_{\text{traditional SA}} - \mu_{\text{hybrid SA}}$. The confidence interval on μ_d does include zero, which implies that at a 95% level of confidence, the data does not support the claim that the traditional SA and the hybrid SA have different mean makespan values. The confidence interval also implies that the traditional SA and the hybrid SA performed similarly.

6.3.3 Test scenario 8 results

The results that were obtained by simulating Test scenario 8 for each of the dispatching rules as well as the metaheuristics will now be discussed. When referring to the confidence intervals and the ANOVA table, as illustrated in Figures 6.8 and 6.9, of the *makespan* KPI, it can be deduced that the dispatching rules, represented by experiments 1 to 6, were outperformed by all three proposed metaheuristics. The 2-opt move algorithm performed the best of all the rules and metaheuristics. For this test scenario, the traditional SA performed better than the hybrid SA and the ANOVA table indicates that there is statistical difference between the two results. Table 6.5 shows the results of each dispatching rule and metaheuristic for the *makespan* KPI in terms of "Days:Hours:Minutes:Seconds".

Table 6.5: Results of Test scenario 8

Experiment number	**Makespan**	**Standard deviation**	**Minimum**	**Maximum**
1: Shortest processing time	107:15:37:42	4:17:11:00	97:22:14:55	119:21:41:41
2: First-come-first-serve	107:11:52:27	2:00:19:48	101:11:59:45	112:23:11:05
3: Most-important-job-first	108:07:43:37	1:22:03:46	101:12:16:20	112:12:11:35
4: Earliest due date	111:22:02:34	3:00:36:44	104:20:00:31	119:20:36:53
5: Critical ratio	107:17:32:56	4:07:22:45	97:21:41:09	122:22:38:01
6: Minimum slack time	104:19:56:54	1:17:33:41	99:21:41:41	108:23:57:28
7: 2-opt move	34:16:46:25	1:07:34:02	33:13:03:34	37:13:05:49
8: Traditional SA	38:16:40:41	1:00:43:07	36:18:25:06	42:12:24:51
9: Hybrid SA	40:03:08:35	2:02:44:20	36:19:13:55	46:15:00:42

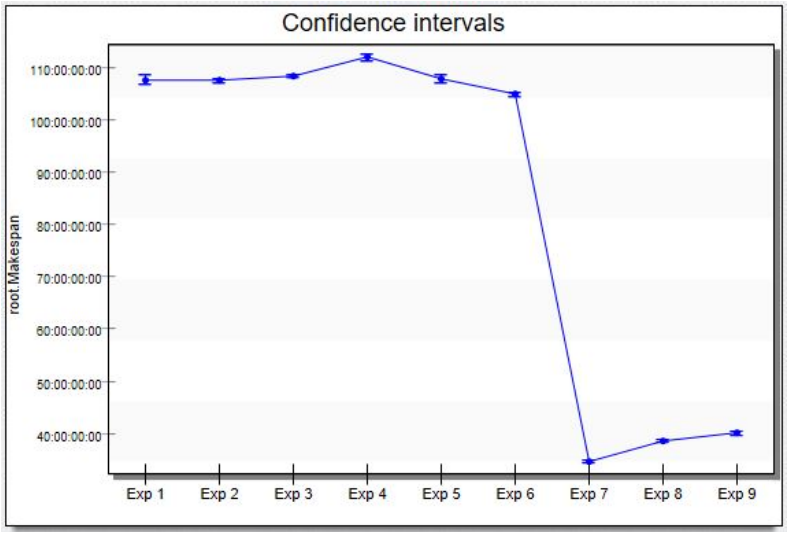


Figure 6.8: Confidence intervals for the makespan KPI of Test scenario 8

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6	Exp 7	Exp 8	Exp 9
Exp 1	0.761	0.19	0	0.9	0	0	0	0
Exp 2		0.003	0	0.62	0	0	0	0
Exp 3			0	0.212	0	0	0	0
Exp 4				0	0	0	0	0
Exp 5					0	0	0	0
Exp 6						0	0	0
Exp 7							0	0
Exp 8								0

Figure 6.9: ANOVA table for the makespan KPI of Test scenario 8

The metaheuristics could then also be compared when referring to the number of iterations and amount of time it took to find the incumbent best solution. When the time to find the incumbent best solution is compared, the 2-opt move algorithm took less than 20 minutes to complete, while both the SA algorithms took close to a minute to complete. This is understandable, due to the large population size of the 2-opt move algorithm, while both SA algorithms used only one solution at a time. Next, when the number of iterations required to find the incumbent best solution for each metaheuristic is compared, it was found that the 2-opt move only required 28 iterations, while it took the traditional and hybrid SA algorithms 100 and 71 iterations respectively. Figure 6.10 illustrates how each metaheuristic changed the incumbent best objective function value over the maximum of 100 iterations. The 2-opt move only accepted improved solutions, while both SA algorithms accepted weaker solutions at the start, but converged to a solution by the end.

It was again decided to perform 10 replications per metaheuristic, after which the paired t-tests will be done. The t-tests that were performed are 2-opt move algorithm versus tra-

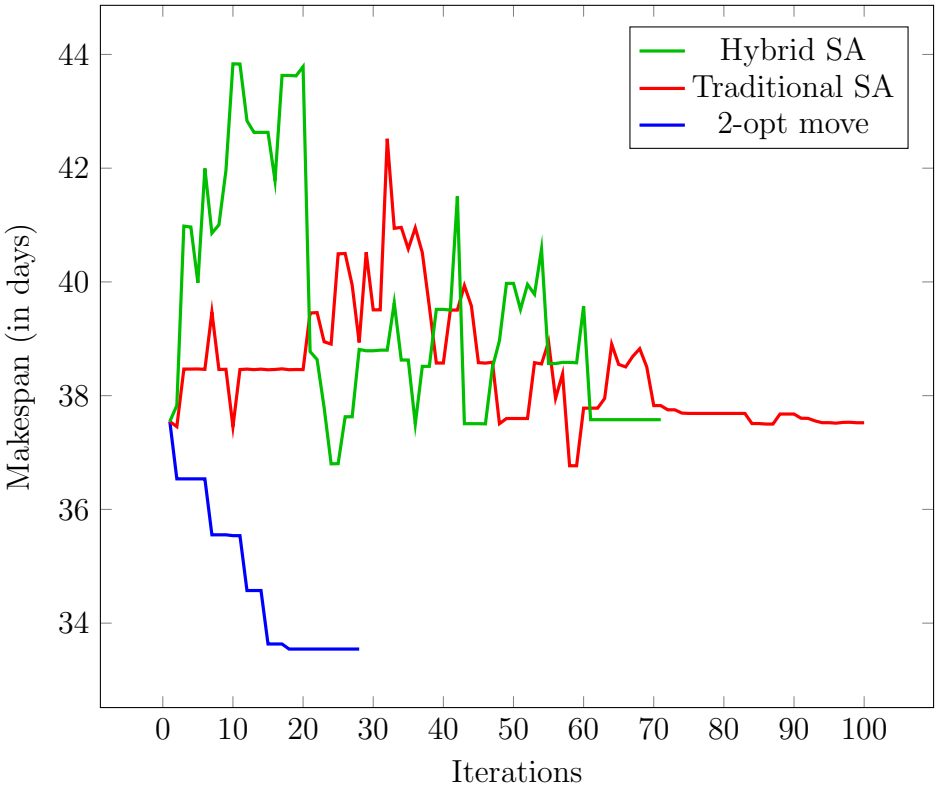


Figure 6.10: Iterations of each metaheuristic for Test scenario 8

ditional SA, 2-opt move algorithm versus hybrid SA, and traditional SA versus hybrid SA.

Paired *t*-test of 2-opt move algorithm *vs* traditional SA

The mean difference ($\bar{d}$) of the 2-opt move algorithm and traditional SA makespan values, was calculated to be -2.703 and the standard deviation (s_d) was calculated as 1.090. With these values, the 95% confidence interval was calculated to be

$$-3.483 \leq \mu_d \leq -1.923,$$

where $\mu_d = \mu_{\text{2-opt move}} - \mu_{\text{traditional SA}}$. The confidence interval on μ_d does not include zero, which implies that at a 95% level of confidence, the data supports the claim that the 2-opt move algorithm and the traditional SA have different mean makespan values. The confidence interval also implies that the 2-opt move algorithm performed better than the traditional SA.

Paired *t*-test of 2-opt move algorithm *vs* hybrid SA

The mean difference ($\bar{d}$) of the 2-opt move algorithm and hybrid SA makespan values, was calculated to be -3.539 and the standard deviation (s_d) was calculated as 1.624. With these

values, the 95% confidence interval was calculated to be

$$-4.700 \leq \mu_d \leq -2.377,$$

where $\mu_d = \mu_{\text{2-opt move}} - \mu_{\text{hybrid SA}}$. The confidence interval on μ_d does not include zero, which implies that at a 95% level of confidence, the data supports the claim that the 2-opt move algorithm and the hybrid SA have different mean makespan values. The confidence interval also implies that the 2-opt move algorithm performed better than the hybrid SA.

Paired *t*-test of traditional SA *vs* hybrid SA

The mean difference ($\bar{d}$) of the traditional SA and hybrid SA makespan values, was calculated to be -0.836 and the standard deviation (s_d) was calculated as 0.913. With these values, the 95% confidence interval was calculated to be

$$-1.489 \leq \mu_d \leq -0.182,$$

where $\mu_d = \mu_{\text{traditional SA}} - \mu_{\text{hybrid SA}}$. The confidence interval on μ_d does not include zero, which implies that at a 95% level of confidence, the data supports the claim that the traditional SA and the hybrid SA have different mean makespan values. The confidence interval also implies that the traditional SA performed better than the hybrid SA. This concludes the testing of the metaheuristics, which will be followed by a synthesis of the results.

6.4 Synthesis: Experiments

This section provides a reflection of the metaheuristics that were discussed, as well as the results that were obtained from the test scenarios. When the results were compared, it became evident that the metaheuristics performed considerably better than the dispatching rules, as the problem size increased. Therefore, it can be inferred that dispatching rules can either be used when the problem size is small or as a starting point for a metaheuristic that can find a good solution. Furthermore, from the paired t-tests, it became evident that the two SA algorithms achieved relatively similar incumbent best solutions for the test scenarios, in a short period of time. The 2-opt move algorithm performed the best of the rules and metaheuristics, which can be attributed to the large population size; however, it came at a larger computational cost with longer simulation times. The average computation time for each of the metaheuristics and test scenario, is provided in Table 6.6, where the time is illustrated in "Hours:Minutes:Seconds". The researcher therefore concludes that if the expert planner or user of the system has sufficient time, then the 2-opt move algorithm

should be used to generate a new schedule. Otherwise, the user should implement either the traditional SA or hybrid SA with the 2-opt move as local search scheme, because both algorithms performed considerably better with the same number of iterations as the dispatching rules.

Another point that needs to be highlighted, is the lack of literature that was found on the implementation of the 2-opt move in a hybrid algorithm to solve the job shop scheduling problem. This indicates that there is a possible gap in the literature and future research can be undertaken to not only incorporate the 2-opt move in a SA algorithm, but also other metaheuristics, to solve the job shop scheduling problem.

Table 6.6: Computation time of the three metaheuristics for each test scenario

Metaheuristic	**50 job test**	**100 job test**	**200 job test**
2-opt move algorithm	0:01:39	0:03:10	0:17:47
Traditional SA	0:00:10	0:00:14	0:01:05
Hybrid SA	0:00:10	0:00:17	0:00:58

6.5 Chapter summary

This chapter described the incorporation of metaheuristics in the simulation scheduler. The metaheuristics that were implemented are the *simulated annealing* and a variation of the *2-opt* algorithm. Detailed analysis and comparison of metaheuristics and dispatching rules were also provided. The following chapter will document the expansion of the simulation scheduler from the single-objective to multi-objective domain. Several metaheuristics will be incorporated to solve the stated multi-objective job shop scheduling problem.

Chapter 7

Formulation and testing of bi-objective simulation scheduler

This chapter documents the expansion of the simulation scheduler from the single-objective to the multi-objective optimisation domain. The chapter will start off with the selection and discussion of the objectives that are used in the scheduler. Thereafter, the metaheuristics that are incorporated into the simulation scheduler will be discussed. Comparison tests of the performance of each metaheuristic will then follow. Finally, the chapter will document the implementation of a newly developed ranking and selection technique, *MMY*, to determine the approximate Pareto set and guarantee probability of correct selection of the best simulated schedules.

7.1 Selection of objectives

To expand a model from the single-objective to the multi-objective domain, it is essential to select appropriate objectives that need to be optimised. Ngatchou *et al.* (2005) stated that a multi-objective problem has conflicting objectives. If the objectives are not conflicting, then the problem can be simplified to a single-objective problem and the global optimum can be determined. For this purpose, literature needed to be consulted to determine which objectives have previously been used in multi-objective JSPs. Examples of objectives that have been selected in literature include the makespan, flow time, total idle time of jobs and the total idle time of machines that were selected and compared in different bi-objective models by Moritz *et al.* (2015). Adibi *et al.* (2010) used the makespan and tardiness performance indicators as objectives in their study. Sha and Lin (2010) also used the makespan, total tardiness and total idle time performance indicators as objectives in their multi-objective JSP. These objectives do not, however, fit the description as stated by Ngatchou *et al.* (2005), as they are not in conflict with each other. If the makespan of a schedule is minimised, one would expect that the total tardiness of jobs, the idle time of machines and the idle time of jobs would also be minimised, and *vice versa*. This assumption was tested, and the output of the test confirmed that when the makespan of a schedule is minimised, the tardiness will also be minimised, as seen in Figure 7.1. The blue line in Figure 7.1 represents the correlation between the two objectives, and is clearly shown to be positive. The correlation coefficient was determined to be 0.738 by conducting a correlation test. The correlation coefficient suggests that the objectives are

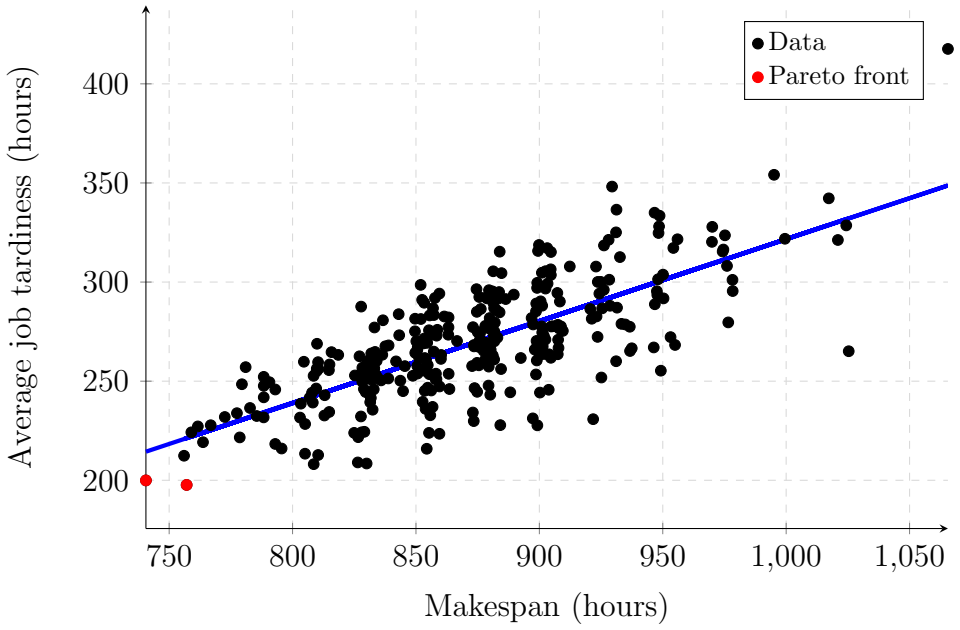


Figure 7.1: Correlation of makespan and average job tardiness as objectives

strongly related and are not in conflict with each other and should therefore not be used in conjunction in a multi-objective JSP.

Due to the assumption that preemption is not allowed, the processing step of a job must be finished without interruption. This can result in overtime if the processing step is still ongoing at the closing time of the shop. The researcher therefore decided on using *makespan* and *total overtime* as objectives. When the makespan of a schedule is minimised, it is expected that more overtime is required and that the total overtime will increase. On the other hand, if the total overtime is minimised, the makespan of the schedule will increase. This assumption was also tested by conducting a correlation test. The correlation coefficient was determined to be -0.066 which indicates that there is low correlation between these two objectives. The blue line in Figure 7.2 again represents the correlation between the two objectives, and this is clearly shown to be random which confirms that the two objectives are conflicting.

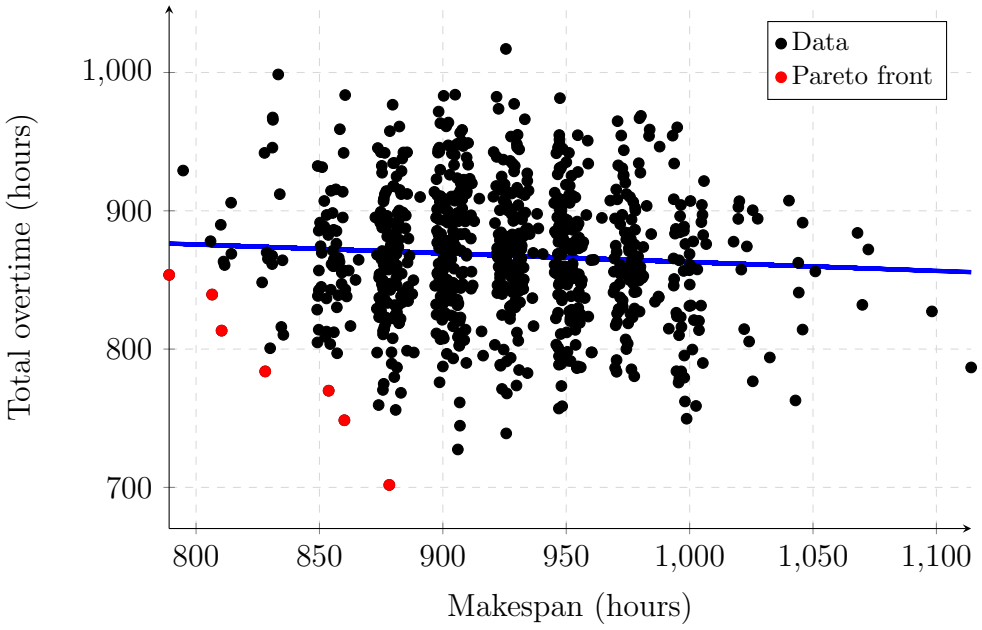


Figure 7.2: Correlation of makespan and total overtime as objectives

Overtime has previously been used as an objective in a multi-objective JSP, as seen in Rohaninejad *et al.* (2015) and Zhang *et al.* (2016). Rohaninejad *et al.* (2015) studied a multi-objective flexible JSP with machine capacity constraints. The authors used a hybrid genetic algorithm based on the ELECTRE method. Zhang *et al.* (2016) determined a robust schedule for a flexible JSP with flexible workdays, by using either a goal-guided multi-objective tabu search or a goal-guided multi-objective hybrid search. Both these studies did, however, assume deterministic processing times, which may not adequately mimic a real-world job shop, as processing times are stochastic. Therefore, what differentiates this research project from the previous studies, is the use of stochastic processing times. These studies also focused on smaller instances for testing, by limiting the number of jobs to 16 in Rohaninejad *et al.* (2015) and 50 in Zhang *et al.* (2016). This study is different because

- stochastic processing times are used, and
- three different test scenarios *i.e.* 50, 100 and 200 jobs, that must be processed on eight machines, are used.

These test scenarios are larger instances and will better represent the number of jobs that are in a real-world job shop. In the next section, the selection and implementation of the metaheuristics will be discussed.

7.2 Selection of metaheuristics

Due to the large amount of research that has gone into implementing different metaheuristics to solve the JSP, as observed in a survey by Chaudhry and Khan (2016), it is difficult to choose a single best approach. For this reason, five different metaheuristics were chosen for implementation to solve the bi-objective JSP presented in this research project. The five metaheuristics are the *Multi-objective Simulated Annealing* (MOSA) algorithm, which is a trajectory-based algorithm; the *Multi-objective 2-opt move* (MO2Opt) algorithm, which is a population-based algorithm that uses the 2-opt move as a local search strategy; the *Non-dominated Sorting Genetic Algorithm II* (NSGAII) and the *Multi-objective Genetic Algorithm* (MOGA), which are biologically inspired population-based algorithms; and the *Multi-objective Optimisation Cross Entropy Method* (MOOCEM), which is another population-based algorithm with a stochastic basis. These five metaheuristics were chosen to provide a diverse pool of methods that can be used to solve the bi-objective JSP and to ensure that the study did not rely on a single type of metaheuristic to provide a solution to the bi-objective JSP. This also presented the opportunity for extensive testing, to determine which metaheuristic performed best for the test scenarios that were created. A discussion of each metaheuristic will now follow.

7.2.1 Multi-objective simulated annealing

The MOSA algorithm is outlined in Algorithm 11. The algorithm starts off by generating a feasible solution $x^{(0)}$, by using Algorithm 7. Furthermore, the iteration limit $t_{\max}$ is set to 1 000 and the initial temperature q is set to 500 000. These values were selected to allow the algorithm enough time to search for good solutions. Thereafter, the incumbent solution $\hat{x}$ is set to $x^{(0)}$, incumbent best objective function values $\hat{z}_1$ and $\hat{z}_2$ are set to $z_1^{(0)}$ and $z_2^{(0)}$ respectively, and the iteration count t is set to 0. Random variables i and j will then be generated and used to perform the swap of numbers in the encoding. When the swap is completed, the new solution is stored in $x^{(t+1)}$ and objective function values of this solution are calculated and stored in $z_1^{(t+1)}$ and $z_2^{(t+1)}$. A test will then be performed to determine if $x^{(t+1)}$ dominates the incumbent best solution $\hat{x}$, or if the random variable r is less than $e^{\frac{-\Delta z_1}{q}}$ or $e^{\frac{-\Delta z_2}{q}}$. When these tests are satisfied, the move is accepted and the incumbent best solution $\hat{x}$ is set to $x^{(t+1)}$ and the incumbent best objective function values $\hat{z}_1$ and $\hat{z}_2$ are set to $z_1^{(t+1)}$ and $z_2^{(t+1)}$ respectively. The iteration number t is then incremented by one. There are two stopping criteria implemented in the algorithm, which stipulate that the algorithm must be stopped if the average incumbent best objective function value, for

each objective, is within 0.01 of the current incumbent best objective function value, or if the maximum iteration counter has been reached. The intervals

$$0.99\hat{z}_1 \leq d_1 \leq 1.01\hat{z}_1 \text{ and}$$

$$0.99\hat{z}_2 \leq d_2 \leq 1.01\hat{z}_2,$$

where d_1 and d_2 are the average incumbent best objective function value for each objective over the last 10 iterations, were chosen to ensure that the algorithm only stops once the current incumbent best objective function values have stagnated and no further improvement has been found. The stopping criteria will not be tested until at least 500 observations have been completed, which ensures that the algorithm has had sufficient opportunity to search the solution space. If the two stopping criteria are not satisfied the temperature change condition is tested. The temperature change condition was chosen to be that the temperature should decrease if the number of iterations t is a multiple of 20. When this condition is met, the temperature is decreased to 70 percent of the current temperature. Once either of the two stopping criteria is met, the approximate Pareto set of the incumbent best solutions will be presented. This concludes the formulation of the MOSA algorithm. Next, the MO2Opt algorithm will be discussed.

Algorithm 11 Multi-objective Simulated Annealing (MOSA)

Generate an initial feasible solution $x^{(0)}$, an iteration limit $t_{\max} \leftarrow 1\,000$, an initial temperature $q \leftarrow 500\,000$
Set incumbent solution $\hat{x} \leftarrow x^{(0)}$, incumbent best objective function value for each objective $\hat{z_1} \leftarrow z_1^{(0)}$ and $\hat{z_2} \leftarrow z_2^{(0)}$, and $t \leftarrow 0$
repeat
    Generate random variables i, j and r
    Perform the swap of numbers i and j in encoding
    Set $x^{(t+1)} \leftarrow$ newly created solution
    Calculate objective function value of $x^{(t+1)}$ for both objectives as $z_1^{(t+1)}$ and $z_2^{(t+1)}$
    if $x^{(t+1)} \succ \hat{x}$ or $r < e^{\frac{-\Delta z_1}{q}}$ or $r < e^{\frac{-\Delta z_2}{q}}$ **then**
        Accept the move
        Set $\hat{x} \leftarrow x^{(t+1)}$ and $\hat{z_1} \leftarrow z_1^{(t+1)}$ and $\hat{z_2} \leftarrow z_2^{(t+1)}$
    end if
    $t \leftarrow t + 1$
    $d_1 \leftarrow \dfrac{\sum_{f=t-10}^{t} z_1^f}{10}$
    $d_2 \leftarrow \dfrac{\sum_{f=t-10}^{t} z_2^f}{10}$
    if $t \geq 500$ and $0.99\hat{z_1} \leq d_1 \leq 1.01\hat{z_1}$ and $0.99\hat{z_2} \leq d_2 \leq 1.01\hat{z_2}$ **then**
        Stop algorithm
    else if Temperature change condition is satisfied **then**
        Set $q \leftarrow q \times 0.7$
    end if
until $t = t_{\max}$
output approximate Pareto set of incumbent solutions

7.2.2 Multi-objective 2-opt move algorithm

The MO2Opt algorithm formulated by the researcher to solve the bi-objective JSP presented in this project, is outlined in Algorithm 12. The algorithm starts off with the creation of the initial population which has a size of 50 solutions. The size of the population used by a metaheuristic plays a significant role in how well the metaheuristic performs. Small population sizes tend to result in faster convergence, but this increases the risk of converging to a local optimum. On the other hand, by adopting large population sizes, more solutions in the solution space can be explored, but this increases the time to convergence. The researcher, therefore, used a population size of 50, which ensures that the

Algorithm 12 Multi-objective 2-opt Move Algorithm (MO2Opt)

Generated initial population of 50 solutions with encoding algorithm
Set stopping criteria $d \leftarrow 0$, IterationCount $\leftarrow 1$, ParetoSet $\leftarrow \emptyset$ and HyperArea $\leftarrow \emptyset$
repeat
    Calculate the objective function values of each solution
    Perform a Pareto sort, to determine the Pareto set of the current population
    Set ParetoSet $\leftarrow$ the Pareto set after the Pareto sort
    Set $\text{HyperArea}_{\text{IterationCount}} \leftarrow$ hyper area of the current Pareto set
    if IterationCount > 10 **then**
        Set $d \leftarrow \dfrac{\sum_{f=\text{IterationCount}-10}^{\text{IterationCount}} \text{HyperArea}_f}{10}$
        if $0.975d \leq \text{HyperArea}_{\text{IterationCount}} \leq 1.025d$ **then**
            Stop algorithm
        end if
    end if
    for $k = 1$ to PopulationSize **do**
        Generate random variables i and j, where $j > i$
        Select a random solution from ParetoSet
        Perform 2-opt move by reversing numbers of encoding from i to j in the selected solution
    next k
    IterationCount $\leftarrow$ IterationCount $+ 1$
until IterationCount $= 1\,000$
output ParetoSet

solution space has been adequately explored while also not requiring excessive computational time to converge. Algorithm 7 is used to create the solutions. Next, the stopping criterion d and IterationCount variable, as well as the sets ParetoSet and HyperArea are created. IterationCount is then set to one and d set to zero. Thereafter, the objective function values are calculated for each solution. A Pareto sort is then performed to determine the Pareto set of the current population, after which ParetoSet is set to the Pareto set that was determined. The hyper area calculation

$$H(PF_{\text{known}}) = \left\{ \bigcup_i a(x_i) \mid \forall x_i \in PF_{\text{known}} \right\}, \tag{7.1}$$

where $a(x_i)$ is the rectangular area bounded by an origin and $f(x_i)$ and $PF_{\text{known}} =$ ParetoSet, is then used to calculate the hyper area value of the current Pareto set. When

more than 10 iterations have been completed, d is set to the average hyper area value over the previous 10 iterations. Next, a test will be performed to determine if the current hyper area value is within 0.025 of d. The interval

$$0.975d \leq \text{HyperArea}_{\text{IterationCount}} \leq 1.025d$$

was selected to ensure that the algorithm only stops once the current hyper area value has stagnated compared with the previous 10 iterations of the algorithm and no further improvement has been found. If this occurs, then the algorithm is stopped and ParetoSet is presented. Otherwise, a *for* loop is initiated which will loop from one to the population size. During each loop, two random variables (i and j) will be generated and a random solution that forms part of ParetoSet will be selected. The 2-opt move will then be performed on the selected solution by reversing the positions of numbers in the encoding from i to j. The random variables i and j will be different for each solution, which will enable the algorithm to search a larger area in the solution space. The IterationCount variable will then also be incremented by one. If the stopping criterion is never met, then the algorithm will run until it reaches 1 000 iterations, after which ParetoSet is presented. This concludes the formulation of the MO2Opt algorithm. As mentioned previously, the MO2Opt algorithm was formulated by the researcher, and the implementation of this algorithm to solve the bi-objective JSP presented in this project, will be the first of its kind. Next, the NSGAII will be discussed.

7.2.3 Non-dominated sorting genetic algorithm II

The NSGAII was proposed by Deb *et al.* (2002) to alleviate the three main criticisms of multi-objective evolutionary algorithms, *i.e.* computational complexity, non-elitism approach and the need for specifying a sharing parameter. There are many studies that were found in literature that used the NSGAII to solve a multi-objective JSP. Some examples of such studies include a study by Rabiee *et al.* (2012), where the authors solved a partial flexible JSP with the objective of minimising both the makespan and total operational cost. Jiang *et al.* (2014) also conducted a study on solving a multi-objective flexible JSP, in which the makespan, processing cost, energy consumption and cost-weighted processing quality are considered as objectives. In another study by Yuan and Xu (2015), the authors propose new memetic algorithms (MAs) for the multi-objective flexible JSP with the objectives to minimise the makespan, total workload, and critical workload. The MAs are developed by incorporating a novel local search algorithm into the NSGAII, which chooses some good individuals from the offspring population for local search using a selection mechanism. Comparison tests were then carried out with state-of-the-art methods for

the multi-objective JSP, and the results showed that the proposed MAs performed much better than all other methods considered in the study.

Before the NSGAII can be formulated for the stated bi-objective JSP, the crossover function must first be addressed. The crossover function that will be used in the algorithm is the same as in Falkenauer and Bouffouix (1991). This specific crossover function prevents the duplication or the omission of positions within a solution. To explain the crossover function, consider two solutions that need to be crossed, *i.e.*

123456789 and
769432158.

First, two crossing locations are chosen at random, say at positions 3 and 6, which we note

123|456|789 and
769|432|158.

Since the contents within the sections are not equal, the second solution will be modified to enable the crossover to occur. To do so, genes 4, 5 and 6 have to be brought from the first solution into the second solution, and genes 4, 3 and 2 from the second solution must be brought over to the first solution. The genes that have to be brought from the first solution are eliminated from the second solution, and *vice versa*. The structure of the second solution when the genes are eliminated is

7**H**9|**H**32|1**H**8,

where **H** represents a 'hole' that still needs to be filled. After the genes are eliminated from the second solution, the holes are moved to the crossing section. This step will yield

793|**HHH**|218.

The holes can now be filled with the genes from the crossing section of the first solution, giving

793|456|218.

These steps can then be repeated to alter the structure of the first solution, to ultimately generate another solution. When the crossover function is completed the new solutions are

156|432|789 and
769|456|218.

The NSGAII that has been adapted to solve the bi-objective JSP proposed in this research project, is outlined in Algorithm 13.

The algorithm starts off with the generation of an initial random population P_0 of N members according to the Algorithm 7. In this case, the population size N was again chosen to be 50, for the same reason as presented in the MO2Opt formulation. This population size will ensure that the solution space has been adequately explored while also not requiring excessive computational time to converge. Each of these members is then evaluated according to the stated objectives. The iteration counter t, HyperArea set and stopping criterion d are then created. P_0 is then sorted based on the non-domination, which assigns a rank to each member equal to its non-domination level (*i.e.* the rank 1 is given to the best level, rank 2 to the next best level, and so on). The crossover and mutation functions are then used to generate the offspring population Q_0. For the crossover function, the parent chromosomes are randomly selected from the members that have a non-domination level of one. If there are not enough members that have a non-domination level of one, then any other member of the current population will be selected. The mutation function is applied to a position in an encoding if the mutation probability is less than 0.03.

After Q_t has been generated, it is combined with P_t into R_t. Each of the members in R_t is then evaluated according to the stated objectives, after which a fast non-dominated sort is performed on R_t. A detailed description of the fast non-dominated sort is provided in Deb *et al.* (2002). The result of the fast non-dominated sort is then stored in $\mathcal{F}$. P_{t+1} is then created and i is set to one. Until the new population P_{t+1} is filled, crowding distance assignment must be performed on $\mathcal{F}_i$ and the ith non-dominated front must be included in the new population P_{t+1}. The variable i must also be incremented by one. The crowding distance assignment is also discussed in more detail in Deb *et al.* (2002).

After this loop, the current $\mathcal{F}_i$ must be sorted in descending order of crowding distance. The first $(N - |P_{t+1}|)$ elements of $\mathcal{F}_i$ must then be added to P_{t+1}. The hyper area value of the current $\mathcal{F}_1$ is then calculated using (7.1), where $a(x_i)$ is the rectangular area bounded by an origin and $f(x_i)$ and $PF_{\text{known}} = \mathcal{F}_1$. When more than 10 iterations have been completed, d is set to the average hyper area value over the previous 10 iterations. Next, a test will be performed to determine if the current hyper area value is within 0.025 of d. The interval

$$0.975d \leq \text{HyperArea}_t \leq 1.025d$$

was selected to ensure that the algorithm only stops once the current hyper area value has stagnated compared with the previous 10 iterations of the algorithm and no further improvement has been found. If this occurs, then the algorithm is stopped and P_{t+1} is presented. Otherwise, the crossover and mutation functions will be used to generate a new offspring population Q_{t+1}, after which t will be incremented by one. If the hyper area

Algorithm 13 Non-dominated Sorting Genetic Algorithm II (NSGAII)

Generate an initial random population P_0 of N members and evaluate each member according to the stated objectives. Set $t \leftarrow 0$, HyperArea $\leftarrow \emptyset$ and the stopping criteria $d \leftarrow 0$
Sort P_0 based on the non-domination
Use crossover and mutation to generate offspring population Q_0
repeat
    Combine parent and offspring populations $R_t \leftarrow P_t \cup Q_t$
    Evaluate each member of R_t according to the stated objectives
    Perform fast non-dominated sort on R_t and store in $\mathcal{F}$
    Store non-dominated fronts of R_t in $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$
    Set $P_{t+1} \leftarrow \emptyset$ and $i \leftarrow 1$
    repeat
        Perform crowding distance assignment on $\mathcal{F}_i$
        $P_{t+1} \leftarrow P_{t+1} \cup \mathcal{F}_i$
        $i \leftarrow i + 1$
    until $|P_{t+1}| + |\mathcal{F}_i| \leq N$
    Sort $\mathcal{F}_i$ in descending order of crowding distance
    Add the first $(N - |P_{t+1}|)$ elements of $\mathcal{F}_i$ to P_{t+1}
    Calculate HyperArea$_t$ using (7.1)
    if $t > 10$ **then**
        Set $d \leftarrow \dfrac{\sum_{i=t-10}^{t} \text{HyperArea}_i}{10}$
        if $0.975d \leq \text{HyperArea}_t \leq 1.025d$ **then**
            Stop algorithm
        end if
    end if
    Use crossover and mutation to generate new offspring population Q_{t+1}
    $t \leftarrow t + 1$
until $t = 1000$
output P_{t+1}

stopping criterion is never met, then the algorithm will run until it reaches 1 000 iterations, after which P_{t+1} is presented. This concludes the formulation of the NSGAII. Next the MOGA will be discussed.

7.2.4 Multi-objective genetic algorithm

The genetic algorithm has been discussed in detail in Section 2.3.2; however, a brief summary will be provided. The term *genetic algorithm* was first used by Holland (1975), and it is motivated by the theory of evolution, with the survival of the fittest being a fundamental property. Genetic algorithms have been designed as general search strategies and optimisation methods. Bekker (2016b) describes the analogy when applying the GA to an optimisation problem as: a *population* of possible solutions is created from the *solution space* of the problem by varying the values of the *decision variables*. Each individual/*chromosome* in the population is an *encoding* of a solution. Some of the individuals (referred to as *parents*) combine with each other to form new individuals or *offspring*. The combination phase is called the *crossover* phase, and good individuals are more likely to be selected for the phase. This phase is also subjected to random variation (referred to as *mutation*) where some part of the chromosome is randomly changed. The new generation of chromosomes replaces some or all of the previous generation, thus causing the population of solutions to improve over time. The MOGA is the expansion of the basic genetic algorithm to the multi-objective domain.

Examples of where the MOGA has been used to solve the multi-objective JSP, include a study by Ponnambalam *et al.* (2001), in which the authors proposed a MOGA to derive the optimal machine-wise priority dispatching rules to resolve the conflict among the contending jobs in the Giffler and Thompson procedure applied for job shop problems. The objectives that were considered include the minimisation of the makespan, total idle time of machines and the total tardiness. The proposed methodology was applied to 28 benchmark problems to illustrate its applicability and usefulness. Rabiee *et al.* (2012) also solved a partial flexible JSP with the objective of minimising both the makespan and total operational cost by incorporating the MOGA. In another study by Sridhar and Rajendran (1996), the authors applied a MOGA to the problem of scheduling in a flow shop and flowline-based cellular manufacturing system with the objectives of minimising the makespan, total flow time and machine idle time. The proposed algorithm performed well when scheduling in a flowline-based cellular manufacturing system.

The MOGA that has been adapted to solve the bi-objective JSP proposed in this research project, is outlined in Algorithm 14.

The algorithm starts off with the generation of an initial random population P_0 of N members according to Algorithm 7. The population size N was again chosen as 50, for the same reason as presented in the MO2Opt and NSGAII. Each of these members is then evaluated according to the stated objectives. The iteration counter t, HyperArea set, stopping criterion d and mutation probability $MutationProb$ are then created. A Pareto sort is then performed on P_0, which assigns a rank to each member equal to the number of members that dominates it. A test is then performed to determine if more than 250 replications have been completed. If that is the case, then d is set to the average hyper area value over the previous 10 iterations, after which a test will be performed to determine if the current hyper area value is within 0.025 of d. The interval

$$0.975d \leq \text{HyperArea}_t \leq 1.025d$$

was selected to ensure that the algorithm only stops once the current hyper area value has stagnated compared with the previous 10 iterations of the algorithm and no further improvement has been found. If both the hyper area and replication conditions have been met, the algorithm stops; otherwise, two individuals are selected from the current Pareto set for crossover. The crossover function performed is the same as for Algorithm 13. Next, the mutation function is performed, which states that if for each position in the offspring the $MutationProb$ is less than 0.03 then two randomly selected positions in the offspring will be swapped. After the mutation function, both offspring are evaluated according to the stated objectives and the member with the worst Pareto rank in P_t is replaced with the better of the two offspring. The variable t is then incremented by one. If the stopping criteria are never met, then the algorithm will run until it reaches 1 000 iterations, after which P_t is presented. This concludes the formulation of the MOGA. Next the MOOCEM will be discussed.

Algorithm 14 Multi-objective Genetic Algorithm (MOGA)

Generate an initial random population P_0 of N members and evaluate each member according to the stated objectives. Set $t \leftarrow 0$, HyperArea $\leftarrow \emptyset$, stopping criteria $d \leftarrow 0$ and the mutation probability $MutationProb \leftarrow 0$.
repeat
    Perform a Pareto sort on P_t
    Set $\text{HyperArea}_t \leftarrow$ hyper area of current Pareto set
    if $t > 250$ **then**
        Set $d \leftarrow \dfrac{\sum_{i=t-10}^{t} \text{HyperArea}_i}{10}$
        if $0.975d \leq \text{HyperArea}_t \leq 1.025d$ **then**
            Stop algorithm
        end if
    end if
    Randomly select two individuals to reproduce from the current Pareto set
    Do crossover between the members
    for $k = 1$ to offspring length **do**
        Set $MutationProb \leftarrow$ random value
        if $MutationProb \leq 0.03$ **then**
            Generate random variables i and j, where $j > i$
            Swap gene i with gene j in both offspring
        end if
    next k
    Evaluate offspring according to the stated objectives
    Replace the member with the worst Pareto rank in P_t, with the better of the two offspring
    $t \leftarrow t + 1$
until $t = 1000$
output P_t

7.2.5 Multi-objective optimisation cross entropy method

The *cross entropy method* (CEM) is the single-objective counterpart of the MOOCEM and was first introduced by Rubinstein and Kroese (2004) as a simple method that generally converges quickly in single-objective problems. However, Bekker and Aldrich (2011) were the first to extend the CEM to problems with more than one objective, and apply the method to computationally expensive time-dependent, stochastic simulation problems

where the optimisation of two or more objectives is pursued.

The CEM has previously been used by Santosa *et al.* (2011) in a hybrid algorithm with a genetic algorithm to solve a single-objective no-wait job-shop scheduling problem where the makespan is considered as an objective. This hybrid algorithm was compared with other metaheuristics which include a genetic algorithm-simulated annealing hybrid and a hybrid tabu search. The proposed hybrid CEM performed better than the other two metaheuristics. Another CEM-genetic algorithm hybrid has also been used by Nurkhalida and Santosa (2012) to solve a multi-objective JSP where the weighted objective approach was used to accommodate the multi-objective calculation. No further literature was found where the CEM was used to solve the JSP; moreover, MOOCEM has not previously been used to solve the multi-objective JSP. This application of the MOOCEM in the multi-objective simulation scheduler is therefore the first of its kind. The MOOCEM that has been adapted to solve the bi-objective JSP proposed in this research project, is outlined in Algorithm 15.

The algorithm starts off by creating the probability matrix P, hyper area set HyperArea, stopping criterion d and choosing a population size N. The population size N was again chosen as 50, for the same reason as presented in the MO2Opt and NSGAII. Next the algorithm creates a population of sequences by using Algorithm 16 and P_t. Each member in the population is then evaluated according to the stated objectives, after which a Pareto sort is performed to rank the members. The top 20 percent of the solutions are selected as part of the elite set ES, which will be used to update P_t. The researcher chose to select the top 20 percent of the population, because only the best solutions should be used to update the probability matrix P_t. The mixture of solutions in the top 20 percent, which will include the non-dominated members and possibly some members that are dominated, will help the algorithm to explore a larger area in the solution space. P_t is then updated by using

$$p_{ij} = \frac{\sum_{b=1}^{N} I_{\{b \in ES\}} I_{\{x_{ij}=1\}}}{\sum_{b=1}^{N} I_{\{b \in ES\}}}. \tag{7.2}$$

After P_t is updated the smoothing function

$$P_t = P_t(\alpha - 1) + P_{t-1}\alpha,$$

where $\alpha = 0.3$, is applied. This smoothing function will prevent the probability matrix from changing too rapidly, while still capturing important patterns in the members of the ES. HyperArea is then updated with the hyper area value of the current elite set. Next, a test is performed to determine if more than 10 iterations have been completed. If that is

the case, then d is set to the average hyper area value over the previous 10 iterations, after which a test will be performed to determine if the current hyper area value is within 0.025 of d. The interval

$$0.975d \leq \text{HyperArea}_t \leq 1.025d$$

was selected to ensure that the algorithm only stops once the current hyper area value has stagnated compared with the previous 10 iterations of the algorithm and no further improvement has been found. If these conditions are met, the algorithm stops; otherwise, t will be incremented by one and the algorithm will repeat. If the hyper area stopping criterion is never met, then the algorithm will run until it reaches 1 000 iterations, after which ES is presented. This concludes the description of the MOOCEM.

Algorithm 15 Multi-objective Optimisation Cross-Entropy Method (MOOCEM)

Let P be the transition matrix of probabilities, d the stopping criteria, N the population size and HyperArea the hyper area set
Set $P \leftarrow \emptyset$, HyperArea $\leftarrow \emptyset$, $d \leftarrow 0$, $t \leftarrow 0$ and $N \leftarrow 50$
Initialise P_t
repeat
    for $k = 1$ to N **do**
        Construct a sequence using Algorithm 16 and P_t
    next k
    Evaluate all members of the population according to the stated objectives
    Perform a Pareto Sort and rank each member
    Select elite set ES
    Update P_t using (7.2)
    Smooth $P_t \leftarrow \alpha P_t + (1 - \alpha)P_{t-1}$
    Set $\text{HyperArea}_t \leftarrow$ hyper area of current ES
    if $t > 10$ **then**
        Set $d \leftarrow \dfrac{\sum_{i=t-10}^{t} \text{HyperArea}_i}{10}$
        if $0.975d \leq \text{HyperArea}_t \leq 1.025d$ **then**
            Stop algorithm
        end if
    end if
    $t \leftarrow t + 1$
until $t = 1\,000$
output ES

Algorithm 16 Generation of sequences of the JSP

1: Let M_0 be the number of operations that need to be scheduled
2: **for** $k = 1$ to M_0 **do**
3: Sample an operation from the probability matrix P
4: Change the probability of selecting the same operation to 0 for all rows in P
5: Calculate the new weighted row of the P
6: **next** k

An example of how Algorithm 16 is applied, will now be discussed. Table 7.1 illustrates the probability matrix for the example, given that the first operation in the sequence is operation '1'. As seen in the table, the probability of selecting '1' has already been changed to 0 for all the rows, because it already forms part of the generated sequence.

Table 7.1: Probability matrix after operation 1 was selected

	1	2	3	4
1	0	0.333	0.333	0.333
2	0	0.333	0.333	0.333
3	0	0.333	0.333	0.333
4	0	0.333	0.333	0.333

The next operation in the sequence is then sampled from the row of the current operation, which is '1' in this case, by generating a random number in Step 3. The next operation that is selected will be the operation where the cumulative selection probability is greater than the random number. Assume the random number is 0.55, then the next operation that is selected is operation '3', as its cumulative selection probability is 0.666 (the probability of selecting operation '2' plus the probability of selecting operation '3'). Operation '3' is then added to the new sequence. In Step 4, the selection probability of the new operation is changed to 0 for all rows in the probability matrix, as illustrated in Table 7.2.

Table 7.2: Updated selection probability for operation 3 in all rows

	1	2	3	4
1	0	0.333	0	0.333
2	0	0.333	0	0.333
3	0	0.333	0	0.333
4	0	0.333	0	0.333

Next, in Step 5, the row of the new operation has to be normalised by calculating the new weighted row. When this step is completed, the selection probability of operations '2' and '4' in the third row will both be 0.5, as shown in Table 7.3. The next operation will then be sampled from the third row.

Table 7.3: New weighted row calculated for operation 3

	1	2	3	4
1	0	0.333	0	0.333
2	0	0.333	0	0.333
3	0	0.5	0	0.5
4	0	0.333	0	0.333

Step 3 is then repeated and a new random number is generated, *e.g.* 0.26. The operation that is selected is operation '2', as its cumulative selection probability of 0.5 is greater than the random number. The selection probability of operation '2' is then changed to 0 for all rows in the probability matrix, in Step 4. Lastly, the row of operation '2' is then normalised again, as illustrated in Table 7.4. From the updated probability matrix, the selection probability of operation '4' is now 1, which means that on the next iteration of the algorithm, it will be selected as the next operation. The final sequence that was generated for this example is 1–3–2–4. This concludes the example of generating a sequence using Algorithm 16.

Table 7.4: New weighted row calculated for operation 2

	1	2	3	4
1	0	0	0	0.333
2	0	0	0	1
3	0	0	0	0.5
4	0	0	0	0.333

It should be noted that each metaheuristic has the same stopping criterion, where the average hyper area value of the Pareto set in the previous 10 iterations is compared with the hyper area value of the current Pareto set. If the current hyper area value is within a specified range, then the algorithm will stop. This stopping criterion was tested in each algorithm only when roughly 500 solutions had already been evaluated, so that each algorithm had the same opportunity to find good solutions before it stopped.

7.3 Bi-objective experiments and results

In this section, comparison tests will be constructed and conducted on all the metaheuristics that were discussed in the previous section. Three test scenarios will again be created for the comparison tests. Weber *et al.* (2019) stated that the standardisation and systematisation of test data for the JSP is still lacking. This is also true for the stochastic JSP, and therefore, randomly generated test data for the three test scenarios containing 50, 100 and 200 jobs respectively, were used. Careful attention was given when the data was randomly generated, to ensure that the data would not favour a single method. The processing times were sampled from a log-normal distribution to introduce stochastic processing times. The log-normal distribution was chosen in this study to ensure that the processing times that were sampled, are all positive, as negative processing times are not allowed. The three test scenarios were defined as:

- **Test scenario 9:** 50 jobs were entered into the system, each with different processing times that were sampled from a log-normal distribution with $\mu =$ estimated processing time and $\sigma = 0.2$(estimated processing time).
- **Test scenario 10:** 100 jobs were entered into the system, each with different processing times that were sampled from a log-normal distribution with $\mu =$ estimated processing time and $\sigma = 0.2$(estimated processing time).
- **Test scenario 11:** 200 jobs were entered into the system, each with different processing times that were sampled from a log-normal distribution with $\mu =$ estimated processing time and $\sigma = 0.2$(estimated processing time).

Each of the chosen metaheuristics was then applied to the stated test scenarios, by conducting 500 experiments per metaheuristic for each scenario. To determine the best-performing metaheuristic, performance metrics needed to be selected in order to compare the generated approximate Pareto sets. The metrics selected for the comparison tests are the *generational distance* (GD), *hyper area ratio* (HR) and *maximum spread* (MS), as presented in Yen and He (2014). Each of the performance metrics required a true Pareto set; however, due to new test scenarios being created with stochastic processing times, there was no true Pareto set available. To overcome this, the researcher decided to perform a Pareto sort on the results of all 2 500 experiments per test scenario. The outcome of this Pareto sort identified a Pareto set with the best solutions from the 2 500 experiments per test scenario. These Pareto sets could therefore contain solutions that were generated with different metaheuristics and can be used as the approximate true Pareto sets in the performance metric calculations. Figure 7.3 will be used to describe this process. Two

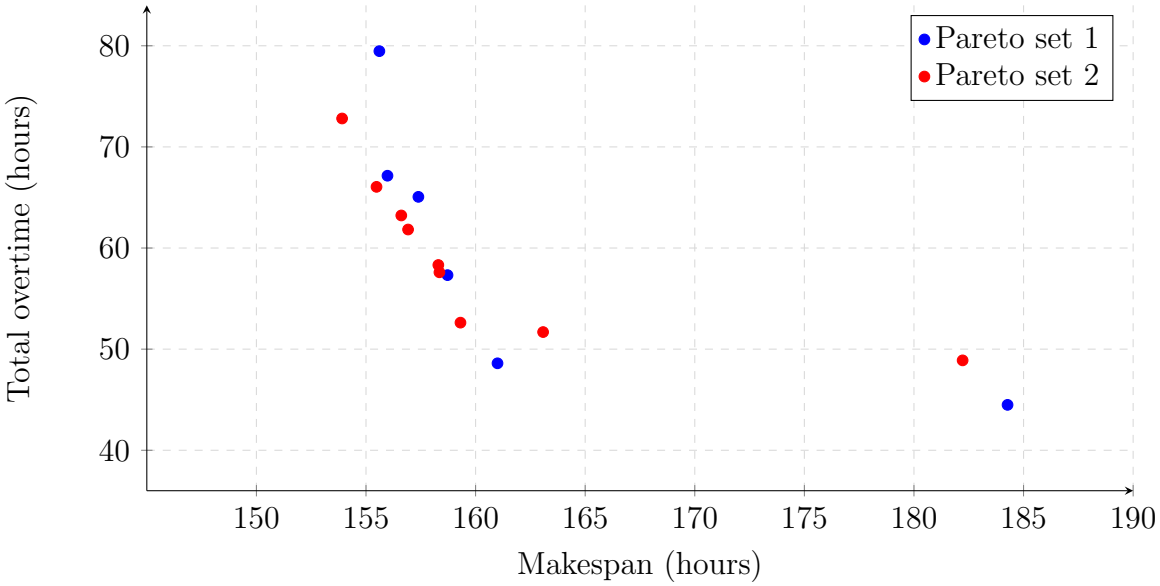


Figure 7.3: Two example Pareto sets on which a Pareto sort is performed

Pareto sets, each generated by a different metaheuristic, are included and shown in Figure 7.3. Both these Pareto sets are used in the new Pareto sort, which identifies the best solutions from both these Pareto sets and stores them as the approximate true Pareto set. The result of the Pareto sort is illustrated in Figure 7.4. The approximate true Pareto set that is displayed in Figure 7.4 contains solutions from both of the example Pareto sets, and this new set can now be used as the true Pareto set in the performance metric calculations.

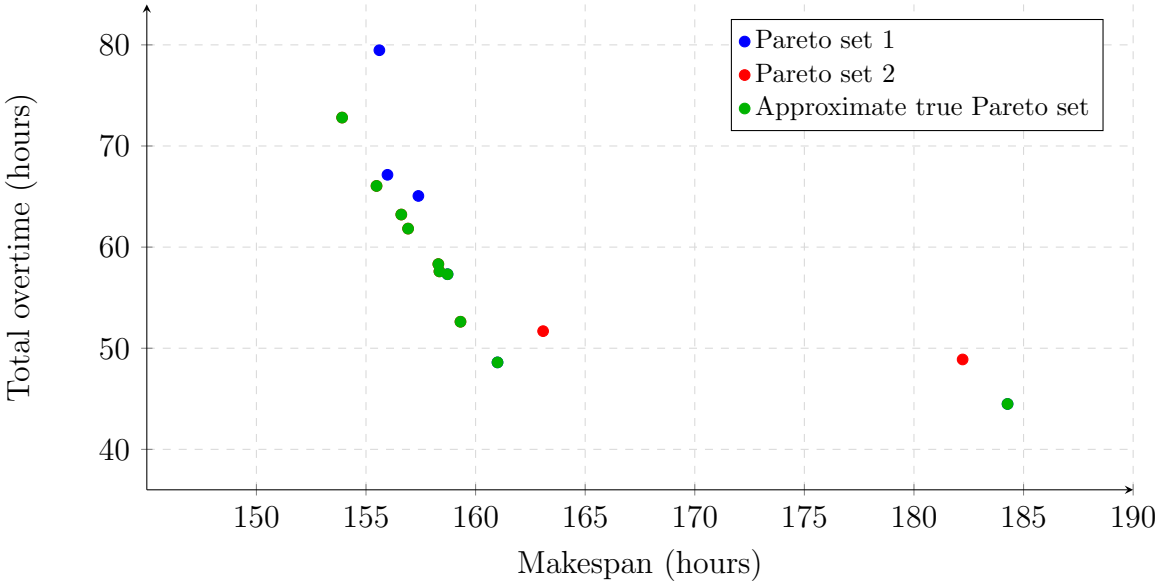


Figure 7.4: Approximate true Pareto set

In this example only two Pareto sets were used, so one can argue that there may be considerably better solutions that were not included and that could have formed part of the approximate true Pareto set. This observation is true; however, for the comparison tests, 2 500 experiments were conducted per test scenario which generated 2 500 different Pareto sets per test scenario. These 2 500 Pareto sets per test scenario were subjected to the Pareto sort, which generated an approximate true Pareto set for each test scenario. With the sheer number of Pareto sets that were used as part of the Pareto sort, one can assume that the approximate true Pareto set is close to the true Pareto set. Therefore, the approximate true Pareto set was used as the true Pareto set in the performance metric calculations.

A discussion of each performance metric will now follow.

1. **Generational distance (GD)**

   The GD performance metric is calculated using

$$GD = \frac{\sqrt{\sum_{i=1}^{u} d_i^2}}{u},$$

   where $d_i = \min_j ||f(x_i) - PF_{\text{true}}(x_j)||$ refers to the distance in objective space between individual x_i and the nearest member in the true Pareto set, PF_{true}, and u is the number of individuals in the approximation set. The lowest value of the GD represents the best performance. An example of the GD is illustrated in Figure 7.5, where the distance from each member of the Pareto set to the nearest member in the true Pareto set was identified. The average of these distances is then taken to calculate the GD.

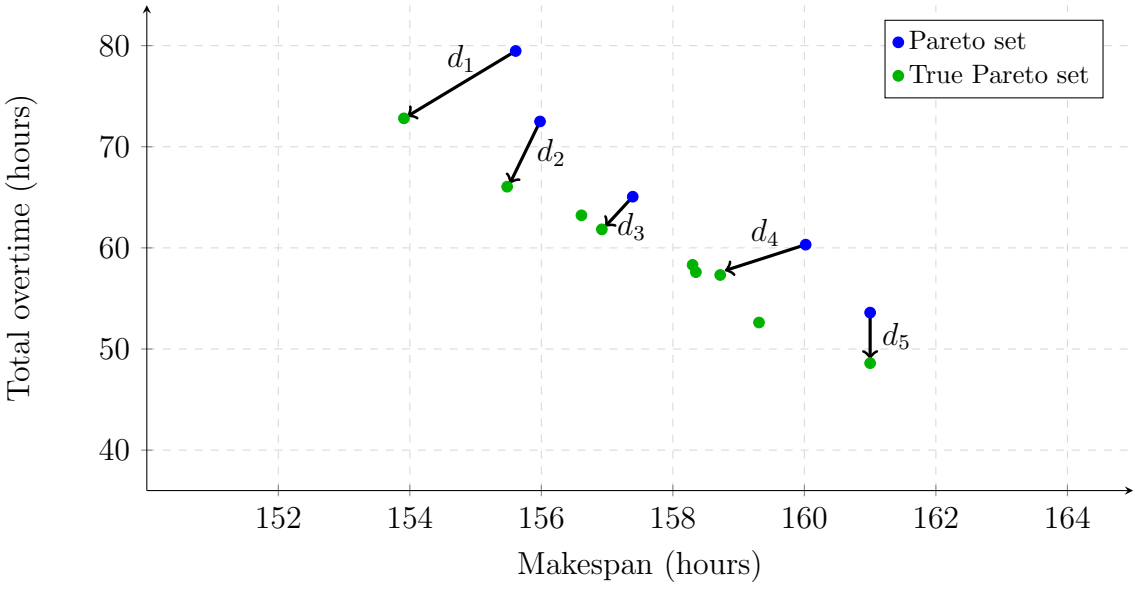


Figure 7.5: Example of the GD performance metric

2. Hyper area ratio (HR)

Before the HR performance metric can be calculated, one must first calculate the hyper area for the approximation set PF_{known} and the true Pareto set PF_{true}. For each member x_i in PF_{known}, there is a rectangular area, $a(x_i)$, bounded by an origin and $f(x_i)$. The union of all these rectangular areas is referred to as the hyper area of PF_{known} and is calculated as

$$H(PF_{\text{known}}) = \left\{ \bigcup_i a(x_i) \mid \forall x_i \in PF_{\text{known}} \right\}.$$

The Hyper area for PF_{true} can be determined in the same way. The coordinate $(0, 0)$ was selected as the origin by the researcher. When the hyper area value of both PF_{known} and PF_{true} are known, the HR performance metric can be calculated as

$$HR = \frac{H(PF_{\text{known}})}{H(PF_{\text{true}})}.$$

The approximation set that yielded the HR value that is the closest to one, represents the best performing set. An example of the HR is illustrated in Figure 7.6, where the hyper area of three Pareto sets are displayed. For this example the best performing Pareto set is Pareto set 1, because its HR value is closer to one than the HR value of Pareto set 2.

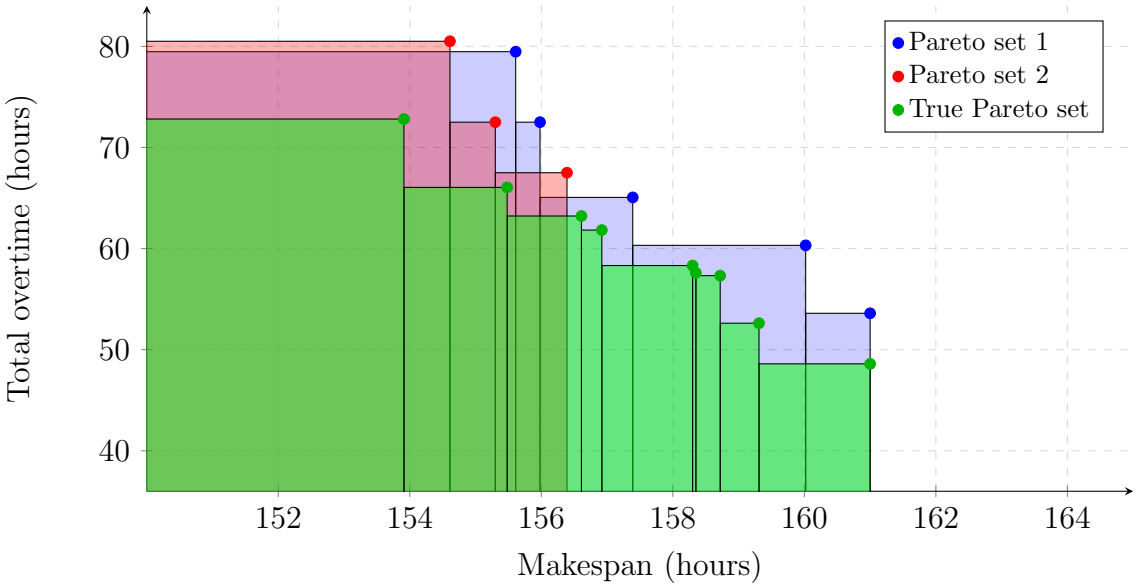


Figure 7.6: Example of the HR performance metric

3. Maximum spread (MS)

This metric is calculated to measure how well the true Pareto set PF_{true} is covered by the approximation set PF_{known}. The metric is calculated as

$$MS = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left[\frac{\min(PF^{\max}_{\text{known,i}}, PF^{\max}_{\text{true,i}}) - \max(PF^{\min}_{\text{known,i}}, PF^{\min}_{\text{true,i}})}{PF^{\max}_{\text{true,i}} - PF^{\min}_{\text{true,i}}}\right]^2},$$

where $PF^{\max}_{\text{known,i}}$ and $PF^{\min}_{\text{known,i}}$ are the maximum and minimum of the ith objective in PF_{known}, respectively; and $PF^{\max}_{\text{true,i}}$ and $PF^{\min}_{\text{true,i}}$ are the maximum and minimum of the ith objective in PF_{true}, respectively. H denotes the number of objectives that are considered. A higher value of MS reflects that a larger portion of PF_{true} is covered by PF_{known}. Therefore, the approximation set with the largest MS value represents the best performing set. An example of the MS is illustrated in Figure 7.7, where three Pareto sets are displayed. For this example the best performing Pareto set is Pareto set 1, because its MS value is larger than that of Pareto set 2, which means that it covers a larger portion of PF_{true}.

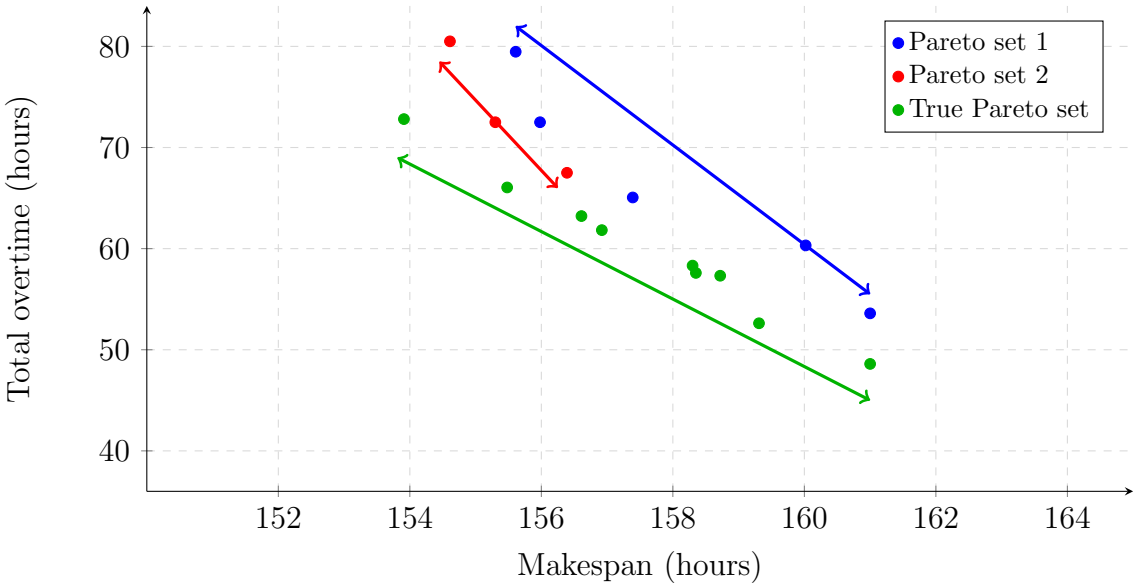


Figure 7.7: Example of the MS performance metric

Now that the performance metrics have been defined, it is important to understand why three performance metrics are used, instead of just one. The GD performance metric focuses solely on the closeness of the solutions to the true Pareto set, but disregards the spread of the solutions. The MS performance metric considers only the spread of solutions, but disregards the closeness of the solutions to the true Pareto set. The HR performance

metric considers both the closeness and diversity of the solutions. Each of these metrics has its own shortcoming, which is why the researcher chose all three. The use of all three metrics ensures that a thorough analysis can be conducted to find the best performing metaheuristic for each test scenario. When there is no metaheuristic that outperformed the others in all three metrics, then the metaheuristic that performed best in most of the metrics will be selected as the best. For example, if the MOGA performed best in both the GD and HR metrics while the MOOCEM performed best in the MS metric, the MOGA will be selected as the best performing metaheuristic.

These metrics could now be applied to each of the experiments that were conducted, after which the results can be compared with paired t-tests. The paired t-test was selected for comparison tests, because the experiments are independent and the researcher assumed normality, as well as unknown and unequal variances for the results. The mean performance metric value for each metaheuristic and test scenario, is provided in Table 7.5. From the paired t-tests that were conducted, a p-value table could be constructed for each of the performance metrics. These p-value tables will now be discussed for each test scenario.

Table 7.5: Mean performance metric values for each metaheuristic and the different test scenarios (number of replication where 500 per metaheuristic)

	50 jobs			100 jobs			200 jobs		
Metaheuristic	GD	HR	MS	GD	HR	MS	GD	HR	MS
MOGA	7.073	0.904	0.492	107.259	1.138	0.404	145.965	1.144	0.333
NSGAII	3.717	0.877	0.588	65.020	1.091	0.535	89.886	1.107	0.481
MOOCEM	9.250	0.888	0.428	132.171	1.134	0.350	202.574	1.160	0.288
MOSA	5.577	0.906	0.555	88.725	1.121	0.467	109.896	1.164	0.447
MO2Opt	3.913	0.800	0.392	40.939	0.974	0.450	57.448	1.020	0.458

7.3.1 Test scenario 9 paired t-test results

The p-values generated for each performance metric are shown in Tables 7.6, 7.7 and 7.8. From Table 7.6, it is apparent that the results of all the metaheuristics are shown to be statistically different as all the p-values are less than 0.05. This means that the metaheuristic that achieved the lowest mean value for the GD performance metric, *i.e.* the NSGAII with a value of 3.717 in Table 7.5, performed best for the GD performance metric.

From Table 7.7, the NSGAII and MOOCEM results are not statistically different for the HR value, with a p-value of 0.060. The MOSA result is not statistically different from the MOGA result, with a p-value of 0.744. The other p-values are all less than 0.05

and therefore, the results of the metaheuristics are statistically different. The metaheuristic that performed best for the HR performance metric, is the metaheuristic whose HR value is the closest to one. Therefore, the MOSA algorithm performed best in the HR performance metric, with a HR value of 0.906. The MOGA HR value of 0.904 is not statistically different from the MOSA HR value, so either metaheuristic can be chosen.

Lastly, from Table 7.8, the results of all the metaheuristics are shown to be statistically different as all the p-values are less than 0.05. The metaheuristic that has the highest mean value for the MS performance metric, *i.e.* the NSGAII with a value of 0.588, performed best in the MS performance metric. Considering all these results, the NSGAII was identified as the best-performing metaheuristic for Test scenario 9. This observation could be made due to the NSGAII performing best in two of the three performance metrics, *i.e.* the GD and MS metrics.

Table 7.6: GD p-value table for Test scenario 9

	NSGAII	MOOCEM	MOSA	MO2Opt
MOGA	0.000	0.000	0.000	0.000
NSGAII		0.000	0.000	0.016
MOOCEM			0.000	0.000
MOSA				0.000

Table 7.7: HR p-value table for Test scenario 9

	NSGAII	MOOCEM	MOSA	MO2Opt
MOGA	0.000	0.015	0.744	0.000
NSGAII		0.060	0.000	0.000
MOOCEM			0.006	0.000
MOSA				0.000

Table 7.8: MS p-value table for Test scenario 9

	NSGAII	MOOCEM	MOSA	MO2Opt
MOGA	0.000	0.000	0.000	0.000
NSGAII		0.000	0.000	0.000
MOOCEM			0.000	0.000
MOSA				0.000

7.3.2 Test scenario 10 paired *t*-test results

The *p*-values generated for each performance metric are shown in Tables 7.9, 7.10 and 7.11. From Table 7.9, it is apparent that the results of all the metaheuristics are statistically different as all the *p*-values are less than 0.05. This means that the metaheuristic that achieved the lowest mean value for the GD performance metric, *i.e.* the MO2Opt algorithm with a value of 40.939 in Table 7.5, performed best.

From Table 7.10, the MOGA and MOOCEM results are not statistically different for the HR value, with a *p*-value of 0.496. The other *p*-values are all less than 0.05 and therefore, the results of the metaheuristics are statistically different. The metaheuristic that performed best for the HR performance metric, is the metaheuristic whose HR value is the closest to one. Therefore, the MO2Opt algorithm performed best with a mean value of 0.974.

Lastly, from Table 7.11, the results of all the metaheuristics are shown to be statistically different as all the *p*-values are less than 0.05. The metaheuristic that has the highest mean value for the MS performance metric, *i.e.* the NSGAII with a value of 0.535, performed best. Considering all these results, the MO2Opt algorithm was identified as the best-performing metaheuristic for Test scenario 10. This observation could be made due to the MO2Opt algorithm performing best in two of the three performance metrics, *i.e.* the GD and HR metrics.

Table 7.9: GD *p*-value table for Test scenario 10

	NSGAII	MOOCEM	MOSA	MO2Opt
MOGA	0.000	0.000	0.000	0.000
NSGAII		0.000	0.000	0.000
MOOCEM			0.000	0.000
MOSA				0.000

Table 7.10: HR *p*-value table for Test scenario 10

	NSGAII	MOOCEM	MOSA	MO2Opt
MOGA	0.000	0.469	0.004	0.000
NSGAII		0.000	0.000	0.000
MOOCEM			0.035	0.000
MOSA				0.000

Table 7.11: MS p-value table for Test scenario 10

	NSGAII	MOOCEM	MOSA	MO2Opt
MOGA	0.000	0.000	0.000	0.000
NSGAII		0.000	0.000	0.000
MOOCEM			0.000	0.000
MOSA				0.018

7.3.3 Test scenario 11 paired t-test results

The p-values generated for each performance metric are shown in Tables 7.12, 7.13 and 7.14. From Table 7.12, it can be seen that the results of all the metaheuristics are statistically different as all the p-values are less than 0.05. This means that the metaheuristic that achieved the lowest mean value for the GD performance metric, *i.e.* the MO2Opt algorithm with a value of 57.448 in Table 7.5, performed best.

From Table 7.13, the MOSA and MOOCEM results are not statistically different for the HR value, with a p-value of 0.493. The other p-values are all less than 0.05 and therefore, the results of the metaheuristics are statistically different. The metaheuristic that performed best for the HR performance metric, is the metaheuristic whose HR value is the closest to one. Therefore, the MO2Opt algorithm performed best with a mean value of 1.020.

Lastly, from Table 7.14, the results of the MO2Opt algorithm and the MOSA algorithm are not statistically different, with a p-value of 0.097. The results of all the other metaheuristics are shown to be statistically different as all the p-values are less than 0.05. The metaheuristic that has the highest mean value for the MS performance metric, *i.e.* the NSGAII with a value of 0.481, performed best. Considering all these results, the MO2Opt algorithm was identified as the best-performing metaheuristic for Test scenario 11. This observation could be made due to the MO2Opt algorithm performing best in two of the three performance metrics, *i.e.* the GD and HR metrics.

Table 7.12: GD p-value table for Test scenario 11

	NSGAII	MOOCEM	MOSA	MO2Opt
MOGA	0.000	0.000	0.000	0.000
NSGAII		0.000	0.000	0.000
MOOCEM			0.000	0.000
MOSA				0.000

Table 7.13: HR p-value table for Test scenario 11

	NSGAII	MOOCEM	MOSA	MO2Opt
MOGA	0.000	0.001	0.000	0.000
NSGAII		0.000	0.000	0.000
MOOCEM			0.493	0.000
MOSA				0.000

Table 7.14: MS p-value table for Test scenario 11

	NSGAII	MOOCEM	MOSA	MO2Opt
MOGA	0.000	0.000	0.000	0.000
NSGAII		0.000	0.000	0.001
MOOCEM			0.000	0.000
MOSA				0.097

Now that the best-performing metaheuristic has been identified for each test scenario, the next step is to perform a ranking and selection procedure on the results of these metaheuristics to determine which solutions should form part of the final Pareto set. To accomplish this, a newly developed ranking and selection procedure for stochastic systems, called Procedure *MMY*, will be implemented to determine the relaxed Pareto set for each test scenario. A detailed discussion and the implementation of Procedure MMY will now follow.

7.4 MMY integration and results

The members of the final Pareto set may initially not be statistically different, so they must finally be separated on a statistical basis. An example of why the members need to be separated on a statistical basis, is shown in Figure 7.8. The distribution for three samples are shown conceptually. Sample I and II are not statistically different due to the small variation in their sample means. This variation is known as *in-sample variation* which suggests that the two samples come from the same system. On the other hand, Sample III is statistically different from both Sample I and II due to the large variation in sample means. This large variation is known as *cross-sample variation*, which suggests that Sample III was selected from a different system from that of Sample I and II.

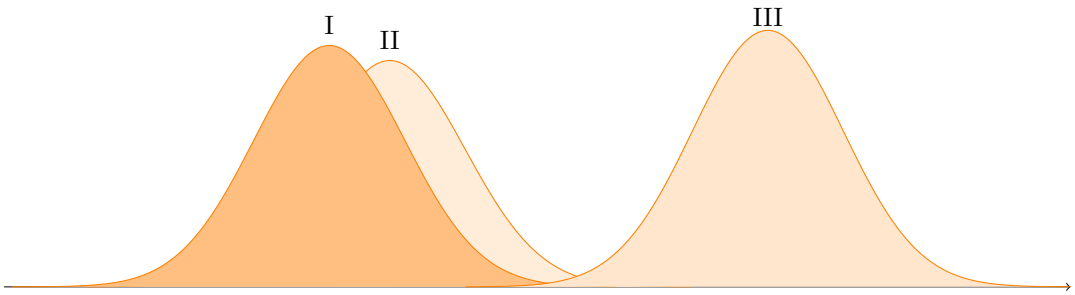


Figure 7.8: In-sample and cross-sample variation example

The in-sample and cross-sample variation between members must now be identified, which will enable one to separate the members on a statistical basis. To accomplish this, a *ranking and selection* (R&S) procedure must be implemented. The aim of R&S procedures is to select the best solution among a number of solutions of which the values for performance measures are estimated through simulation. There are two important approaches in R&S procedures, *i.e.* the *indifference-zone* (IZ) approach and the *optimal computing budget allocation* (OCBA) framework (Lee *et al.*, 2010). Procedures with the IZ approach identify the minimum number of simulation replications that is required for each solution to guarantee the probability of selecting the correct solution, $P(\text{CS})$, is greater than a prespecified value P^*. On the other hand, procedures in the OCBA framework find the optimal number of simulation replications for each solution to maximise $P(\text{CS})$ when the total number of simulation replications is limited to a certain value, referred to as the budget. An upper bound for $P(\text{CS})$ is found.

In multi-objective optimisation problems with two or more conflicting objectives, a single best solution does not exist. A set of non-dominated solutions is accepted as the desired output of the *multi-objective ranking and selection* (MORS) procedures. Most of the R&S research has focused on single-objective problems, and as a result, MORS procedures have only fairly recently appeared in literature. The *multi-objective optimal computing budget allocation* (MOCBA) procedure (Lee *et al.*, 2010), was the first MORS procedure that found a Pareto set. New MORS procedures have also only recently been proposed, such as Branke and Zhang (2015), Feldman *et al.* (2015) and Hunter and Feldman (2015), but none of these are based on the IZ approach. This makes the novel MORS procedures that were proposed by Yoon (2018), so much more valuable.

The novel procedures, which are *MMY*, *MMY1* and *MMY2*, are all based on the IZ approach, that guarantees the $P(\text{CS})$ requirement. The MMY procedure was designed to find a relaxed Pareto set, while MMY1 and MMY2 were devised to seek the true Pareto set without the indifference-zone (IZ) concept and the true Pareto set with the IZ concept,

respectively. For this study, the MMY procedure was selected for generating the relaxed Pareto set. This procedure was selected because both the MMY1 and MMY2 procedures are very strict regarding the inclusion of solutions into the Pareto set, while the MMY procedure would rather include the solutions where indifference is present. The MMY procedure was also selected ahead of the MOCBA, because the purpose for implementing a MORS procedure in this research project, was to find the minimum number of simulation replications that is needed for each solution, while also guaranteeing $P(\text{CS}) > P^*$ to ensure that the correct solutions are selected from the sets.

Before discussing the MMY procedure, the concept of *relaxed Pareto optimality*, as defined by Yoon (2018), must first be addressed. Relaxed Pareto optimality is defined as follows:

- Consider a true Pareto solution set Q based on true means without the IZ concept, and a true Pareto solution set Q_{IZ} based on true means with the IZ concept.
- Construct the union of these two Pareto sets: $Q_U = Q \cup Q_{IZ}$.
- Consider a subset of Q_{IZ}, called Q_S, which contains all solutions that have indifferent solutions, *i.e.*

$$Q_S = \{i \in Q_{IZ} \mid \exists j \in Q_{IZ} \text{ such that } i \simeq_{IZ} j\}.$$

- Divide Q_S into z sets, each consisting of solutions indifferent to each other, *i.e.*

$$Q_{S,p} = \{i, j \in Q_S \mid i \simeq_{IZ} j\} \ (p = 1, \ldots, z).$$

  It must be noted that $Q_{S,p}$ $(p = 1, \ldots, z)$ are z subsets of Q_S whose union is Q_S.
- A set of Pareto optimal solutions is called a *relaxed Pareto set*, denoted by Q_R , if it is a subset of Q_U, containing all solutions in $Q_{IZ} - Q_s$ and at least one solution from each $Q_{S,p}$ $(p = 1, \ldots, z)$.

It should be noted that when the description of relaxed Pareto optimality is applied, multiple relaxed Pareto sets can be present. Therefore, Yoon (2018) defined the set of all possible relaxed Pareto sets $\mathbb{Q}_{\mathbb{T}}$ as

$$\mathbb{Q}_{\mathbb{T}} = \{Q_R \subset Q_U \mid Q_{IZ} - QS \subset Q_R, |Q_R \cap Q_{S,p}| \geq 1, \forall p \, (p = 1, \ldots, z)\}. \tag{7.3}$$

Procedure MMY considers any element of the set of all possible relaxed Pareto sets $\mathbb{Q}_{\mathbb{T}}$, defined in (7.3), as correct. The procedure also guarantees that the probability of correct selection $P(\text{CS})$ is greater than or equal to a prespecified value P^*. The MMY procedure,

as defined by Yoon (2018), is outlined in Algorithm 17 and will now be discussed. The notation used for the multi-objective ranking and selection problems are:

M_S	the number of solutions in the population.
S	the solution set, *i.e.* $S = \{1, \ldots, M_S\}$.
I	the set of solutions that are still in competition.
H	the number of objectives.
B	the objective set, *i.e.*, $B = \{1, \ldots, H\}$.
μ_{ib}	the unknown true mean of solution i for objective b ($i \in S$ and $b \in B$).
σ^2_{ib}	the unknown variance of solution i for objective b ($i \in S$ and $b \in B$).
N_i	the total number of simulation replications assigned to solution i.
X_{ibl}	the lth observation from solution i for objective b.
$\overline{X}_{ib}(N_i)$	the sample mean of solution i for objective b based on N_i observations.
$S^2_{ib}(N_i)$	the sample variance of solution i for objective b based on N_i observations, *i.e.*

$$S^2_{ib}(N_i) = \frac{1}{N_i - 1} \sum_{l=1}^{N_i} (X_{ibl} - \overline{X}_{ib}(N_i))^2.$$

Q	the true Pareto set based on μ_{ib} ($i \in S$ and $b \in B$).
Q_c	the true non-Pareto set based on μ_{ib} ($i \in S$ and $b \in B$).
Q_{IZ}	the true Pareto set with IZ.
Q^c_{IZ}	the true non-Pareto set with IZ.
Q_R	the true relaxed Pareto set.
$\mathbb{Q}_{\mathbb{T}}$	the set of all possible true relaxed Pareto sets, defined in (7.3).
S_p	the observed Pareto set based on $\overline{X}_{ib}(N_i)$ ($i \in S$ and $b \in B$).
S^c_p	the observed non-Pareto set based on $\overline{X}_{ib}(N_i)$ ($i \in S$ and $b \in B$).
S_{IZ}	the observed Pareto set with IZ.
S^c_{IZ}	the observed non-Pareto set with IZ.
n_0	the number of simulation replications at the first stage.
δ^*_b	the indifference-zone value for objective b.
P^*	the minimum required value for $P(\text{CS})$.

First, let

$$\delta_{ijb} = \max\{\delta^*_b, \overline{X}_{jb}(N_j) - \overline{X}_{ib}(N_i)\}, \tag{7.4}$$

and $\lceil x \rceil$ denote the smallest integer greater than x. Next, consider a pair of solutions (i, j) where solution i is observed as non-dominated and solution j is any other solution in S. This pair is tested in either Step 4 or 5 of Algorithm 17. B_1 and b' can then be defined as

$$B_1 = \{k \mid |\overline{X}_{jb}(N_j) - \overline{X}_{ib}(N_i)| \leq \delta_b^*, b \in B\} \tag{7.5}$$

and

$$b' = \underset{b \in B - B_1}{\arg\max}\, \Phi \left(\frac{\overline{X}_{jb}(N_j) - \overline{X}_{ib}(N_i)}{\sqrt{\frac{S_{ib}^2(N_i)}{N_i} + \frac{S_{jb}^2(N_j)}{N_j}}} \right), \tag{7.6}$$

where Φ is the cumulative distribution function of the standard normal distribution. The variables B_1 and b' should be defined for each pair of (i, j) ($i \in S_p$ and $j \in S$, $j \neq i$). If $B_1 = B$, which states that solutions i and j are observed to be indifferent to each other, then Step 4 of Algorithm 17 should be followed; otherwise, Step 5 should be followed. Therefore, if Step 4 is followed, then Step 5 is skipped and *vice versa*.

In Steps 4 and 5, the constants h_1 and h_2 are used as part of the constraints. These constants are calculated by solving

$$\left[\int_0^\infty \left[\int_0^\infty \Phi \left(\frac{h_1}{\sqrt{(N_i - 1)\frac{1}{x} + (N_j - 1)\frac{1}{y}}} \right) f_1(x)\mathrm{d}x \right] f_2(y)\mathrm{d}y \right]^H = 1 - \gamma \tag{7.7}$$

and

$$\int_0^\infty \left[\int_0^\infty \Phi \left(\frac{h_2}{\sqrt{(N_i - 1)\frac{1}{x} + (N_j - 1)\frac{1}{y}}} \right) f_1(x)\mathrm{d}x \right] f_2(y)\mathrm{d}y = 1 - \gamma, \tag{7.8}$$

where $\gamma = \frac{\beta}{M_S - 1}$, and f_1 and f_2 are the probability density functions of the χ^2 distribution with $N_i - 1$ and $N_j - 1$ degrees of freedom, respectively.

Next, the solutions observed as dominated, *i.e.* $j \in S_p^c$, are considered in Step 7 of Algorithm 17. For each solution $j \in S_p^c$, there exists at least one solution $i \in S_p$, such that solution j is dominated by i; otherwise solution j would not have been observed as dominated. Such a solution $i \in S_p$ should be found for $j \in S_p^c$. If more than one solution in S_p dominates solution j, choose solution i as

$$i = \underset{i' \in S_p, i' \hat{\prec} j}{\arg\max} \prod_{b=1}^{H} \Phi \left(\frac{\overline{X}_{jb}(N_j) - \overline{X}_{i'b}(N_{i'})}{\sqrt{\frac{S_{i'b}^2(N_{i'})}{N_{i'}} + \frac{S_{jb}^2(N_j)}{N_j}}} \right). \tag{7.9}$$

A solution i should be defined for each $j \in S_p^c$. This pair (i, j) $(i \in S_p, j \in S, i \hat{\prec} j)$ is then considered in Step 7 of Algorithm 17. The constant h_3 is also used in this step, and can be calculated by solving

$$\left[\int_0^\infty \left[\int_0^\infty \Phi\left(\frac{h_3}{\sqrt{(N_i-1)\frac{1}{x}+(N_j-1)\frac{1}{y}}}\right) f_1(x)\mathrm{d}x\right] f_2(y)\mathrm{d}y\right]^H = 1-\beta. \quad (7.10)$$

Algorithm 17 The MMY procedure (Yoon, 2018)

1: Select the desired probability of correct selection $P^* = 1-\alpha$, the indifference-zone value δ_b^* for each objective $b \in B$, and the first-stage sample size $n_0 \geq 2$. Set $I = \{1, 2, ..., M_S\}$ and $\beta = \dfrac{\alpha}{M_S}$.

2: Simulate n_0 replications for all M_S solutions, and calculate sample means $\overline{X}_{ib}(n_0)$ and sample variances $S_{ib}^2(n_0)$ ($i \in S$ and $b \in B$). Let $N_i = n_0$.

3: Observe the Pareto set S_p and the non-Pareto set S_p^c based on the sample means $\overline{X}_{ib}(N_i)$ ($i \in S$ and $b \in B$) without the indifference-zone concept.

4: For each solution $i \in S_p$ and $j \in S (j \neq i)$ with $B_1 = B$, check if the following two conditions are met:

$$N_i \geq \left\lceil \max_b \left(\frac{h_1 S_{ib}(N_i)}{\delta_{ijb}} \right)^2 \right\rceil \text{ and } N_j \geq \left\lceil \max_b \left(\frac{h_1 S_{jb}(N_j)}{\delta_{ijb}} \right)^2 \right\rceil, \tag{7.11}$$

where δ_{ijb} and B_1 are defined in (7.4) and (7.5), respectively, and h_1 is calculated using (7.7).

5: For each solution $i \in S_p$ and $j \in S (j \neq i)$ with $B_1 \neq B$, check if the following two conditions are met:

$$N_i \geq \left\lceil \left(\frac{h_2 S_{ib'}(N_i)}{\delta_{ijb'}} \right)^2 \right\rceil \text{ and } N_j \geq \left\lceil \left(\frac{h_2 S_{jb'}(N_j)}{\delta_{ijb'}} \right)^2 \right\rceil, \tag{7.12}$$

where b' is defined in (7.6) and h_2 is the solution to (7.8).

6: Delete solution i from I if conditions (7.11) or (7.12) are satisfied for all $j \in S$ ($j \neq i$).

7: For each solution $j \in S_p^c$, find solution $i \in S_p$ as defined in (7.9). Check if the following two conditions are met:

$$N_i \geq \left\lceil \max_b \left(\frac{h_3 S_{ib}(N_i)}{\delta_{ijb}} \right)^2 \right\rceil \text{ and } N_j \geq \left\lceil \max_b \left(\frac{h_3 S_{jb}(N_j)}{\delta_{ijb}} \right)^2 \right\rceil, \tag{7.13}$$

where h_3 is the solution to (7.10).

8: Delete solution j from I if the conditions in (7.13) are satisfied.

9: If $|I| = 0$, then stop and present the current Pareto set S_p as the final solution set. Otherwise, for each solution $i \in S_p \cap I$, that is, solutions in S_p that were not deleted from I in Step 6, add solution $j \in S$ ($j \neq i$) to I if it does not satisfy conditions (7.11) or (7.12). Similarly, for each solution $j \in S_p^c \cap I$, that is, solutions in S_p^c that were not deleted from I in Step 8, add the corresponding solution $i \in S_p$ to I if it does not satisfy (7.13). Go to Step 10.

10: Take one additional observation X_{i,b,N_i+1} from each solution $i \in I$, and set $N_i \leftarrow N_i + 1$ ($\forall i \in I$). Set $I = \{1, 2, \ldots, M_S\}$ and update $\overline{X}_{ib}(N_i)$ and $S_{ib}^2(N_i)$ for all $i \in S$ and $b \in B$. Go to Step 3.

Now that the MMY procedure has been discussed, it can be applied to the bi-objective JSP defined in this research project. Due to the novelty of this procedure, this application will be the first of its kind. Before the MMY procedure can be applied, the NSGAII was again applied to Test scenario 9 and the MO2Opt algorithm to Test scenarios 10 and 11, to generate a set of 50 solutions for each test scenario. This set of 50 solutions for all three test scenarios consists of both dominated and non-dominated solutions. For each of these solutions in the different test scenarios, the simulation model was run 1 000 times independently as a preliminary step in order to estimate the unknown true means of the two objectives as best as possible. The sample means of these 1 000 simulation runs are considered to be very close to the unknown true means due to the strong law of large numbers (Law and Kelton, 2000). The process of running the simulation model for 1 000 replications for each member in the set of 50 solutions, was only done for experimental reasons. The actual process that should be used when the MMY procedure is applied in the real-world manufacturing environment, is that 10 replications should be run for each member in the solution set, after which Step 3 of Algorithm 17 should start. Each step in the algorithm should then be followed, and if more simulation replications are required (*i.e.* if N_i was incremented by one) then another replication is conducted for each solution i that forms part of set I. This will ensure that the correct number of simulation replications is conducted and will prevent the use of extra computational resources that were not required.

It should be noted that the values for the IZ variables δ_b^* are chosen arbitrarily by the user of the system. If δ_b^* is set to a lower value, the procedure will require more simulation replications to find the relaxed Pareto set while also guaranteeing the probability of correct selection of the best solutions. The researcher decided to use δ_b^* values that are less than 15 percent of the difference between the maximum and minimum of the true means for each objective; for example, if the maximum and minimum of the true means for the makespan objective are 150 hours and 120 hours respectively, then δ_1^* will be selected as 4.5 hours. The reason why the researcher chose this small percentage to calculate each IZ value, is to ensure that the procedure does not include solutions in the relaxed Pareto set that are clearly outperformed. The application of the Procedure MMY to each of the test scenarios will now be discussed.

7.4.1 Test scenario 9 MMY results

The estimated true means of the 50 solutions of Test scenario 9 are presented in Figure 7.9. When the MMY procedure was applied to the generated data, the IZ value for each objective was set to $\delta_1^* = 4.25$ hours and $\delta_2^* = 2$ hours respectively. Both the IZ values that were selected are within 15 percent of the difference between the maximum and minimum

of the true means for each objective, as was defined previously. The probability of correct selection $P(\text{CS})$ was chosen to be 0.90, which means that the solutions included in the relaxed Pareto set were chosen correctly with a probability of 90 percent. The first stage sample size was set to be $n_0 = 10$. The procedure requires $n_0 \geq 2$, but if n_0 is chosen very small, initial large variance estimations are obtained. That is why n_0 was set to 10. When the true means in Figure 7.9 are examined, we obtain the true Pareto set $Q = \{12,50\}$, illustrated with red dots. These true means were determined by calculating the mean of the results of all 1 000 simulation runs that were done for each solution in the set. The MMY procedure then identified the relaxed Pareto set as $Q_R = \{12,50\}$, which is the same as Q. The total number of simulation replications required for the solutions in Q_R were $N_{12} = 621$ and $N_{50} = 621$. The resulting mean values of both objectives for each solution in Q_R are provided in Table 7.15, and are also displayed as green squares on Figure 7.9. The results of the MMY procedure suggests that with a confidence of 90 percent, the Pareto set will include solutions 12 and 50. Moreover, the MMY procedure only required 621 simulation replications for these two solutions to be included in the relaxed Pareto set, instead of the 1 000 simulation replications that were performed at the start.

Table 7.15: Mean values of both objectives for each solution in Q_R in Test scenario 9

Solution	Makespan (hours)	Total Overtime (hours)
12	156.28	68.87
50	159.36	61.05

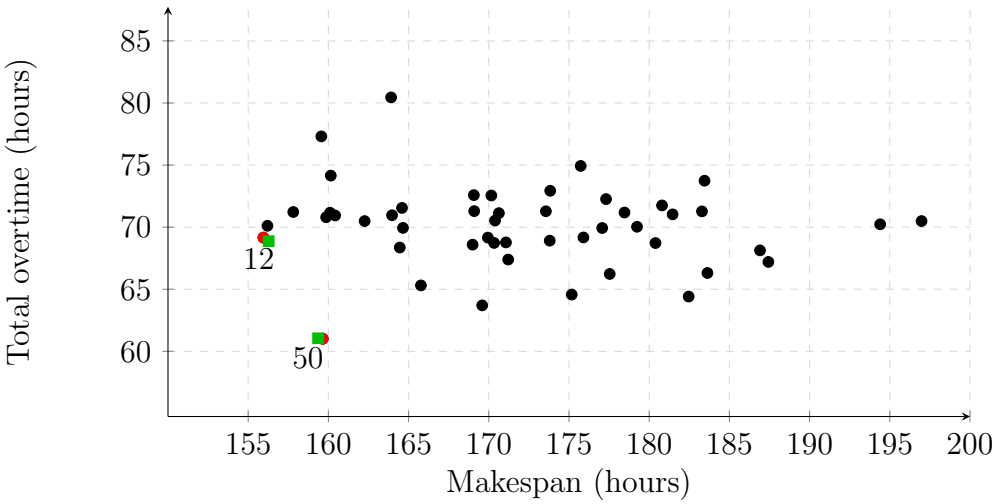


Figure 7.9: The estimated true means of the 50 solutions of Test scenario 9

7.4.2 Test scenario 10 MMY results

The estimated true means of the 50 solutions of Test scenario 10 are presented in Figure 7.10. When the MMY procedure was applied to the generated data, the IZ value for each objective was set to $\delta_1^* = 30$ hours and $\delta_2^* = 17$ hours respectively. The value for δ_1^* was correctly selected within 15 percent of the difference between the maximum and minimum of the true means for the makespan objective; however, when δ_2^* was also chosen within the 15 percent range, the procedure required $N_i \to \infty$ to find the relaxed Pareto set. This large number of simulation replications required to distinguish between the solutions, is due to the true means of the solutions that are very close together. For that reason, the IZ value for the total overtime objective was increased to $\delta_2^* = 17$ hours, to overcome this obstacle. The probability of correct selection $P(\text{CS})$ was chosen to be 0.90, which means that the solutions included in the relaxed Pareto set were chosen correctly with a probability of 90 percent. The first stage sample size was set to be $n_0 = 10$. When the true means in Figure 7.10 are examined, we obtain the true Pareto set $Q = \{1,4,29,43,46\}$, illustrated with red dots. However, the MMY procedure identified the relaxed Pareto set as $Q_R = \{1,4,32,43,46\}$. This result suggests that solutions 1 and 32 are observed to be indifferent to each other. It can also be observed that solution 29 is not part of the relaxed Pareto set. The total number of simulation replications assigned to the solutions in Q_R was $N_1 = 721$, $N_4 = 721$, $N_{32} = 762$, $N_{43} = 963$ and $N_{46} = 762$. The resulting mean values of both objectives for each solution in Q_R are provided in Table 7.16, and are also displayed as green squares on Figure 7.10. The observed members of the Pareto set were selected with a guaranteed probability of correct selection, due to Procedure MMY.

Table 7.16: Mean values of both objectives for each solution in Q_R in Test scenario 10

Solution	Makespan (hours)	Total Overtime (hours)
1	1520.87	1252.83
4	1568.34	1245.32
32	1502.03	1258.74
43	1465.53	1272.14
46	1469.63	1260.25

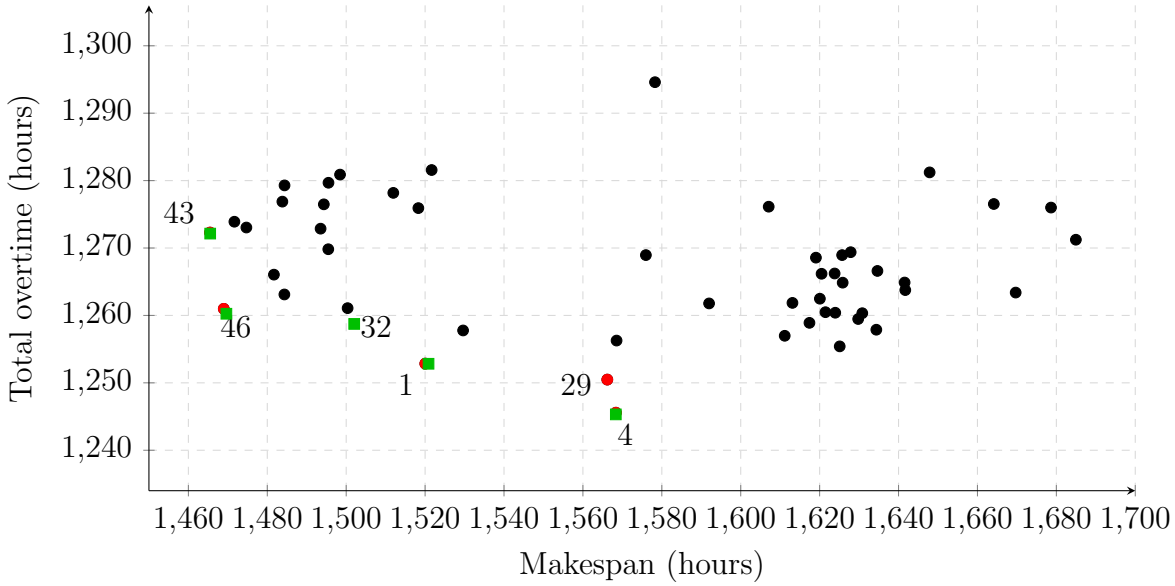


Figure 7.10: The estimated true means of the 50 solutions of Test scenario 10

7.4.3 Test scenario 11 MMY results

The estimated true means of the 50 solutions of Test scenario 11 are presented in Figure 7.11. When the MMY procedure was applied to the generated data, the IZ value for each objective was set to $\delta_1^* = 60$ hours and $\delta_2^* = 22$ hours respectively. The value for δ_1^* was correctly selected within 15 percent of the difference between the maximum and minimum of the true means for the makespan objective; however, when δ_2^* was also chosen within the 15 percent range, the procedure required $N_i \to \infty$ to find the relaxed Pareto set. This large number of simulation replications required to distinguish between the solutions, is due to the true means of the solutions being very close together. For that reason, the IZ value for the total overtime objective was increased to $\delta_2^* = 22$ hours, to overcome this obstacle. The probability of correct selection $P(\text{CS})$ was chosen to be 0.90, which means that the solutions included in the relaxed Pareto set were chosen correctly with a probability of 90 percent. The first stage sample size was set to be $n_0 = 10$. When the true means in Figure 7.11 are examined, we obtain the true Pareto set $Q = \{3,14,19\}$, illustrated with red dots. However, the MMY procedure identified the relaxed Pareto set as $Q_R = \{3,14,19,27\}$. This means that solutions 19 and 27 are observed as indifferent to each other, which is why solution 27 is included in the relaxed Pareto set. The total number of simulation replications assigned to the solutions in Q_R was $N_3 = 959$, $N_{14} = 958$, $N_{19} = 961$ and $N_{27} = 948$. The resulting mean value of both objectives for each solution in Q_R are provided in Table 7.17, and are

also displayed as green squares on Figure 7.11. The observed members of the Pareto set were selected with a guaranteed probability of correct selection, due to Procedure MMY.

Table 7.17: Mean values of both objectives for each solution in Q_R in Test scenario 11

Solution	Makespan (hours)	Total Overtime (hours)
3	2676.62	2359.93
14	2658.36	2369.76
19	2672.94	2367.35
27	2676.15	2363.93

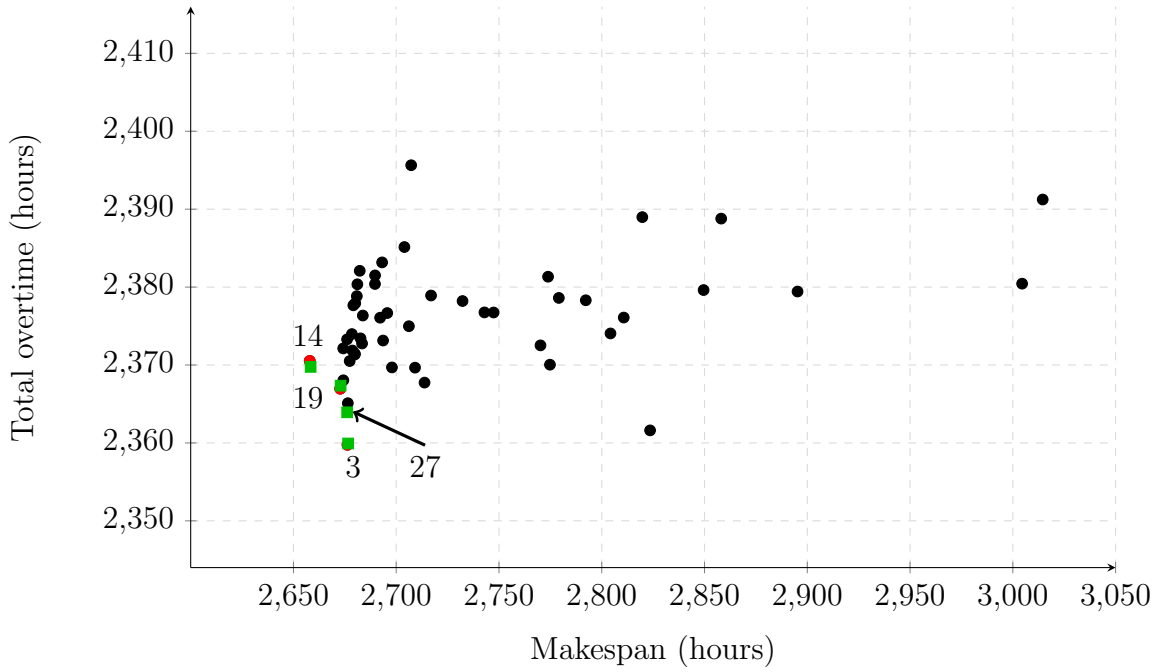


Figure 7.11: The estimated true means of the 50 solutions of Test scenario 11

Now that Procedure MMY has been applied to all three test scenarios, the performance of the procedure can be analysed. Table 7.18 provides the performance metric values for the different test scenarios after the MMY procedure was applied. The table also contains the number of simulation replications required to find the relaxed Pareto set while still fulfilling the probability of correct selection of the best solutions. The sets that were compared were the true Pareto set and the relaxed Pareto set. Considering these performance metrics, the GD value should be zero and the HR and MS values should be one for the relaxed Pareto set to be considered equal to the true Pareto set. The MMY procedure did perform well

in all three test scenarios, as it achieved small GD values compared to those in Table 7.5, as well as HR and MS values close to one in all three test scenarios. The procedure also determined the relaxed Pareto set while using less simulation replications compared to the true Pareto set that required 1 000 replications per member.

Table 7.18: Mean performance metric values for the MMY procedure and the different test scenarios

Test scenario	GD	HR	MS	Simulation replications
50 jobs	0.010	1.000	0.977	$N_{12} = 621$ $N_{50} = 621$
100 jobs	3.801	1.000	1.000	$N_1 = 721$ $N_4 = 721$ $N_{32} = 762$ $N_{43} = 963$ $N_{46} = 762$
200 jobs	1.076	1.000	0.945	$N_3 = 959$ $N_{14} = 958$ $N_{19} = 961$ $N_{27} = 948$

This concludes the discussion of the application of Procedure MMY to the bi-objective JSP that was defined in this research project. Next, a synthesis will be provided to critically assess all the observations that were made throughout the implementation of the five different metaheuristics and the MMY procedure.

7.5 Synthesis of methods applied to bi-objective JSP

This first observation that can be made from the implementation of the different metaheuristics, is that only the MOSA, MOGA and the NSGAII have been widely used in literature to solve multi-objective JSPs. The lack of literature that was found on the implementation of the 2-opt move in a hybrid algorithm to solve a multi-objective JSP, suggests that the MO2Opt algorithm presented in this chapter is one of the first, if not the first, implementation of this algorithm to solve this kind of problem. The same observation can be made for the MOOCEM, which suggests it is the first implementation of this method to solve a multi-objective JSP. The average computation time for each of the metaheuristics and each test scenario, are provided in Table 7.19, where the time is

illustrated in "Hours:Minutes:Seconds". The specifications of the computer that was used for these tests are:

- Processor – Intel(R) Xeon(R) CPU W3550 @ 3.07GHz
- Installed RAM – 12 GB.

Table 7.19: Computational time for each metaheuristic and the different test scenarios

Metaheuristic	Test scenario 9	Test scenario 10	Test scenario 11
MOSA	0:01:18	0:02:36	0:06:58
MOOCEM	0:02:11	0:04:00	0:12:07
NSGAII	0:01:41	0:04:06	0:10:26
MOGA	0:01:32	0:02:49	0:06:27
MO2Opt	0:01:11	0:03:57	0:11:00

From the results of the paired t-tests conducted to compare the outcomes of each metaheuristic, it indicated that the MO2Opt algorithm did not perform as well as other metaheuristics in the small test scenario, but as the size of the problem increased, it performed better. The only drawback of the MO2Opt algorithm is that it does not perform as well as some of the other metaheuristics when considering the MS performance metric. This can potentially be attributed to the selection of a Pareto member on which the 2-opt move must be performed. By only selecting from the current Pareto set, the MO2Opt algorithm generates good solutions but fails to produce a diverse set of solutions. A possible remedy to this problem is to change the rule for selecting members to not only select from the current Pareto set, but also from some of the other solutions in the current population. This remedy can however have a negative effect on the performance of this algorithm in the other performance metrics. Further research can be conducted to improve on the MO2Opt algorithm that has been presented in this project.

The objectives that were identified and used in this project have not been widely used in literature. The researcher found limited literature on the use of these objectives in multi-objective JSPs. The literature that was found detailed the use of these objectives in deterministic job shops with smaller instances that were used for testing. What differentiates the research presented in this project from the previous implementations, is

- the use of stochastic processing times, and
- larger test instances.

The work done in this project, therefore, contributes to the limited literature on using the *makespan* and *total overtime* as objectives when solving a multi-objective JSP.

The last observation that can be made is the successful use of the MMY procedure as part of the simulation scheduler presented in this project. Due to the novelty of this procedure, this project documents the first implementation of this procedure on a multi-objective JSP. This procedure was used to determine the approximate Pareto set and guarantee the probability of correct selection of the members in these sets. When this procedure was applied to Test scenarios 10 and 11, the researcher observed that the procedure required $N_i \to \infty$ to distinguish between solutions and find the relaxed Pareto set when the IZ value are chosen too small. This is due to the true mean values of solutions that are close. To overcome this obstacle, the researcher had to increase the IZ values for the objectives. After increasing the IZ values, the procedure successfully found the relaxed Pareto set in both test scenarios.

7.6 Chapter summary

This chapter documented the expansion of the simulation scheduler from the single-objective to the multi-objective domain. The chapter started off with the selection and discussion of the objectives that were used in the scheduler. Thereafter, the metaheuristics that were incorporated into the simulation scheduler were discussed. Comparison tests of the performance of each metaheuristic then followed. Finally, the chapter also documented the implementation of a newly developed ranking and selection technique, *MMY*, to determine the approximate Pareto set and guarantee probability of correct selection of the best solutions.

Chapter 8

Project conclusion

This chapter describes the conclusion of the research project. It contains a summary of the work that was covered in the project, as well as the research contributions of this project. Future research opportunities will also be discussed. This chapter will end off with a self-assessment and reflection of the project.

8.1 Project summary

This section provides a summary of the work that was done in this project. It will refer to the goals and objectives set out at the start of the project, and what was done to fulfil them.

The first objective was to create an operational simulation scheduler that generates new production schedules for a job shop. To achieve this objective, the researcher had to acquire knowledge of *simulation-based real-time scheduling systems* as well as *scheduling of job shops*. From literature it was gathered that there are multiple dispatching rules that can be applied to generate schedules for the job shop; however, due to the vast number of rules, the researcher focused only on six common dispatching rules. It was also found that if these rules are applied on job-level, they will always produce a feasible schedule for the job shop. Performance indicators were identified and the mathematical models to determine these values were discussed. Further literature that also needed to be reviewed to achieve the stated objective included the use of simulation modelling to solve the JSP. All the literature that was reviewed was documented in **Chapter 2**.

Before the development of the proposed system could commence, an architecture had to be created to define how the components of the system should interact with each other. Literature regarding the creation of software architectures was documented in **Chapter 3** and the architecture created for this project was documented in **Chapter 4**. This architecture defined that the simulation scheduler should be cloud-based, which is why the researcher set up a computer server which can be accessed via the internet. The simulation scheduler was then developed in *Tecnomatix Plant Simulation* on the server. The scheduler incorporated the dispatching rules to generate schedules, as well as the performance indicators to compare the rules to determine which rule performed best. The development of the simulation scheduler was documented in **Chapter 5**, which fulfils Objective 1.

The second objective was to create the information system to store data, as well as web pages for capturing and displaying data. The information system was also defined in the architecture to be cloud-based, which is why the researcher chose *Google Cloud Platform* as a cloud service provider to host the information system. The relational model and data dictionary of the information system was discussed in **Chapter 5**. Regarding the development of the web pages for capturing and displaying data, these were also set up to send a notification to the user or manager of the shop floor when a data change was logged. There was also a web page created for displaying the report that was generated by the simulation scheduler, which will enable the manager to determine the best performing dispatching rule. Finally, a web page was created where the manager could then select the new schedule to be implemented. The development of the web pages was also documented in **Chapter 5**, which fulfilled Objective 2.

The third objective was to develop a sensorised job shop that captures the movement of jobs through the shop in real time. This was accomplished by conducting the necessary research on what components would be required to track the jobs through the shop. It was finally decided that the movement of jobs would be tracked with RFID cards. A specific card is linked to a job and as the job arrives at a machine, the card could be swiped and a new status selected on the sensor with the press of a button. The design and functionality of the sensor is described in detail in **Chapter 5**, which fulfils Objective 3.

The final objective was to construct a cloud-based server to host the web pages, simulation scheduler and information system. The cloud-based server that was created on *Google Cloud Platform* hosts the simulation scheduler, information system and web pages. Due to all these components being cloud-based, it enables the user of the system to access each of the components via the internet, which fulfils Objective 4.

After the first four objectives were successfully addressed, the system was tested extensively to determine if the system was running as expected. Five test scenarios were created, and the results of these tests proved that the system is operational and provides the correct schedules. The tests and their results are discussed in **Appendix B** and **C**.

The functionality of the simulation scheduler was then extended to incorporate three different metaheuristics. The metaheuristics that were selected for implementation in the simulation schedule include the simulated annealing and 2-opt algorithms. It was also decided to create a hybrid simulated annealing algorithm that used a 2-opt move as a local search scheme to find new solutions. Three more test scenarios were created to compare the performance of the dispatching rules to the performance of the metaheuristics. It could then be concluded that all the metaheuristics outperformed the dispatching rules. The formulation of the metaheuristics was discussed in **Chapter 6**.

The next objective was to further expand the simulation scheduler from the single-objective to multi-objective optimisation domain. This was achieved by selecting two strongly conflicting objectives, *i.e.* makespan and total overtime, as well as implementing five multi-objective metaheuristics in the simulation scheduler. The metaheuristics that were selected are the Multi-objective Simulated Annealing (MOSA), Multi-objective Genetic Algorithm (MOGA), Non-dominated Sorting Genetic Algorithm II (NSGAII), Multi-objective Optimisation Cross-Entropy Method (MOOCEM) and the Multi-objective 2-opt Move Algorithm (MO2Opt). Both the MOOCEM and MO2Opt have not previously been used to solve the multi-objective JSP. After these metaheuristics were successfully implemented, 500 experiments were completed for each metaheuristic on three different problem sizes and comparison tests were conducted on three different performance metrics, *i.e.* generational distance, hyper area ratio and maximum spread. From the paired t-tests it was concluded that the NSGAII performed best for the smallest problem size of 50 jobs. In both the 100 and 200 job test scenarios, the MO2Opt algorithm performed best. The incorporation of these metaheuristics to solve the bi-objective JSP was documented in **Chapter 7** and this fulfils Objective 5.

For the final objective, a newly developed ranking and selection procedure, called Procedure MMY, was implemented in the simulation scheduler, to determine the approximate Pareto set and guarantee the probability of correct selection of the members in these sets. The implementation of Procedure MMY was documented in **Chapter 7**, fulfilling Objective 6. Due to the successful development of the proposed system, the researcher succeeded in fulfilling the purpose of the research project. This concludes the project summary, which will be followed by the research contributions.

8.2 Research contribution

In this section, the research contributions of this project will be discussed, focusing on the utility of the research and how the utility was demonstrated. Hevner *et al.* (2004) provides five testing statements through which the research contributions could be evaluated to determine their significance, and in doing so, proving the utility of the research in the project.

(1) *"If existing artefacts are adequate, then the research that creates a new artefact is unnecessary."*

Existing artefacts for the stochastic bi-objective JSP are not adequate for a number of reasons. First, to the researcher's knowledge, this project demonstrates the

first implementation of the *MOOCEM* and *MO2Opt* methods to solve the stochastic bi-objective JSP. The MO2Opt method performed the best in larger test instances, compared with a number of different metaheuristics that were incorporated in the simulation scheduler. Secondly, this project provides the first implementation of the newly developed ranking and selection procedure, called Procedure *MMY*, on the stochastic bi-objective JSP. This procedure was implemented to determine the approximate Pareto set and guarantee the probability of correct selection of members in these sets. This project also documented the use of two strongly conflicting objectives, *i.e. makespan* and *total overtime*, in the stochastic bi-objective JSP. Currently, there is limited research regarding the use of these objectives, and the research that has incorporated these objectives was on deterministic JSPs. To the researcher's knowledge, this project presents the first use of these objectives to solve a stochastic JSP. Lastly, a number of research projects have previously been undertaken to develop a working job shop scheduler, that incorporates information systems, sensorised workstations and simulation modelling; however, this project documented the first successful integration of a system that has sensorised workstations as well as a *cloud-based* information system and simulation scheduler, focused on solving the stochastic bi-objective JSP.

(2) *"If the new artefact does not map adequately to the real world, it cannot provide utility."*

The proposed scheduling system has been designed and developed to represent and mimic the behaviour of a real-world job shop, and therefore it adequately maps to the real world. The simulation scheduler was designed to cope with possible disruptions, such as a machine that fails, an operator that is absent, or a new job that enters the system, by generating schedules that include these disruptions. In this project, stochastic processing times are considered, because they are a better representation of the real-world environment, as the exact processing time of a job is not always known beforehand. The simulation scheduler was also expanded from a single-objective to a multi-objective optimisation domain to more adequately mimic the decision variables of a real-world job shop.

(3) *"If the artefact does not solve the problem, it has no utility."*

The metaheuristics that were implemented in the simulation scheduler produced better solutions compared with the solutions of the common dispatching rules found in literature. Therefore, metaheuristics were chosen for implementation in the bi-objective simulation scheduler. All the metaheuristics that were incorporated in the simulation scheduler provided feasible solutions from which the best solutions could be selected and illustrated in Pareto sets. These Pareto sets were then also subjected to Procedure MMY which determined the approximate Pareto sets. Due to all the methods in the scheduler generating feasible solutions, the scheduler proves that it can solve the JSP defined in this project.

(4) *"If utility is not demonstrated, then there is no basis upon which to accept the claims that it provides any contribution."*

First, validation tests were performed to test whether the simulation scheduler generated schedules correctly according to the different dispatching rules. The sensors were also tested to determine whether the appropriate data changes occurred. Operational tests were also conducted to determine whether the simulation scheduler would adapt the schedules according to disruptions that could occur, such as a machine that fails, an operator that is absent, or a new job that enters the system. All these tests proved successful which demonstrates the utility of the system. Furthermore, the metaheuristics that were incorporated into the simulation scheduler were also extensively tested to determine the best performing metaheuristic for three different test scenarios.

(5) *"If the problem, the artefacts, and their utility are not presented in a manner such that the implications for research and practice are clear, then publication in the literature is not appropriate."*

The final validation of the research project depends on whether it is deemed acceptable for publication. It is the decision of the examiners if the work presented in this research document is acceptable for publication. However, the research conducted in this project has been subjected to multiple double-blind peer review processes and subsequently led to four publications, *i.e.* Snyman and Bekker (2017), Snyman *et al.*

(2018), Snyman and Bekker (2019a) and Snyman and Bekker (2019b). More publications are expected to come from the findings in this project. These publications show the significance of the research in this project in the area of JSPs.

8.3 Future work

This section will highlight areas where further research may be possible in the near future.

- Although the MO2Opt algorithm that has been developed in this project performed best in the larger test scenarios, the Pareto set it found did not perform well in the maximum spread performance metric. This occurrence could be due to only selecting members part of the current Pareto set, on which the 2-opt move must be performed. Further research can be conducted to experiment with different selection strategies to determine which provide the best Pareto sets.
- The research in this project was done solely on the JSP without taking the raw material inventory into account. The system and architecture provided in this project can therefore be used as the basis for future research where inventory management is also incorporated into the system. This will add to the complexity of the current system, but will provide a better representation of the real-world job shop environment.
- The system that was developed can be implemented in a real-life job shop environment. An industry partner should therefore be approached for possible implementation of the developed system.

This concludes the future research opportunities. The next section will provide a self-assessment of the project and what was accomplished.

8.4 Self-assessment of project

At the start of the project, the task of developing such a complex system consisting of several components seemed particularly daunting. Good time management proved to be key to successfully completing the stated purpose of the project. A considerable amount of time was spent on research and literature review to fully understand the problem before attempting to develop the different components. It also helped the researcher to separate the different components and finish each part of the development before the final integration of all components.

Furthermore, due to this lack of knowledge and experience in working with electronic components, the researcher found it very interesting to work with these components which

industrial engineers do not frequently come into contact with. However, the lack of experience caused the researcher to spend a considerable amount of time and effort in developing operational sensors. The sensors were, however, developed to account for human error, *i.e.* an operator swipes a card twice or swipes a card at the wrong machine.

Although the simulation scheduler is operational, it was created only to solve the job shop problem for a shop with eight machines of specific types. It is currently not flexible enough to adapt if there is a change in the number of machines present in the system or if a machine is replaced with a newer model. The scheduler will therefore have to be made more flexible if such a scenario should come about.

The integration of the different components into a functioning system, fascinated the researcher. The researcher also learned to utilise more of the functionalities of programs like *Tecnomatix Plant Simulation*, *Visual Studio* and *MySQL*, which are common tools that are used by industrial engineers in industry. The work that was done to formulate and adapt metaheuristics to solve the JSP described in this project, was a unique and enjoyable experience for the researcher.

8.5 Reflection

The *Fourth Industrial Revolution* and its global impact are growing rapidly due to the intense research that has been conducted with regard to *digitisation*, the *Internet of Things*, and smart knowledge and systems. There is still considerable debate as to the best way to exploit the fast pace of technological innovation and one of the trends that companies currently follow, is to force the implementation of the best and most modern technologies on the market. These companies are so focussed on staying up to date with the technologies they use, instead of first determining what is actually required to satisfy the needs of the stakeholders. This trend discourages smaller companies to invest in the concept of Industry 4.0, as they do not have the financial means to implement these types of technologies.

This project is a good example of developing a system that satisfies the needs of all stakeholders while also considering the affordability of the solution. The system that was developed in this project required cheap sensor technology as well as cheap cloud data storage. The project also illustrated that it is possible to construct a digital replica of a real-world shop on readily available simulation software packages which can then provide sufficiently high quality schedules through sophisticated optimisation algorithms. The work done in this project therefore suggests that it is not a prerequisite to implement the best and most modern technologies, to exploit the value of the Fourth Industrial Revolution.

The project also provides encouragement to smaller companies that do not have the financial means to implement the best technologies, by illustrating how cheap and readily available software and technology can be developed and integrated into a Industry 4.0 type of solution.

8.6 Chapter summary

This chapter described the conclusion of the research project. It contains a summary of the work that was covered in the project, as well as the research contributions of this project. Future research opportunities were also discussed. This chapter ended with a self-assessment and reflection of the project.

References

AA Portable Power Corp. Specification Cylindrical Li-ion 18650 2600mAh Rechargeable cell. http://www.batteryspace.com/prod-specs/18650_2600.pdf, 2004. [Online, Accessed 1 March 2018].

J. Adams, E. Balas, and D. Zawack. The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*, 34(3):391–401, 1988. DOI: https://doi.org/10.1287/mnsc.34.3.391.

M. Adibi, M. Zandieh, and M. Amiri. Multi-Objective Scheduling of Dynamic Job Shop using Variable Neighborhood Search. *Expert Systems with Applications*, 37:282–287, 2010. DOI: https://doi.org/10.1016/j.eswa.2009.05.001.

M. Andresen, H. Bräsel, M. Mörig, J. Tusch, F. Werner, and P. Willenius. Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop. *Mathematical and Computer Modelling*, 48(7):1279–1293, 2008. DOI: https://doi.org/10.1016/j.mcm.2008.01.002.

Ardu Shop. USB 2.0 TO TTL UART ON FTDI FT232RL (Arduino Pro Mini Programmer). https://ardushop.ro/en/home/86-usb-20-to-ttl-uart-on-stc-cp2120-arduino-pro-mini-programmer.html, 2017. [Online, Accessed 20 March 2018].

Arduino. Getting started with the Arduino Pro Mini. https://www.arduino.cc/en/Guide/ArduinoProMini, 2017. [Online, Accessed 1 November 2017].

P. Aurich, A. Nahhas, T. Reggelin, and J. Tolujew. Simulation-based Optimization for Solving a Hybrid Flow Shop Scheduling Problem. In *Proceedings of the 2016 Winter Simulation Conference*, pages 2809–2819, 2016. DOI: https://doi.org/10.1109/WSC.2016.7822317.

M. E. Aydin and T. C. Fogarty. A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application. *Journal of Intelligent Manufacturing*, 15(6):805–814, 2004. DOI: https://doi.org/10.1023/B:JIMS.0000042665.10086.cf.

M. E. Aydin and E. Öztemel. Dynamic job shop scheduling using intelligent agents. *Robotics and Autonomous Systems*, 33:169–178, 2000. DOI: http://dx.doi.org/10.1016/S0921-8890(00)00087-7.

K. R. Baker. Minimizing earliness and tardiness costs in stochastic scheduling. *European Journal of Operational Research*, 236(2):445–452, 2014. DOI: http://dx.doi.org/10.1016/j.ejor.2013.12.011.

S. Bandyopadhyay and R. Bhattacharya. *Discrete and Continuous Simulation: Theory and Practice*. CRC Press, illustrate edition, 2014. DOI: https://doi.org/10.1201/b17127.

Bang Good. TP4056 5V 1A Lipo Battery Mini USB Charging Board Charger Module. https://www.banggood.com/TP4056-1A-Lipo-Battery-Charging-Board-Charger-Module-Mini-USB-Interface-p-1027027.html?cur_warehouse=CN, 2016. [Online, Accessed 20 March 2018].

S. Bangsow. *Tecnomatix Plant Simulation. Modeling and Programming by Means of Examples*. Springer, 2015. DOI: https://doi.org/10.1007/978-3-319-19503-2.

J. Banks. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. A Wiley-Interscience publication. Wiley, 1998. DOI: https://doi.org/10.1002/9780470172445.

J. Banks, B. L. Nelson, J. S. Carson, and D. M. Nicol. *Discrete-Event System Simulation*. Pearson Education, 4 edition, 2010. DOI: https://doi.org/10.2307/1268124.

J. W. Barnes and J. B. Chambers. Solving the job shop scheduling problem with tabu search. *IIE Transactions*, 27(2):257–263, 1995. DOI: https://doi.org/10.1080/07408179508936739.

A. Bauer, B. Bullnheimer, R. F. Hartl, and C. Strauss. An ant colony optimization approach for the single machine total tardiness problem. In *In Proceedings of the 1999 congress on evolutionary computation*, pages 1445–1450. IEEE Press New York, 1999. DOI: https://doi.org/10.1109/CEC.1999.782653.

K. Beck. *Extreme Programming Explained: Embrace Change*. Boston: Addison-Wesley Publishing Co., 2000. ISBN 0201616416.

J. Bekker. Simulation 442: Short notes on aspects of Discrete-event Simulation. University Lecture, 2016a.

J. Bekker. The Genetic algorithm and Simulation-optimisation – a brief introduction, 2016b.

J. Bekker and C. Aldrich. The cross-entropy method in multi-objective optimisation: An assessment. *European Journal of Operational Research*, 2011:112–121, 2011. DOI: https://doi.org/10.1016/j.ejor.2010.10.028.

G. Bengü. A simulation-based scheduler for flexible flowlines. *International Journal of Production Research*, 32(2):321–344, 1994. DOI: https://doi.org/10.1080/00207549408956936.

H. Benítez-Pérez and C. Miguel. RECONFIGURABLE DISTRIBUTED SYSTEM BASED ON SOM NETWORK APPROACH. *IFAC Proceedings Volumes*, 38(2):200–205, 2005. DOI: https://doi.org/10.3182/20051114-2-MX-3901.00028.

S. Bergmann, N. Feldkamp, and S. Strassburger. Approximation of Dispatching Rules for Manufacturing Simulation using Data Mining Methods. In *Proceedings of the 2015 Winter Simulation Conference*, pages 2329–2340, 2015. DOI: https://doi.org/10.1109/WSC.2015.7408344.

O. Bilkay, O. Anlagan, and S. E. Kilic. Job shop scheduling using fuzzy logic. *The International Journal of Advanced Manufacturing Technology*, 23(7-8):606–619, 2004. DOI: https://doi.org/10.1007/s00170-003-1771-2.

W. Birch, G. Wirth, K. Mick, and N. Hennegan. Use of simulation and modeling for efficient and effective scheduling of offshore supply vessels. https://www.simio.com/resources/papers/WinterSim2015/Scheduling-Offshore.php, 2015. [Online, Accessed 4 June 2017].

J. Branke and W. Zhang. A new myopic sequential sampling algorithm for multi-objective problems. In *In Proceedings of the 2015 Winter Simulation Conference*, pages 3589–3598. IEEE Press, 2015.

P. Brucker, B. Jurisch, and B. Sievers. A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics*, 49(1-3):107–127, mar 1994. DOI: https://doi.org/10.1016/0166-218X(94)90204-6.

G. Buttazzo, G. Lipari, L. Abeni, and M. Caccamo. *Soft Real-Time Systems: Predictability vs. Efficiency*. Springer, 2005. DOI: https://doi.org/10.1007/0-387-28147-9.

X. Cai, X. Wu, and X. Zhou. *Optimal Stochastic Scheduling*. International Series in Operations Research & Management Science. Springer US, 2014. DOI: https://doi.org/10.1007/978-1-4899-7405-1.

B. Çaliş and S. Bulkan. A research survey: review of AI solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing*, 26(5):961–973, 2015. DOI: https://doi.org/10.1007/s10845-013-0837-8.

L. Calvet, A. A. Juan, V. Fernandez-Viagas, and J. M. Framinan. Combining Simulation with Metaheuristics in Distributed Scheduling Problems with Stochastic Processing Times. In *Proceedings of the 2016 Winter Simulation Conference*, pages 2347–2357, 2016. DOI: https://doi.org/10.1109/WSC.2016.7822275.

U. K. Chakraborty. *Computational Intelligence in Flow Shop and Job Shop Scheduling*. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2009. DOI: https://doi.org/10.1007/978-3-642-02836-6.

I. Chaudhry and A. Khan. A Research Survey: Review of Flexible Job Shop Scheduling Techniques. *International Transactions in Operational Research*, 23:551–591, 2016. DOI: https://doi.org/10.1111/itor.12199.

J. C. Chen, K. H. Chen, J. J. Wu, and C. W. Chen. A study of the flexible job shop scheduling problem with parallel machines and reentrant process. *The International Journal of Advanced Manufacturing Technology*, 39(3-4):344–354, 2008. DOI: https://doi.org/10.1007/s00170-007-1227-1.

D. Cinar, Y. Topcu, and J. Oliveira. A Taxonomy for the Flexible Job Shop Scheduling Problem. 130:17–38, 2015. DOI: https://doi.org/10.1007/978-3-319-18567-5_2.

P. Clements and L. Bass. Relating Business Goals to Architecturally Significant Requirements for Software Systems. Technical Report May, 2010.

R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Dover Books on Computer Science. Dover Publications, 2012. ISBN 9780486151786.

G. Croes. A Method for Solving Traveling-Salesman Problems. *Operations Research*, 6(6): 791–812, 1958. DOI: https://doi.org/10.1287/opre.6.6.791.

S. Dauzère-Pérès and J. Paulli. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 70:281–306, 1997. DOI: https://doi.org/10.1023/A:1018930406487.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:551–591, 2002. DOI: https://doi.org/10.1109/4235.996017.

F. Della Croce, R. Tadei, and G. Volta. A genetic algorithm for the job shop problem. *Computers & Operations Research*, 22(1):15–24, jan 1995. DOI: https://doi.org/10.1016/0305-0548(93)E0015-L.

M. Dell'Amico and M. Trubian. Applying Tabu Search to the Job-Shop Scheduling Problem. *Annals of Operations Research*, 41:231–252, 1993. DOI: https://doi.org/10.1007/BF02023076.

D. M. Dhamdhere. *Operating Systems: A Concept-based Approach,2E.* McGraw-Hill Higher Education, 2006. ISBN 9780070611948.

P. D. D. Dominic, S. Kaliyamoorthy, and M. S. Kumar. Efficient dispatching rules for dynamic job shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 24(1-2):70–75, 2004. DOI: https://doi.org/10.1007/s00170-002-1534-5.

D. Dori. *Object-Process Methodology: A Holistic Systems Paradigm*, volume 276. Springer, 2002. ISBN 9783642562099. https://books.google.co.za/books?id=rmirCAAAQBAJ&dq=Object+process+methodology+dori&source=gbs_navlinks_s.

M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477, 1999. DOI: http://dx.doi.org/10.1109/CEC.1999.782657.

M. Englert, H. Roglin, and B. Vocking. Worst Case and Probabilistic Analysis of the 2-Opt Algorithm for the TSP. *Algorithmica*, 68:190–264, 2014. DOI: https://doi.org/10.1007/s00453-013-9801-4.

F. Esposito. *Innovations in Applied Artificial Intelligence: 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2005, Bari, Italy, June 22-24, 2005, Proceedings.* Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005. DOI: https://doi.org/10.1007/b137656.

N. F. O. Evbuomwan, S. Sivaloganathan, and A. Jebb. A survey of design philosophies, models, methods and systems. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 210(4):301–320, 1996. DOI: https://doi.org/10.1243/PIME_PROC_1996_210_123_02.

E. Falkenauer and S. Bouffouix. A genetic algorithm for job shop. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 824–829, April 1991.

G. Feldman, S. Hunter, and R. Pasupathy. Multi-objective simulation optimization on finite sets: optimal allocation via scalarization. In *In Proceedings of the 2015 Winter Simulation Conference*, pages 3610–3621. IEEE Press, 2015.

D. Fonseca and D. Navaresse. Artificial neural networks for job shop simulation. *Advanced Engineering Informatics*, 16(4):241–246, 2002. DOI: https://doi.org/10.1016/S1474-0346(03)00005-3.

M. Frantzén, A. H. Ng, and P. Moore. A simulation-based scheduling system for real-time optimization and decision making support. *Robotics and Computer-Integrated Manufacturing*, 27(4):696–705, 2011. DOI: https://doi.org/10.1016/j.rcim.2010.12.006.

E. Frazzon, M. Kück, and M. Freitag. Data-driven production control for complex and dynamic manufacturing systems. *CIRP Annals - Manufacturing Technology*, 67:515–518, 2018. DOI: https://doi.org/10.1016/j.cirp.2018.04.033.

S. French. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop*. Ellis Horwood, illustrate edition, 1982. DOI: https://doi.org/10.1002/net.3230130218.

K. Ganesh. *International and Interdisciplinary Studies in Green Computing*. Premier reference source. Information Science Reference, 2012. DOI: https://doi.org/10.4018/978-1-4666-2646-1.

M. Gen, J. Gao, and L. Lin. Multistage-Based Genetic Algorithm for Flexible Job-Shop Scheduling Problem. In M. Gen, D. Green, O. Katai, B. McKay, A. Namatame, R. A. Sarker, and B.-T. Zhang, editors, *Intelligent and Evolutionary Systems*, pages 183–196. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. DOI: https://doi.org/10.1007/978-3-540-95978-6_13.

F. Geyik and A. T. Dosdogru. Process plan and part routing optimization in a dynamic flexible job shop scheduling environment: an optimization via simulation approach. *Neural Computing and Applications*, 23(6):1631–1641, 2012. DOI: https://doi.org/10.1007/s00521-012-1119-7.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, jan 1986. DOI: https://doi.org/10.1016/0305-0548(86)90048-1.

F. Glover. Tabu Search - Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989. DOI: https://doi.org/10.1287/ijoc.1.3.190.

J. F. Gonçalves, J. J. d. M. Mendes, and M. G. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1):77–95, 2005. DOI: https://doi.org/10.1016/j.ejor.2004.03.012.

S. Goss, S. Aron, J. Deneubourg, and J. Pasteels. Self-organized shortcuts in the Argentine Ant . *The Science of Nature*, 76(12):579–581, 1989. DOI: https://doi.org/10.1007/BF00462870.

M. P. Groover. *Principles of Modern Manufacturing*. John Wiley and Sons, 5rd edition, 2013. ISBN 9781118474204.

J. Gu, M. Gu, C. Cao, and X. Gu. A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. *Computers & Operations Research*, 37(5):927–937, 2010. DOI: https://doi.org/10.1016/j.cor.2009.07.002.

S. Heragu and A. Alfa. Experimental analysis of simulated annealing based algorithms for the layout problem. *European Journal of Operational Research*, 57:190–202, 1992. DOI: https://doi.org/10.1016/0377-2217(92)90042-8.

J. W. Herrmann. *Handbook of Production Scheduling*. International Series in Operations Research & Management Science. Springer US, 2006. DOI: https://doi.org/10.1007/0-387-33117-4.

A. Hevner, A. March, J. Park, and S. Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28:75–105, 2004.

R. Hilliard, I. Malavolta, H. Muccini, and P. Pelliccione. On the composition and reuse of viewpoints across architecture frameworks. *Proceedings of the 2012 Joint Working Conference on Software Architecture and 6th European Conference on Software Architecture, WICSA/ECSA 2012*, pages 131–140, 2012. DOI: https://doi.org/10.1109/WICSA-ECSA.212.21.

K. L. Hoffman, M. Padberg, and G. Rinaldi. *Traveling Salesman Problem*, pages 1573–1578. Springer US, Boston, MA, 2013. DOI : http://dx.doi.org/10.1007/978-1-4419-1153-7_1068.

J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975. DOI: https://doi.org/10.1086/418447.

J. H. Holland and J. H. Miller. Artificial adaptive agents in economic theory. *American Economic Review*, 81(2):365–370, 1991. https://www.jstor.org/stable/2006886.

K.-L. Huang and C.-J. Liao. Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers & Operations Research*, 35(4):1030–1046, 2008. DOI: https://doi.org/10.1016/j.cor.2006.07.003.

S. Hunter and G. Feldman. Optimal sampling laws for bi-objective simulation optimization on finite sets. In *In Proceedings of the 2015 Winter Simulation Conference*, pages 3749–3757. IEEE Press, 2015.

J. Hurink, B. Jurisch, and M. Thole. Tabu search for the job-shop scheduling problem with multi-purpose machines. *OR Spektrum*, 15(4):205–215, 1994. DOI: https://doi.org/10.1007/BF01719451.

ISO. ISO 19450 Automation systems and integration: Object-process Methodology. https://www.iso.org/standard/62274.html, 2015. [Online, Accessed 8 August 2017].

M. T. Jensen. Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures. *Applied Soft Computing*, 1(1):35–52, 2001. DOI: https://doi.org/10.1016/S1568-4946(01)00005-9.

Z. Jiang, Z. Le, and M. E. Study on multi-objective flexible job-shop scheduling problem considering energy consumption. *Journal of Industrial Engineering and Management (JIEM)*, 7(3):589–604, 2014. DOI: https://doi.org/10.3926/jiem.1075.

D. Johnson and L. McGeoch. The Traveling Salesman Problem: A Case Study in Local Optimization. 1995. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.7639&rep=rep1&type=pdf.

K. Juvva. Real-Time Systems. https://users.ece.cmu.edu/~koopman/des_s99/real_time/, 1998. [Online, Accessed 3 May 2017].

B. Kádár, A. Pfeiffer, and L. Monostori. Discrete event simulation for supporting production planning and scheduling decisions in digital factories. *Proceedings of the 37th CIRP international seminar on manufacturing systems*, pages 444–448, 2004. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.507.8124&rep=rep1&type=pdf.

K. E. Kendall and J. E. Kendall. *Systems Analysis and Design*. Pearson Education Limited, 9 edition, 2013. ISBN 9780273788515.

J. Kennedy and R. Eberhart. Particle swarm optimisation. In *Proceedings of IEEE internantional conference on neural networks*, pages 1942–1948, 1995. DOI: https://doi.org/10.1109/ICNN.1995.488968.

C. Kessler and J. Sweitzer. *Outside-in Software Development: A Practical Approach to Building Successful Stakeholder-based Products*. IBM Press. Pearson Education, 2007. ISBN 9780132704373. https://books.google.co.za/books?id=_7fIuKB7geQC.

M. H. Kim and Y.-D. Kim. Simulation-based real-time scheduling in a flexible manufacturing system. *Journal of Manufacturing Systems*, 13(2):85–93, jan 1994. DOI: https://doi.org/10.1016/0278-6125(94)90024-8.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983. DOI: https://doi.org/10.1126/science.220.4598.671.

H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Real-Time Systems Series. Springer US, 2011. DOI: https://doi.org/10.1007/978-1-4419-8237-7.

E. Kosiba, J. Wright, and A. Cobbs. Discrete event sequencing as a Traveling Salesman Problem. *Computers in Industry*, 19:317–327, 1992. DOI: https://doi.org/10.1016/0166-3615(92)90069-Y.

P. Krüchten. The 4+1 view model of software architecture. *IEEE Software*, 12(6):42–50, 1995. DOI: https://doi.org/10.1109/52.469759.

J. Kuhpfahl. *Job Shop Scheduling with Consideration of Due Dates: Potentials of Local Search Based Solution Techniques*. Springer, illustrate edition, 2015. DOI: https://doi.org/10.1007/978-3-658-10292-0.

P. A. Laplante and S. J. Ovaska. *Real-Time Systems Design and Analysis: Tools for the Practitioner*. Wiley Desktop Editions. Wiley, 2011. DOI: https://doi.org/10.1002/9781118136607.

A. Law and W. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, 3 edition, 2000.

A. M. Law. *Simulation Modeling and Analysis*. McGraw-Hill series in industrial engineering and management science. McGraw-Hill, 2007. ISBN 9780071255196. https://books.google.co.za/books?id=OvY9PgAACAAJ.

Lazanda. NFC RFID-RC522 RF IC Card Sensor RFID Reader Module w/ S50 Card / Keychain for Arduino. https://www.lazada.com.my/products/nfc-rfid-rc522-rf-ic-card-sensor-rfid-reader-module-w-s50-card-keychain-for-arduino-i99920199-s12283320.html, 2018. [Online, Accessed 20 March 2018].

L. Lee, C. Chen, E. Chew, J. Li, N. Pujowidianto, and S. Zhang. A review of optimal computing budget allocation algorithms for simulation optimization problem. *International Journal of Operations Research*, 7:19–31, 2010.

D. Lei. Minimizing makespan for scheduling stochastic job shop with random breakdown. *Applied Mathematics and Computation*, 218(24):11851–11858, 2012. DOI: https://doi.org/10.1016/j.amc.2012.04.091.

J. Li, Q. Pan, and Y. Liang. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, 59(4):647–662, 2010a. DOI: https://doi.org/10.1016/j.cie.2010.07.014.

J. Li, Q. Pan, S. Xie, B. Jia, and Y. Wang. A Hybrid Particle Swarm Optimization and Tabu Search Algorithm for Flexible Job-Shop Scheduling Problem. *International Journal of Computer Theory and Engineering*, 2(2):189–194, 2010b. DOI: https://doi.org/10.7763/IJCTE.2010.V2.139.

J. Li, Q. Pan, P. N. Suganthan, and T. J. Chua. A Hybrid Tabu Search Algorithm With an Efficient Neighborhood Structure for the Flexible Job Shop Scheduling Problem. *The International Journal of Advanced Manufacturing Technology*, 52(5-8):683–697, 2011. DOI: https://doi.org/10.1007/s00170-010-2743-y.

T. Lin, S. Horng, T. Kao, Y. Chen, R. Run, R. Chen, J. Lai, and I. Kuo. An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, 37:2629–2636, 2010. DOI: https://doi.org/10.1016/j.eswa.2009.08.015.

L. Lindstrom and R. Jeffries. Extreme Programming and Agile Software Development Methodologies. *Information Systems Management*, 21(3):41–52, 2004. DOI: https://doi.org/10.1201/1078/44432.21.3.20040601/82476.7.

T. Liu, Y. Chen, and J. Chou. Solving Distributed and Flexible Job-Shop Scheduling Problems for a Real-World Fastener Manufacturer. *IEEE Access*, 2:1598–1606, 2014. DOI: https://doi.org/10.1109/ACCESS.2015.2388486.

Makerfabs. Arduino Pro Mini 328 - 5V/16MHz. https://www.makerfabs.com/index.php?route=product/product&product_id=193, 2016. [Online, Accessed 20 March 2018].

J. Martin. *Information Engineering: Design and construction*. Information Engineering. Prentice Hall, 1990. ISBN 9780134655017. https://books.google.co.za/books?id=XrpZAAAAYAAJ.

A. Mascis and D. Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, 2002. DOI: https://doi.org/10.1016/S0377-2217(01)00338-1.

W. S. McCulloch and W. Pitts. A Logical Calculus of the Idea Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. DOI: https://doi.org/10.1007/BF02478259.

D. Merkle, M. Middendorf, and H. Schmeck. Ant Colony Optimization for Resource-Constrained Project Scheduling. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 6(4):333–346, 2002. DOI: https://doi.org/10.1109/TEVC.2002.802450.

J. Miltenburg. *Manufacturing Strategy: How to Formulate and Implement a Winning Plan, Second Edition*. Taylor & Francis, 2005. ISBN 9781563273179. https://books.google.co.za/books?id=k6f5tti-_cIC.

R. K. Mitchell, D. J. Wood, and B. Agle. Toward a Theory of Stakeholder Identification and Salience : Defining the Principle of Who and What Really Counts. *Academy of Management Review*, 22(4):853–886, 1997. DOI: https://doi.org/10.5465/AMR.1997.9711022105.

M. Mitsuishi, K. Ueda, and F. Kimura. *Manufacturing Systems and Technologies for the New Frontier: The 41st CIRP Conference on Manufacturing Systems May 26–28, 2008, Tokyo, Japan*. Springer London, 2008. DOI: https://doi.org/10.1007/978-1-84800-267-8.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, nov 1997. DOI: https://doi.org/10.1016/S0305-0548(97)00031-2.

M. Montazeri and L. Van Wassenhove. Analysis of scheduling rules for an FMS. *International Journal of Production Research*, 28:785–802, 1990. https://doi.org/10.1080/00207549008942754.

D. Montgomery and G. Runger. *Applied Statistics and Probability for Engineers*. Wiley, 2013. ISBN 9781118802267. https://books.google.co.za/books/about/Applied_Statistics_and_Probability_for_E.html?id=eHpbAgAAQBAJ&redir_esc=y.

E. Moradi, S. Fatemi Ghomi, and M. Zandieh. Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem. *Expert Systems with Applications*, 38:7169–7178, 2011. https://doi.org/10.1016/j.eswa.2010.12.043.

R. Moritz, E. Reich, M. Schwarz, M. Bernt, and M. Middendorf. Refined Ranking Relations for Selection of Solutions in Multi-objective Metaheuristics. *European Journal of Operational Research*, 243:454–464, 2015. DOI: https://doi.org/10.1016/j.ejor.2014.10.044.

Murata. Lora module FAQ. https://www.murata.com/support/faqs/products/lpwa/lora/hardware/0008, 2018. [Online, Accessed 10 February 2018].

P. R. Murthy. *Production And Operations Management*. New Age International (P) Limited Publishers, 2005. ISBN 9788122415582. URL https://books.google.co.za/books?id=KQToFyMfZdYC.

N. Nasr and E. A. Elsayed. Job shop scheduling with alternative machines. *International Journal of Production Research*, 28(9):1595, 1990. ISSN 00207543. https://doi.org/10.1080/00207549008942818.

P. Ngatchou, A. Zarei, and A. El-Sharkawi. Pareto multi objective optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pages 84–91, Nov 2005. DOI: https://doi.org/10.1109/ISAP.2005.1599245.

L. Nurkhalida and B. Santosa. Pendekatan cross entropy-genetic algorithm pada permasalahan multi objective job shop scheduling, 2012.

NXP. MFRC522 data sheet. https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf, 2016. [Online, Accessed 1 December 2017].

F. Ogbu and D. Smith. The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem. *Computers & Operations Research*, 17(3):243–253, jan 1990. DOI: https://doi.org/10.1016/0305-0548(90)90001-N.

J. Pan and J. Chen. Mixed binary integer programming formulations for the reentrant job shop scheduling problem. *Computers & Operations Research*, 32(5):1197–1212, 2005. DOI: https://doi.org/10.1016/j.cor.2003.10.004.

C. T. Papadopoulos, M. E. J. O'Kelly, M. J. Vidalis, and D. Spinellis. *Analysis and Design of Discrete Part Production Lines*. Springer Optimization and Its Applications. Springer New York, 2009. DOI: https://doi.org/10.1007/978-0-387-89494-2.

P. M. Pardalos. *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*. Nonconvex Optimization and Its Applications. Springer US, 2013. ISBN 9781475731453. DOI: https://doi.org/10.1007/978-1-4757-3145-3.

F. Pezzella and E. Merelli. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, 120(2):297–310, 2000. DOI: https://doi.org/10.1016/S0377-2217(99)00158-7.

F. Pezzella, G. Morganti, and G. Ciaschetti. A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers & Operations Research*, 35(10):3202–3212, 2008. DOI: http://dx.doi.org/10.1016/j.cor.2007.02.014.

M. L. Pinedo. *Scheduling*. Springer, 5 edition, 2016. DOI: https://doi.org/10.1007/978-3-319-26580-3.

S. Ponnambalam, V. Ramkumar, and N. Jawahar. A multiobjective genetic algorithm for job shop scheduling. *Production Planning & Control*, 12(8):764–774, 2001. DOI: https://doi.org/10.1080/09537280110040424.

E. A. Puente and L. Nemes. *Information Control Problems in Manufacturing Technology 1989: Selected papers from the 6th IFAC/IFIP/IFORS/IMACS Symposium, Madrid, Spain, 26-29 September 1989*. IFAC Symposia Series. Elsevier Science, 2014. https://doi.org/10.1016/C2009-0-01279-3.

A. Purhonen, E. Niemelä, and M. Matinlassi. Viewpoints of DSP software and service architectures. *Journal of Systems and Software*, 69(1-2):57–73, 2004. DOI: https://doi.org/10.1016/S0164-1212(03)00050-5.

M. Rabiee, M. Zandieh, and P. Ramezani. Bi-objective partial flexible job shop scheduling problem: Nsga-ii, nrga, moga and paes approaches. *International Journal of Production Research*, 50(24):7327–7342, 2012. DOI: https://doi.org/10.1080/00207543.2011.648280.

V. Ramesh. *Principles of Operating Systems*. Laxmi Publications Pvt Limited, 2010. ISBN 9789380386171. https://books.google.co.za/books?id=YDxAiNDQVlUC.

R. L. Rardin. *Optimization in Operations Research*. Prentice Hall, 1998. ISBN 9780023984150. https://books.google.co.za/books?id=9WVRAAAAMAAJ.

Raspberry Pi Foundation. Raspberry Pi - Teach, Learn, and Make with Raspberry Pi. https://www.raspberrypi.org/faqs/, 2014. [Online, Accessed 20 November 2017].

M. Rebai, I. Kacem, and K. H. Adjallah. Earliness-tardiness minimization on a single machine to schedule preventive maintenance tasks: Metaheuristic and exact methods. *Journal of Intelligent Manufacturing*, 23(4):1207–1224, 2012. DOI: https://doi.org/10.1007/s10845-010-0425-0.

G. Z. Rey, A. Bekrar, D. Trentesaux, and B. H. Zhou. Solving the flexible job-shop just-in-time scheduling problem with quadratic earliness and tardiness costs. *International Journal of Advanced Manufacturing Technology*, 81(9-12):1871–1891, 2015. DOI: https://doi.org/10.1007/s00170-015-7347-0.

RF Wireless World. Advantages and Disadvantages of Lora or LoRaWAN. http://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Lora-or-LoRaWAN.html, 2017. [Online, Accessed 10 February 2018].

M. Rohaninejad, A. Khierkhah, P. Fattahi, and B. Vahedi-Nouri. A Hybrid Multi-Objective Genetic Algorithm Based on the ELECTRE Method for a Capacitated Flexible Job Shop Scheduling Problem. *International Journal for Advanced Manufacturing Technologies*, 77:51–66, 2015. DOI: https://doi.org/10.1007/s00170-014-6415-1.

O. Rose, T. Zhang, and S. Xie. Flexible Job-shop Scheduling with Overlapping Machine Sets. In *Proceedings of the 2015 Winter Simulation Conference*, pages 2307–2316, 2015. ISBN 9781467397438. DOI: https://doi.org/10.1109/WSC.2015.7408342.

V. Roshanaei, B. Naderi, F. Jolai, and M. Khalili. A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. *Future Generation Computer Systems*, 25(6):654–661, 2009. DOI: https://doi.org/10.1016/j.future.2009.01.004.

V. Roshanaei, A. Azab, and H. ElMaraghy. Mathematical modelling and a meta-heuristic for flexible job shop scheduling. *International Journal of Production Research*, 51(20): 6247–6274, 2013. URL http://www.tandfonline.com/doi/abs/10.1080/00207543.2013.827806.

N. Rozanski and E. Woods. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 1 edition, 2005. ISBN 0321112296. https://books.google.co.za/books?id=dnBlz-FLEnIC&dq=isbn+0321112296&source=gbs_navlinks_s.

N. Rozanski and E. Woods. *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 2011. ISBN 9780321718334. https://books.google.co.za/books?id=ka4QO9kXQFUC.

R. Rubinstein and D. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer, 2004.

R. Ruiz, E. Vallada, and C. Fernandez-Martinez. Scheduling in flowshops with no-idle machines. In *Computational Intelligence in Flow Shop and Job Shop Scheduling*, pages 21–51. Springer Berlin Heidelberg, 2009. DOI: https://doi.org/10.1007/978-3-642-02836-6_2.

I. Sabuncuoglu and M. Bayız. Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, 126(3):567–586, 2000. DOI: https://doi.org/10.1016/S0377-2217(99)00311-2.

N. Sadeh. *Look-ahead techniques for micro-opportunistic job shop scheduling*. PhD thesis, Carnegie Mellon University, Pennsylvania, 1991.

M. Saidi-Mehrabad and P. Fattahi. Flexible job shop scheduling with tabu search algorithms. *The International Journal of Advanced Manufacturing Technology*, 32:563–570, 2007. DOI: https://doi.org/10.1007/s00170-005-0375-4.

M. Sakawa and R. Kubota. Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms. *European Journal of Operational Research*, 120(2):393–407, 2000. DOI: https://doi.org/10.1016/S0377-2217(99)00094-6.

B. Santosa, M. Budiman, and S. Wiratno. A cross entropy-genetic algorithm for m-machines no-wait job-shop scheduling problem. *Journal of Intelligent Learning Systems and Applications*, 3:171–180, 2011. DOI: https://doi.org/10.4236/jilsa.2011.33018.

T. Satake, K. Morikawa, K. Takahashi, and N. Nakamura. Simulated annealing approach for minimizing the makespan of the general job-shop. *International Journal of Production Economics*, 60:515–522, 1999. DOI: https://doi.org/10.1016/S0925-5273(98)00171-6.

T. J. Schriber, D. T. Brunner, and J. S. Smith. Inside discrete-event simulation software: How it works and why it matters. In *Proceedings of the 2015 Winter Simulation Conference*, pages 1–15, 2015. DOI: https://doi.org/10.1109/WSC.2015.7408149.

Scott Tattersall. How to build custom IoT hardware with Arduino. https://opensource.com/article/17/12/how-build-custom-iot-hardware-arduino, 2018. [Online, Accessed 10 February 2018].

C. Scrich, V. Armentano, and M. Laguna. Tardiness minimization in a flexible job shop: A tabu search approach. *Journal of Intelligent Manufacturing*, 15(1):103–115, 2004. DOI: https://doi.org/10.1023/B:JIMS.0000010078.30713.e9.

Seeed Studio. RFM98 Ultra-long Range Transceiver Module/LoRa Module/support 433M frequency. https://www.seeedstudio.com/RFM98-Ultra-long-Range-Transceiver-Module{%}2FLoRa-Module{%}2Fsupport-433M-frequency-p-2806.html, 2017. [Online, Accessed 20 March 2018].

V. Sels, N. Gheysena, and M. Vanhouckeabc. A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions. *International Journal of Production Research*, 50(15):4255–4270, 2012. DOI: https://doi.org/10.1080/00207543.2011.611539.

Semtech. Lora transceivers. https://www.semtech.com/products/wireless-rf/lora-transceivers, 2018. [Online, Accessed 10 February 2018].

D. Sha and H. Lin. A Multi-Objective PSO for Job-Shop Scheduling Problems. *Expert Systems with Applications*, 37:1065–1070, 2010. DOI: https://doi.org/10.1016/j.eswa.2009.06.041.

S. Snyman and J. Bekker. Real-time scheduling in a sensorised factory using cloud-based simulation with mobile device access. *South African Journal of Industrial Engineering*, 28(4), 2017. DOI: http://dx.doi.org/10.7166/28-4-1860.

S. Snyman and J. Bekker. A real-time scheduling system in a sensorised job shop. In *In Proceedings of the International Conference on Competitive Manufacturing*, 2019a.

S. Snyman and J. Bekker. Comparing the performance of different metaheuristics when solving a stochastic bi-objective job shop scheduling problem. In *In Proceedings of the 48th ORSSA Annual Conference*, 2019b.

S. Snyman, J. Bekker, and J. Botha. A real-time scheduling system for a sensorised job shop using cloud-based simulation with mobile device access. In *In Proceedings of the 29th SAIIE Annual Conference*, 2018.

J. Sridhar and C. Rajendran. Scheduling in flowshop and cellular manufacturing systems with multiple objectives— a genetic algorithmic approach. *Production Planning & Control*, 7(4):764–774, 1996. DOI: https://doi.org/10.1080/09537289608930365.

D. H. Stamatis. *Essential Statistical Concepts for the Quality Professional*. CRC Press, 2012. ISBN 9781439894576. https://books.google.co.za/books?id=SzQPsKtbK_QC&dq=isbn+9781439894576&source=gbs_navlinks_s.

J. A. Stankovic, M. Spuri, K. Ramamritham, and G. Buttazzo. *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*. The Springer International Series in Engineering and Computer Science. Springer US, 2012. DOI: https://doi.org/10.1007/978-1-4615-5535-3.

P. Surekha and S. Sumathi. Solving Fuzzy based Job Shop Scheduling Problems using Ga and Aco. *Journal of Emerging Trends in Computing and Information Sciences*, 1(2): 95–102, 2010. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.302.6981&rep=rep1&type=pdf.

R. K. Suresh and K. M. Mohanasundaram. Pareto archived simulated annealing for job shop scheduling with multiple objectives. *International Journal of Advanced Manufacturing Technology*, 29(1-2):184–196, 2006. DOI: https://doi.org/10.1007/s00170-004-2492-x.

H. Suwa and H. Sandoh. *Online Scheduling in Manufacturing: A Cumulative Delay Approach*. SpringerLink : Bücher. Springer London, 2012. DOI: https://doi.org/10.1007/978-1-4471-4561-5.

C. Tarantilis and C. Kiranoudis. A list-based threshold accepting method for job shop scheduling problems. *International Journal for Production Economics*, 77:159–171, 2002. DOI: https://doi.org/10.1016/S0925-5273(01)00231-6.

Teach Engineering. Engineering Design Process. https://www.teachengineering.org/k12engineering/designprocess#, 2016. [Online, Accessed 29 October 2018].

The Pi Hut. Raspberry Pi 2 - Model B (V1.2). https://thepihut.com/products/raspberry-pi-2-model-b?variant=18198528708, 2017. [Online, Accessed 20 March 2018].

R. M. Thiesing and C. D. Pegden. Simio Applications in Scheduling. In *Proceedings of the 2016 Winter Simulation Conference*, pages 3620–3629, 2016. https://www.informs-sim.org/wsc16papers/345.pdf.

D. Trentesaux, C. Pach, A. Bekrar, Y. Sallez, T. Berger, T. Bonte, P. Leitão, and J. Barbosa. Benchmarking flexible job-shop scheduling and control systems. *Control Engineering Practice*, 21(9):1204–1225, 2013. DOI: https://doi.org/10.1016/j.conengprac.2013.05.004.

J. M. Usher. Negotiation-based routing in job shops via collaborative agents. *Journal of Intelligent Manufacturing*, 14(5):485–499, 2003. DOI: https://doi.org/10.1023/A:1025705426184.

P. Van Laarhoven, E. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):113–125, 1992. https://pdfs.semanticscholar.org/60b3/ffcdcfd3cacf050ba096b4201adc1ce296ba.pdf.

G. Vilcot and J. Billaut. A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. *European Journal of Operational Research*, 190(2):398–411, 2008. DOI: https://doi.org/10.1016/j.ejor.2007.06.039.

S. Voß, S. Martello, I. H. Osman, and C. Roucairol. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Springer Science & Business Media, 2012. ISBN 9781461557753. https://books.google.co.za/books?id=6CzoBwAAQBAJ&dq.

T. Vredeveld. Stochastic online scheduling. *Computer Science - Research and Development*, 27(3):181–187, 2012. DOI: https://doi.org/10.1007/s00450-011-0153-5.

D. Walker, E. Shanks, D. Montoya, C. Weiman, E. Pérez, and L. DePagter. Towards a Simulation Based Methodology for Scheduling Patient and Providers At Outpatient Clinics. In *Proceedings of the 2015 Winter Simulation Conference*, pages 1515–1524, 2015. DOI: https://doi.org/10.1109/WSC.2015.7408273.

L. Wang and D. Zheng. A Modified Genetic Algorithm for Job Shop Scheduling. *The International Journal of Advanced Manufacturing Technology*, 20(1):72–76, 2002. DOI: https://doi.org/10.1007/s001700200126.

L. Wang, A. H. C. Ng, and K. Deb. *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*. Springer London, 2011. DOI: https://doi.org/10.1007/978-0-85729-652-8.

S. Wang, B. Zhou, and L. Xi. A filtered-beam-search-based heuristic algorithm for flexible job-shop scheduling problem. *International Journal of Production Research*, 46(11):3027–3058, 2008. DOI: https://doi.org/10.1080/00207540600988105.

Y. M. Wang, H. L. Yin, and K. D. Qin. A novel genetic algorithm for flexible job shop scheduling problems with machine disruptions. *International Journal of Advanced Manufacturing Technology*, 68(5-8):1317–1326, 2013. DOI: https://doi.org/10.1007/s00170-013-4923-z.

J. L. Ward. *Project Management Terms: A Working Glossary*. ESI International, 2000. ISBN 1890367257. https://books.google.co.za/books?id=GFPBkOlUgKwC&dq=isbn+1890367257&hl=en&sa=X&ved=0ahUKEwi6iOyvktvcAhUJbcAKHd0LDWAQ6AEIKDAA.

E. Weber, A. Tiefenbacher, and N. Gronau. Need for standardization and systematization of test data for job-shop scheduling. *Data*, 4, 2019. DOI: https://doi.org/10.3390/data4010032.

G. R. Weckman, C. V. Ganduri, and D. A. Koonce. A neural network job-shop scheduler. *Journal of Intelligent Manufacturing*, 19(2):191–201, 2008. DOI: https://doi.org/10.1007/s10845-008-0073-9.

D. Whitley, T. Starkweather, and D. Fuquay. Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator. In *In Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 133–140. Morgan Kaufmann Publishers Inc., 1989.

W. L. Winston and J. B. Goldberg. *Operations Research: Applications and Algorithms*. Thomson Brooks/Cole, 2004. ISBN 9780534423582. https://books.google.co.za/books?id=tg5DAQAAIAAJ.

J. D. Wisner. *Operations Management: A Supply Chain Process Approach*. SAGE Publications, 2016. ISBN 9781483383057. https://books.google.co.za/books?id=qH-zDAAAQBAJ.

E. Woods. Experiences Using Viewpoints for Information Systems Architecture: An Industrial Experience Report. *Software Architecture*, pages 182–193, 2004. DOI: http://dx.doi.org/10.1007/978-3-540-24769-2_13.

F. Xhafa and A. Abraham. *Metaheuristics for Scheduling in Industrial and Manufacturing Applications*. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2008. ISBN 9783540789840. https://books.google.co.za/books?id=sn4ksQOnE5MC.

X. Xu, Y. Zhao, H. Li, Z. Zhou, and Y. Liu. Stochastic Customer Order Scheduling using Simulation-Based Genetic Algorithm. In *Proceedings of the 2015 Winter Simulation Conference*, number 1, pages 425–436, 2015. ISBN 9781467397438. DOI: https://doi.org/10.1109/WSC.2015.7408343.

M. Yazdani, M. Gholami, M. Zandieh, and M. Mousakhani. A simulated annealing algorithm for flexible job-shop scheduling problem. *Journal of Applied Sciences*, 9(4): 662–670, 2009. DOI: https://doi.org/10.3923/jas.2009.662.670.

M. Yazdani, M. Amiri, and M. Zandieh. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, 37(1):678–687, 2010. DOI: https://doi.org/10.1016/j.eswa.2009.06.007.

G. Yen and Z. He. Performance Metric Ensemble for Multiobjective Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 18:131–144, 2014. DOI: https://doi.org/10.1109/TEVC.2013.2240687.

M. Yoon. New multi-objective ranking and selection procedures for discrete stochastic simulation problems, 2018.

Y. Yuan and H. Xu. Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science and Engineering*, 12(1):336–353, Jan 2015. ISSN 1545-5955. DOI: https://doi.org/10.1109/TASE.2013.2274517.

L. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, jun 1965. DOI: https://doi.org/10.1016/S0019-9958(65)90241-X.

C. Y. Zhang, P. Li, Y. Rao, and Z. Guan. A very fast TS/SA algorithm for the job shop scheduling problem. *Computers & Operations Research*, 35:282–294, 2008. DOI: https://doi.org/10.1016/j.cor.2006.02.024.

G. Zhang, X. Shao, P. Li, and L. Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56(4):1309–1318, 2009. ISSN 03608352. DOI: https://doi.org/10.1016/j.cie.2008.07.021.

J. Zhang, J. Yang, and Y. Zhou. Robust Scheduling for Multi-Objective Job-Shop Problems with Flexible Workdays. *Engineering Optimization*, 28:1973–1989, 2016. DOI: https://doi.org/10.1080/0305215X.2016.1145216.

R. Zhang and C. Wu. A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing*, 10(1):79–89, 2010. DOI: https://doi.org/10.1016/j.asoc.2009.06.008.

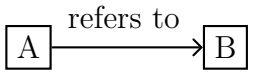
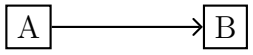
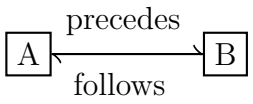
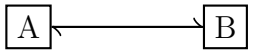
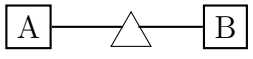
G. Zobolas, C. Tarantilis, and G. Ioannou. A hybrid evolutionary algorithm for the job shop scheduling problem. *Journal of the Operational Research Society*, 60:221–235, 2009. DOI: https://doi.org/10.1057/palgrave.jors.2602534.

Appendix A

OPM symbols and links

The set of symbols used by OPM to describe the relationship between the different building blocks, is listed in Table A.1.

Table A.1: Set of symbols in OPM (Dori, 2002).

Link Name	Object Process Diagram Symbol	Object Process Language	Description
Tagged		A refers to B.	Relation from source object to destination object.
(Null)		A relates to B.	Relation from source object to destination object with no tag.
Bi-directional Tagged		A precedes B. B follows A.	Relation between two objects.
(Null) Bi-directional		A and to B are related.	Relation between two objects with no tag.
Aggregation		A consists of B.	Relates a whole to its parts.
Exhibition		A exhibits B.	Relates an exhibitor to its attributes.
Generalisation		B is an A.	Relates a general thing to its specialisations.
Instantiation		B is an instance of A.	Relates a class of things to its instances.

Appendix B

Development validation and testing

This chapter describes the validation and testing of the scheduling process, as well as the operations of the sensors. For the validation of the scheduling process, test scenarios are defined. The expected solution when applying each dispatching rule to the test scenarios will be determined and then compared to the schedule generated by the simulation scheduler. To validate the sensors, they will be tested to determine whether they can successfully and accurately change the operation status of a chosen job on the information system.

B.1 Validation of scheduling process

This section addresses the validation of the scheduling process, and starts off by defining four test scenarios that will determine if the dispatching rules generate the correct schedules. The section will also discuss the validation of the dispatching rules, after which the outcomes of each dispatching rule in relation to the KPIs will be discussed, and the best dispatching rule for each KPI will be identified.

B.1.1 Defining test scenarios

Four test scenarios will be created to validate the scheduling process as well as to test the robustness of the process. The four scenarios are as follows:

- **Scenario one:** Five jobs are entered into the system, where four jobs have similar expected processing times. There is, however, one job which is a clear outlier because the processing times of its operations are considerably longer than those of the other jobs. (A small number of jobs are selected in order to be able to analyse the results manually.)
- **Scenario two:** Five jobs are entered into the system, each having similar, but not exactly equal, expected processing times.
- **Scenario three:** Five jobs are entered into the system, each with exactly the same expected processing time.
- **Scenario four:** Here 50 jobs are entered into the system, having similar expected processing times. This scenario is used to test the robustness of the scheduler, i.e. whether it can handle a relatively large number of jobs. Due to the extreme nature

Table B.1: Data entered into `tblOperators`

Operator_ID	OperatorName	OpStatus_IDFK
1	Jan	1
2	Hector	1
3	John	1
4	Robert	1
5	Michael	1
6	William	1
7	Richard	1
8	David	1

Table B.2: Data entered into `tblMachines`

Machine_ID	MachineName	MachineStatus_IDFK
1	Turning1	1
2	Turning2	1
3	Milling1	1
4	Milling2	1
5	Grinding1	1
6	Grinding1	1
7	Induction Hardening	1
8	Gear Milling	1

of this scenario, the input data will not be provided; however, the results of the tests will be compared and discussed.

For the first three test scenarios, a small number of jobs are selected in order to be able to manually analyse the results. Before the test scenarios could be set up, all the data tables in the information system needed to be cleared of all data. Thereafter, the web pages were used to log the test scenarios. For each test scenario, eight operators were entered into `tblOperators`, as illustrated in Table B.1, each with an `OpStatus_IDFK` of "1" which indicates that they are in the *Working* state.

Next, `tblMachines` needed to be populated. Due to the predefined machines that were included in the simulation model, the same machines were also included in the data table, as illustrated in Table B.2, each with a `MachineStatus_IDFK` of "1" which indicates that they are in the *Operational* state. This data also stayed constant throughout all four test scenarios.

After the resources had been entered into their respective tables, `tblJobs` and `tblOperations` could be populated for the different test scenarios. After each scenario, `tblJobs` and `tblOperations` had to be cleared and populated with the data of the next scenario. The operations per job ware decided to be between one and the total number of different types of machines. Next, the data for each test scenario will be provided.

B.1.1.1 Test scenario 1 data

The jobs that were entered into `tblJobs` for Test scenario 1 are illustrated in Table B.3.

Table B.3: Data entered into `tblJobs` for Test scenario 1

Job _ID	Job Name	Job Date	Job Description	Job Priority	Job Quantity	Job DueDate
1	J1	2018/06/04 11:40:51	Ball joints	5	7	2018/06/30
2	J2	2018/06/04 11:41:34	Brackets	3	1	2018/06/15
3	J3	2018/06/04 11:47:56	Fittings	1	5	2018/06/13
4	J4	2018/06/04 11:48:50	Housings	3	2	2018/06/30
5	J5	2018/06/04 11:53:07	Rollers	2	8	2018/06/26

Each job was then assigned operations, and the data populated in `tblOperations` are illustrated in Table B.4. This concludes the definition and population of Test scenario 1 into the information system. The next step is to run the simulation scheduler which will generate schedules according to each dispatching rule. In the next section, the generated schedules for this scenario will be displayed and discussed.

Table B.4: Data entered into `tblOperations` for Test scenario 1

Operation _ID	Job _IDFK	Operation Number	Machine _IDFK	Expected Processing Time	Earliest Start Time	Latest Start Time	Status _IDFK
1	1	1	1	01:33:50	2018/06/12 12:00	2018/06/25 12:00	1
2	1	2	3	00:15:30	2018/06/13 12:00	2018/06/26 12:00	1
3	1	3	5	01:53:40	2018/06/13 15:00	2018/06/27 12:00	1
4	1	4	7	01:43:48	2018/06/14 13:00	2018/06/28 12:00	1
5	1	5	8	01:58:19	2018/06/15 12:00	2018/06/29 12:00	1
6	2	1	3	00:51:32	2018/06/12 12:00	2018/06/12 16:00	1
7	2	2	5	01:16:41	2018/06/13 12:00	2018/06/13 16:00	1
8	2	3	7	01:57:17	2018/06/14 12:00	2018/06/14 16:00	1
9	3	1	5	00:36:44	2018/06/12 14:00	2018/06/13 10:00	1
10	3	2	8	00:40:15	2018/06/12 16:00	2018/06/13 13:40	1
11	4	1	7	01:23:16	2018/06/12 16:00	2018/06/15 15:00	1
12	4	2	8	00:12:52	2018/06/14 14:00	2018/06/16 10:00	1
13	4	3	2	01:10:04	2018/06/15 08:00	2018/06/16 16:00	1
14	4	4	4	00:41:06	2018/06/16 11:00	2018/06/17 12:00	1
15	4	5	6	00:39:48	2018/06/17 09:00	2018/06/20 16:00	1
16	5	1	4	00:30:00	2018/06/12 12:00	2018/06/26 13:00	1

B.1.1.2 Test scenario 2 data

The jobs that were entered into `tblJobs` for Test scenario 2 are illustrated in Table B.5.

Table B.5: Data entered into `tblJobs` for Test scenario 2

Job_ID	Job Name	Job Date	Job Description	Job Priority	Job Quantity	Job DueDate
1	J1	2018/05/21 08:00	Ball joints	5	7	2018/06/20
2	J2	2018/05/21 08:10	Brackets	3	1	2018/06/20
3	J3	2018/05/21 08:20	Fittings	1	5	2018/06/20
4	J4	2018/05/21 08:30	Housings	3	2	2018/06/19
5	J5	2018/05/21 08:40	Rollers	2	8	2018/06/22

Each job was then assigned operations, and the data populated in `tblOperations` are illustrated in Table B.6. This concludes the definition and population of Test scenario 2 into the information system. The next step is to run the simulation scheduler which will generate schedules according to each dispatching rule. In the next section, the generated schedules for this scenario will be displayed and discussed.

Table B.6: Data entered into `tblOperations` for Test scenario 2

Operation _ID	Job _IDFK	Operation Number	Machine _IDFK	Expected Processing Time	Earliest Start Time	Latest Start Time	Status _IDFK
1	1	1	1	0:13:19	2018/06/05 13:00	2018/06/19 14:30	1
2	1	2	3	0:15:59	2018/06/05 14:33	2018/06/19 16:00	1
3	1	3	5	0:19:59	2018/06/05 16:25	2018/06/20 09:00	1
4	1	4	7	0:10:39	2018/06/06 08:44	2018/06/20 13:00	1
5	1	5	8	0:06:39	2018/06/06 09:59	2018/06/20 16:00	1
6	2	1	3	2:50:54	2018/06/05 13:00	2018/06/19 16:13	1
7	2	2	5	3:53:03	2018/06/05 15:50	2018/06/20 12:30	1
8	2	3	7	1:02:09	2018/06/06 09:43	2018/06/20 16:00	1
9	3	1	5	0:32:38	2018/06/05 13:00	2018/06/19 12:00	1
10	3	2	8	1:00:35	2018/06/05 15:40	2018/06/19 16:13	1
11	4	1	7	0:46:37	2018/06/05 13:00	2018/06/19 09:13	1
12	4	2	8	1:14:34	2018/06/05 14:33	2018/06/19 10:47	1
13	4	3	2	0:18:39	2018/06/06 08:02	2018/06/19 13:16	1
14	4	4	4	0:55:56	2018/06/06 08:39	2018/06/19 13:53	1
15	4	5	6	0:37:17	2018/06/06 10:31	2018/06/19 16:00	1
16	5	1	4	0:58:16	2018/06/05 13:00	2018/06/21 16:13	1

B.1.1.3 Test scenario 3 data

The jobs that were entered into `tblJobs` for Test scenario 3 are illustrated in Table B.7.

Table B.7: Data entered into `tblJobs` for Test scenario 3

Job_ID	Job Name	Job Date	Job Description	Job Priority	Job Quantity	Job DueDate
1	J1	2018/05/21 08:00	Ball joints	5	7	2018/06/20
2	J2	2018/05/21 08:10	Brackets	3	1	2018/06/20
3	J3	2018/05/21 08:20	Fittings	1	5	2018/06/20
4	J4	2018/05/21 08:30	Housings	3	2	2018/06/19
5	J5	2018/05/21 08:40	Rollers	2	8	2018/06/22

Each job was then assigned operations, and the data populated in `tblOperations` are illustrated in Table B.8. The expected processing times of the operations will not be altered in the scheduler for this scenario, which will ensure that the processing times are the same. This concludes the definition and population of Test scenario 3 into the information system. The next step is to run the simulation scheduler which will generate schedules according to each dispatching rule. In the next section, the generated schedules for this scenario will be displayed and discussed.

Table B.8: Data entered into `tblOperations` for Test scenario 3

Operation _ID	Job _IDFK	Operation Number	Machine _IDFK	Expected Processing Time	Earliest Start Time	Latest Start Time	Status _IDFK
1	1	1	1	0:13:19	2018/06/05 13:00	2018/06/19 14:30	1
2	1	2	3	0:15:59	2018/06/05 14:33	2018/06/19 16:00	1
3	1	3	5	0:19:59	2018/06/05 16:25	2018/06/20 09:00	1
4	1	4	7	0:10:39	2018/06/06 08:44	2018/06/20 13:00	1
5	1	5	8	0:06:39	2018/06/06 09:59	2018/06/20 16:00	1
6	2	1	3	2:50:54	2018/06/05 13:00	2018/06/19 16:13	1
7	2	2	5	3:53:03	2018/06/05 15:50	2018/06/20 12:30	1
8	2	3	7	1:02:09	2018/06/06 09:43	2018/06/20 16:00	1
9	3	1	5	0:32:38	2018/06/05 13:00	2018/06/19 12:00	1
10	3	2	8	1:00:35	2018/06/05 15:40	2018/06/19 16:13	1
11	4	1	7	0:46:37	2018/06/05 13:00	2018/06/19 09:13	1
12	4	2	8	1:14:34	2018/06/05 14:33	2018/06/19 10:47	1
13	4	3	2	0:18:39	2018/06/06 08:02	2018/06/19 13:16	1
14	4	4	4	0:55:56	2018/06/06 08:39	2018/06/19 13:53	1
15	4	5	6	0:37:17	2018/06/06 10:31	2018/06/19 16:00	1
16	5	1	4	0:58:16	2018/06/05 13:00	2018/06/21 16:13	1

B.1.2 Dispatching rule validation

This section describes the validation process for the dispatching rules that were used by the simulation scheduler. The scheduler will be used to generate a schedule for each dispatching rule, using the data provided in each test scenario. There will therefore be 24 schedules generated, which will then be discussed with reference to the correctness of the sequence of operations allocated to machines.

B.1.2.1 Shortest processing time rule

The discussion of the schedules generated from the *shortest processing time* dispatching rule for each test scenario will now follow.

When the scheduler was run with the data defined in the test scenarios, the *shortest processing time* dispatching rule generated a sequence in which jobs must be processed for the different scenarios as shown in Tables B.9, B.10 and B.11. The processing times listed in these tables are the results of the triangular distribution that was applied to the processing times of the operations. The stochastic processing times of the operations of each job are then summed to calculate the total processing time for each job. The output of the rule proved to be correct, because the job with the shortest processing time was first in each sequence, while the job with the longest processing time was last. With these sequences, the scheduler generated a schedule of the shop for all four test scenarios as illustrated in Figures B.2, B.3, B.4, and B.5, where the different machines are denoted by M1 – M8. The total length of the schedule is also provided in *"Days:Hours:Minutes:Seconds"*.

Table B.9: Shortest processing time sequence for Test scenario 1

Job_ID	Processing time (hh:mm:ss)
5	3:52:48
2	3:55:27
3	6:11:19
4	8:30:21
1	49:59:48

Table B.10: Shortest processing time sequence for Test scenario 2

Job_ID	Processing time (hh:mm:ss)
5	7:32:09
3	7:34:08
1	7:37:33
2	7:38:33
4	8:05:04

Table B.11: Shortest processing time sequence for Test scenario 3

Job_ID	Processing time (hh:mm:ss)
1	7:46:05
3	7:46:05
2	7:46:06
4	7:46:06
5	7:46:08

For purposes of explanation, the schedule generated for Test scenario 1 will be manually analysed to determine if it is correct. Figure B.1 was manually created, to perform the evaluation of the sequence. The operations of the jobs were manually allocated to the machines, keeping in mind that the utilisation on identical machines should be similar. Operations can therefore be shifted from one machine to an identical machine, to balance the utilisation of both machines. First, the sequence of machines for each job must be evaluated, which will now follow.

J_1 : The sequence of machines that J_1 must visit is (M_1,M_3,M_5,M_7,M_8), according to the data. The sequence of machines that J_1 was allocated to is (M_2,M_4,M_6,M_7,M_8), as shown in the graph. The machine difference between M_1 and M_2, or M_3 and M_4, or M_5 and M_6 does not have an effect, because these machines are identical.

J_2 : The sequence of machines that J_2 must visit is (M_3,M_5,M_7), according to the data. The sequence of machines that J_2 was allocated to is (M_4,M_6,M_7), as shown in the graph. The machine difference between M_3 and M_4, or M_5 and M_6 does not have an effect, because these machines are identical.

J_3 : The sequence of machines that J_3 must visit is (M_5,M_8), according to the data. The sequence of machines that J_3 was allocated to is (M_5,M_8), as shown in the graph.

J_4 : The sequence of machines that J_4 must visit is (M_7,M_8,M_2,M_4,M_6), according to the data. The sequence of machines that J_4 was allocated to is (M_7,M_8,M_1,M_4,M_6), as shown in the graph. The machine difference between M_1 and M_2 does not have an effect, because these machines are identical.

J_5 : The sequence of machines that J_5 must visit is (M_4), according to the data. The sequence of machines that J_5 was allocated to is (M_3), as shown in the graph. The machine difference between M_3 and M_4 does not have an effect, because these machines are identical.

Secondly, each machine must be consulted to determine if the sequence follows the sequence of Table B.9, which will now follow.

M_1 : The sequence on M_1 according to the graph is (J_4), which adheres to the correct sequence due to there being only one operation.

M_2 : The sequence on M_2 according to the graph is (J_1), which adheres to the correct sequence due to there being only one operation.

M_3 : The sequence on M_3 according to the graph is (J_5), which adheres to the correct sequence due to there being only one operation.

M_4 : The sequence on M_4 according to the graph is (J_2,J_4,J_1), which adheres to the correct sequence due to Table B.9 indicating that J_2 must come first, followed by J_4 and lastly J_1.

M_5 : The sequence on M_5 according to the graph is (J_3), which adheres to the correct sequence due to there being only one operation.

M_6 : The sequence on M_6 according to the graph is (J_2,J_4,J_1), which adheres to the correct sequence due to Table B.9 indicating that J_2 must come first, followed by J_4 and lastly J_1.

M_7 : The sequence on M_7 according to the graph is (J_2,J_4,J_1), which adheres to the correct sequence due to Table B.9 indicating that J_2 must come first, followed by J_4 and lastly J_1.

M_8 : The sequence on M_8 according to the graph is (J_3,J_4,J_1), which adheres to the correct sequence due to Table B.9 indicating that J_3 must come first, followed by J_4 and lastly J_1.

When Figure B.1 is compared to the schedule generated by the simulation scheduler, as illustrated in Figure B.2, is can be seen that the scheduler generated the exact same

schedule. The same evaluations done previously was also done with the generated schedule, which proved that the generated schedule is correct. The only difference between the two Gantt charts, is that the manually drawn schedule does not incorporate the stoppage time when the shop is closed between 17:00 and 08:00, while the schedule generated by the scheduler does incorporate the stoppage time.

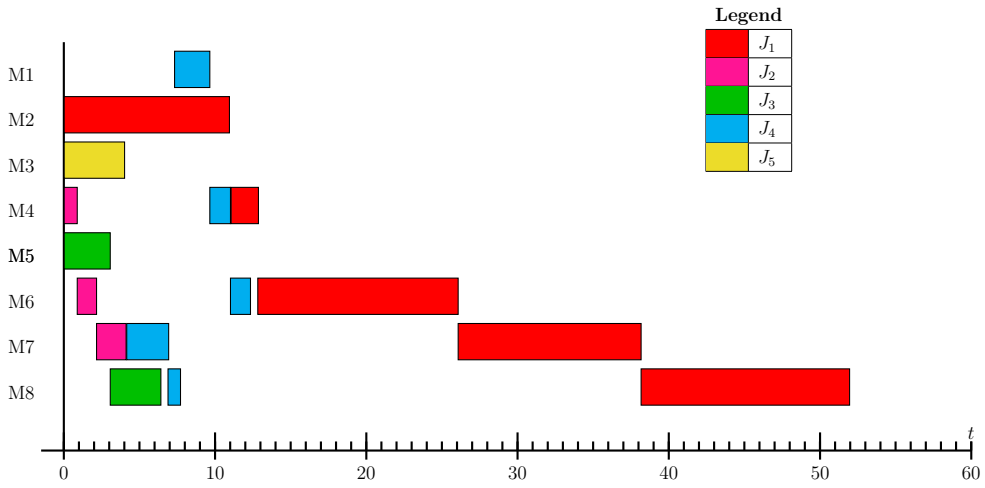


Figure B.1: Schedule drawn manually through shortest processing time dispatching rule for Test scenario 1

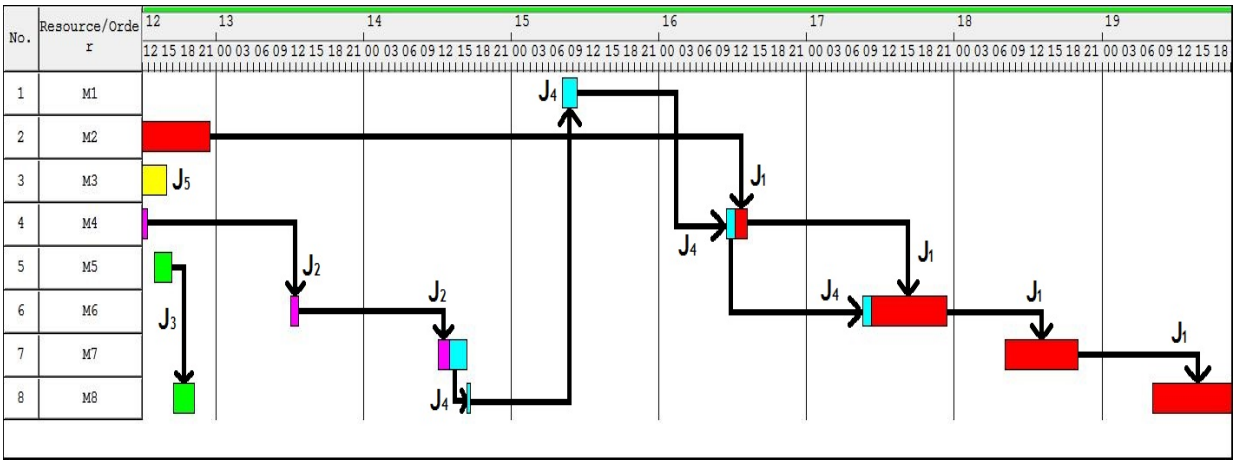


Figure B.2: Schedule generated through shortest processing time dispatching rule for Test scenario 1 (total length is 7:09:13:10)

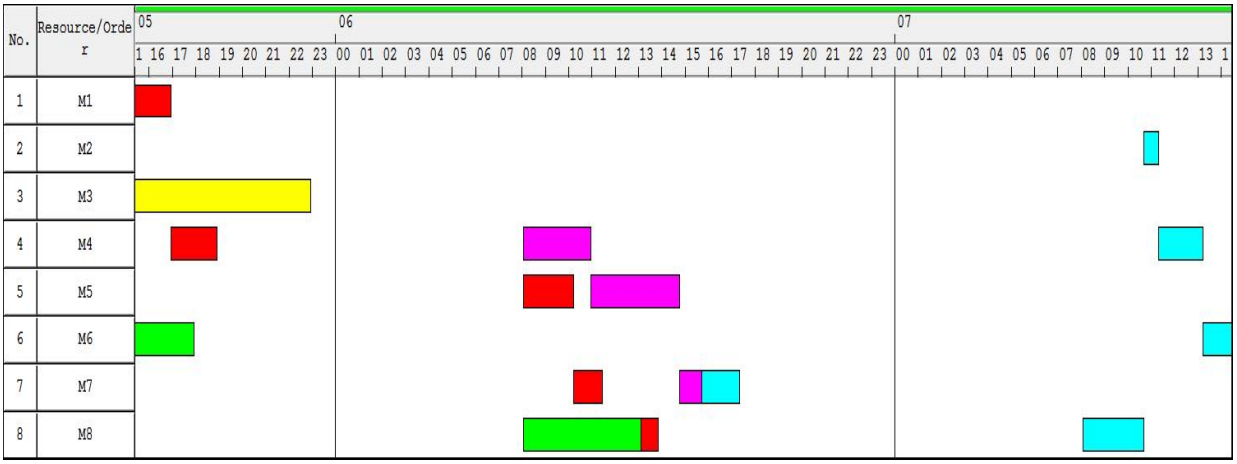


Figure B.3: Schedule generated through shortest processing time dispatching rule for Test scenario 2 (total length is 1:23:51:34)

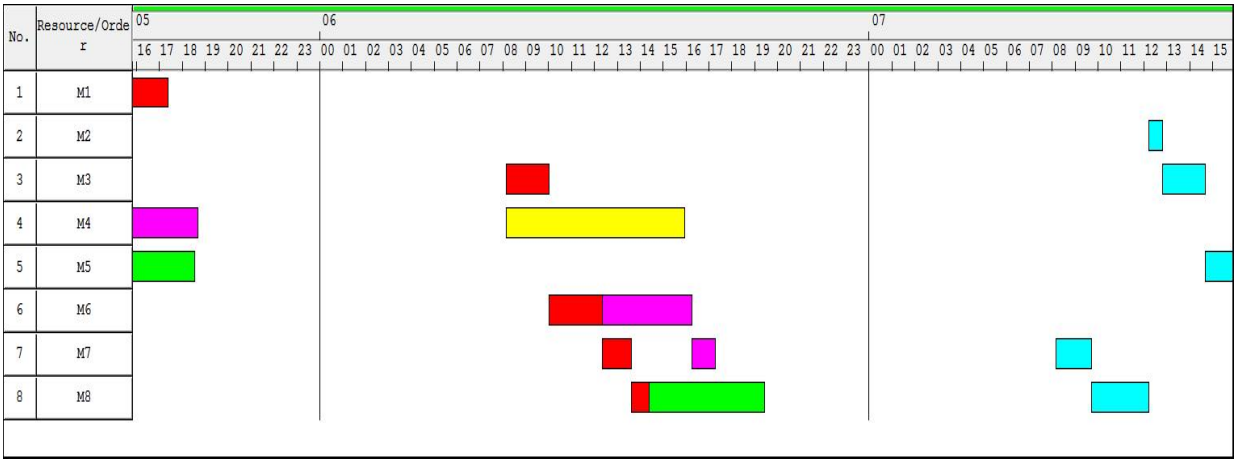


Figure B.4: Schedule generated through shortest processing time dispatching rule for Test scenario 3 (total length is 2:00:06:06)

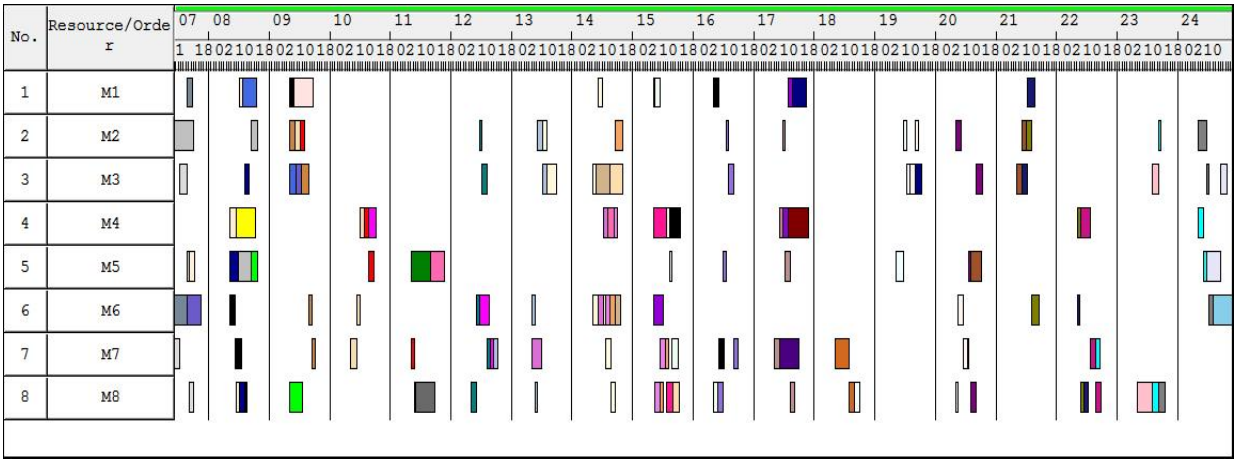


Figure B.5: Schedule generated through shortest processing time dispatching rule for Test scenario 4 (total length is 17:04:23:13)

These evaluations proved that the schedule generated by the rule for Test scenario 1, is indeed correct. The same evaluations can be done for the other schedules that were generated; however, it will only be done once for purposes of explanation.

When comparing the shortest processing time sequences and the schedules that were generated for each test scenario, it could be concluded that the job with the shortest processing time was scheduled first, followed by the sequence of entries in Tables B.9, B.10 and B.11. This illustrates that the logic behind the *shortest processing time* dispatching rule was adhered to in the first three test scenarios, and that the output of the simulation scheduler for this rule can be assumed as correct. The sequence of Test scenario 4 was also reviewed, and it proved to be in the correct order.

B.1.2.2 First-come-first-serve rule

The discussion of the schedules generated from the *first-come-first-serve* dispatching rule for each test scenario, will now follow.

When the scheduler was run with the data defined in the test scenarios, the *first-come-first-serve* dispatching rule generated the same sequence in which jobs must be processed for the first three scenarios, as shown in Table B.12. The sequence is the same for the first three test scenarios, because the order in which the jobs entered the system stayed constant in each scenario. The output of the dispatching rule was shown to be in the correct sequence, because the generated sequence follows the same sequence in which the jobs entered the system. This sequence was then used by the scheduler to generate a schedule for each test scenario as illustrated in Figures B.6, B.7, B.8 and B.9, where the

different machines are denoted by M1 – M8. The total length of the schedule is also provided in *"Days:Hours:Minutes:Seconds"*.

Table B.12: First-come-first-serve sequence for the first three test scenarios

Job_ID	1	2	3	4	5

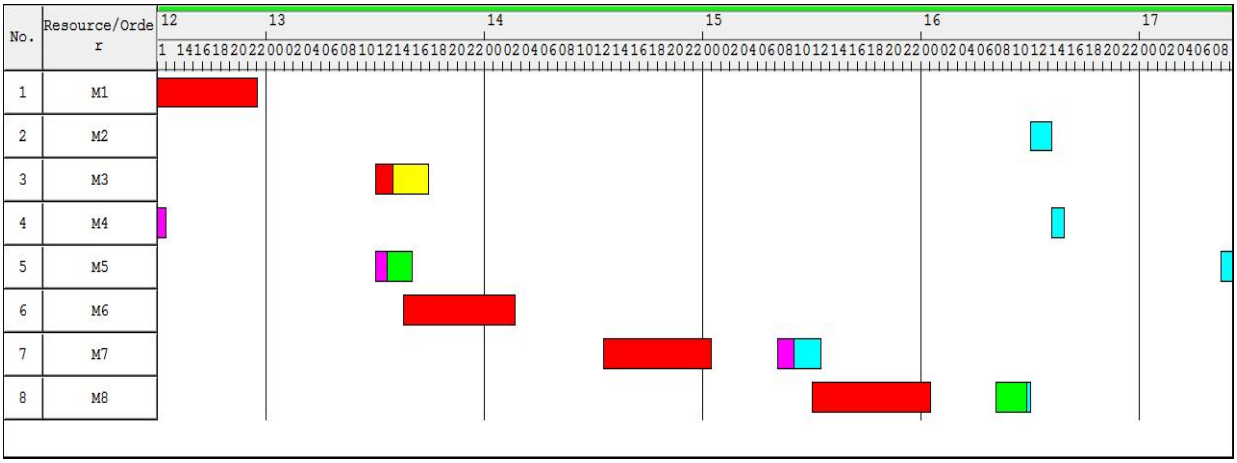


Figure B.6: Schedule generated through first-come-first-serve dispatching rule for Test scenario 1 (total length is 4:22:13:23)

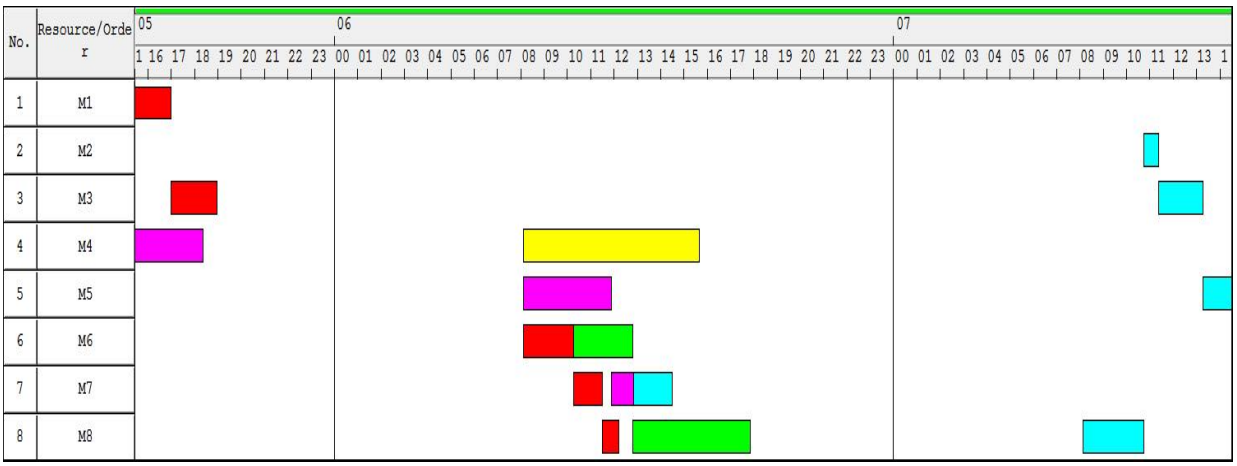


Figure B.7: Schedule generated through first-come-first-serve dispatching rule for Test scenario 2 (total length is 1:23:07:52)

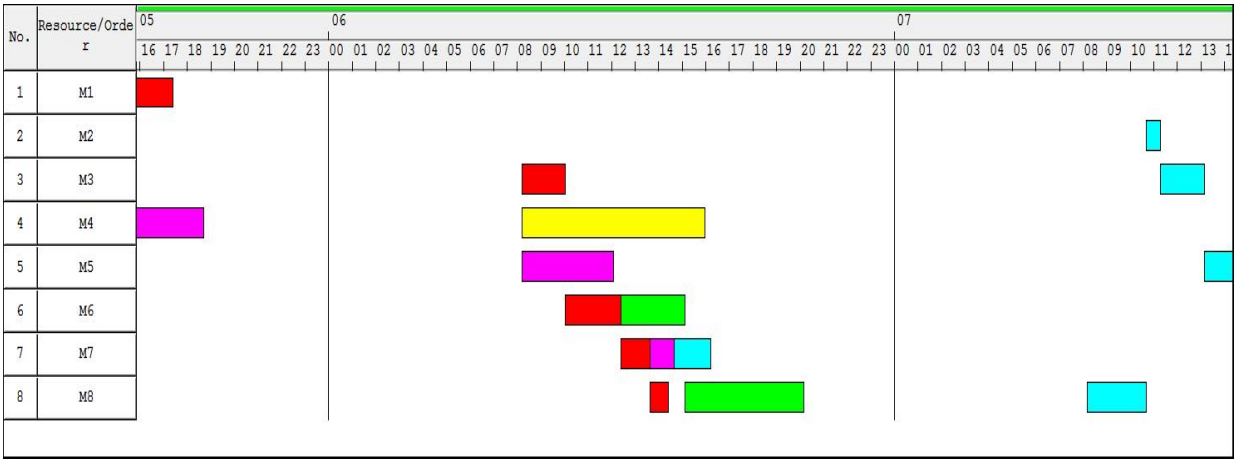


Figure B.8: Schedule generated through first-come-first-serve dispatching rule for Test scenario 3 (total length is 1:22:32:52)

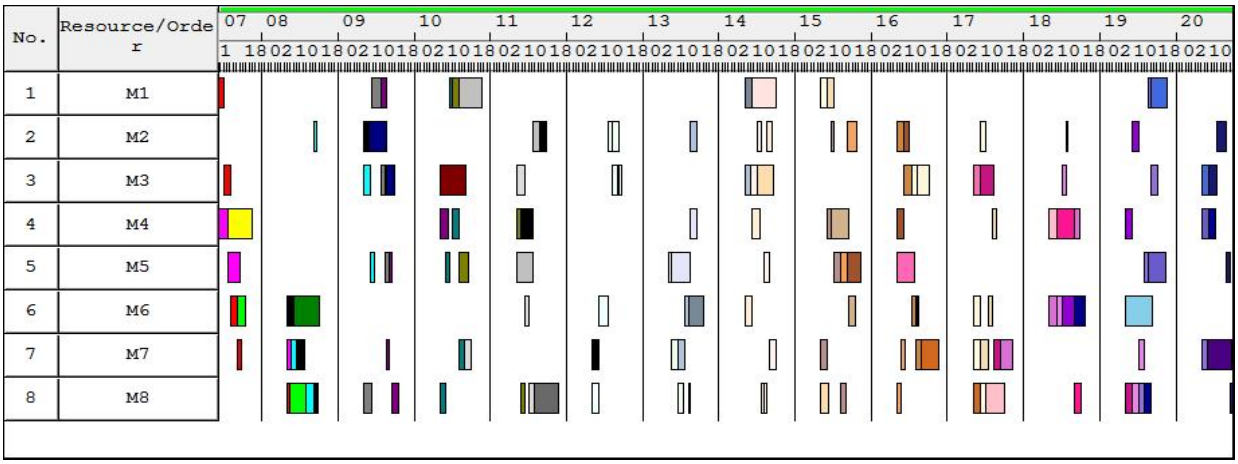


Figure B.9: Schedule generated through first-come-first-serve dispatching rule for Test scenario 4 (total length is 13:07:40:17)

When comparing the sequence of job allocation and the schedules that were generated for each test scenario, it is evident that the job that entered the system first was scheduled first, followed by the sequence of entries in Table B.12. This illustrates that the logic behind the *first-come-first-serve* dispatching rule was adhered to for the first three test scenarios, and that the output of the simulation scheduler for this rule can be assumed to be correct. The sequence of Test scenario 4 was also reviewed, and it proved to be in the correct order.

B.1.2.3 Most-important-job-first rule

The discussion of the schedules generated from the *most-important-job-first* dispatching rule for each test scenario, will now follow.

When the scheduler was run with the data defined in the test scenarios, the *most-important-job-first* dispatching rule generated the same sequence in which jobs must be processed for the first three scenarios as shown in Table B.13. The reason why the same sequence was generated for the first three test scenarios was that the priorities of the jobs stayed constant in the scenarios. The output of the dispatching rule showed to be in the correct sequence, because the generated sequence is in the order of most important, denoted by "1", to the least important, denoted by "5". This sequence was then used by the scheduler to generate a schedule for each test scenario as illustrated in Figures B.10, B.11, B.12 and B.13, where the different machines are denoted by M1 – M8. The total length of the schedule is also provided in *"Days:Hours:Minutes:Seconds"*.

Table B.13: Most-important-job-first sequence for the first three test scenarios

Job_ID	Job priority
3	1
5	2
2	3
4	3
1	5

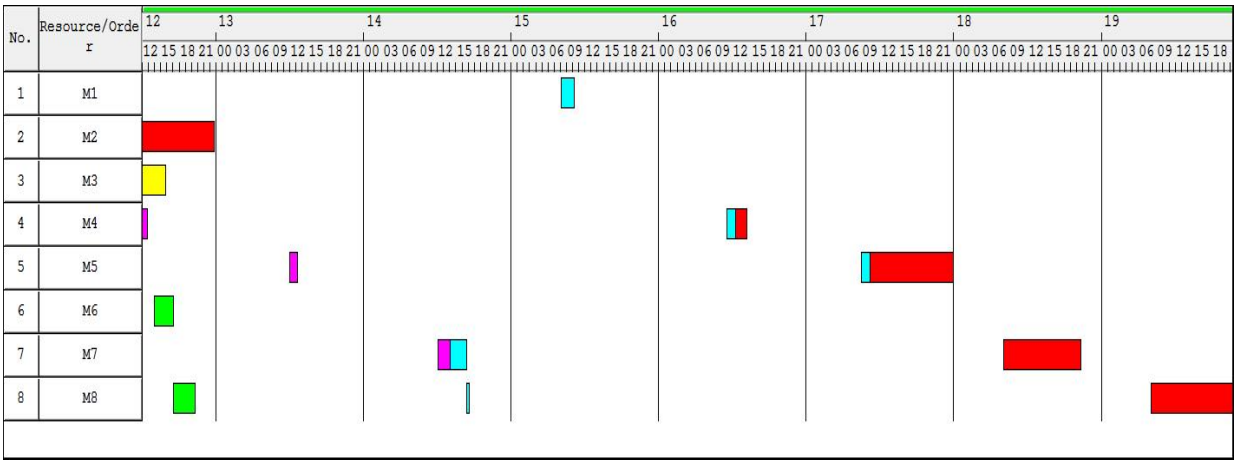


Figure B.10: Schedule generated through most-important-job-first dispatching rule for Test scenario 1 (total length is 7:09:18:15)

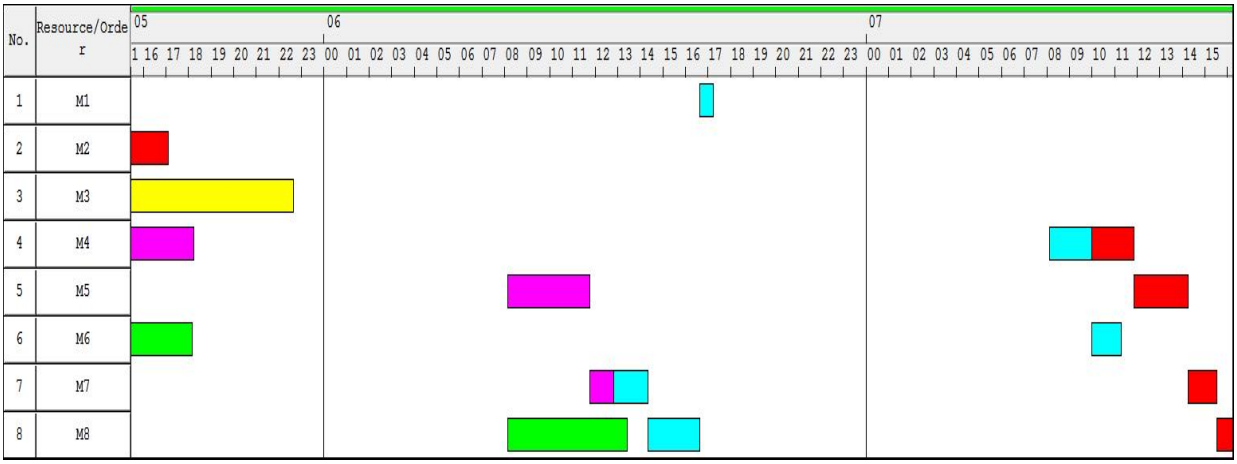


Figure B.11: Schedule generated through most-important-job-first dispatching rule for Test scenario 2 (total length is 2:00:48:31)

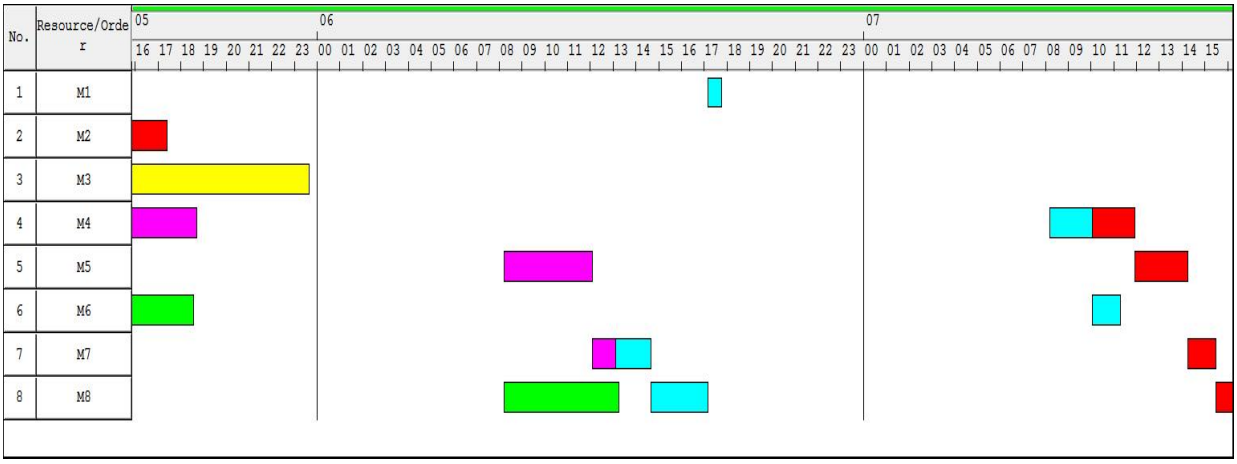


Figure B.12: Schedule generated through most-important-job-first dispatching rule for Test scenario 3 (total length is 2:00:11:56)

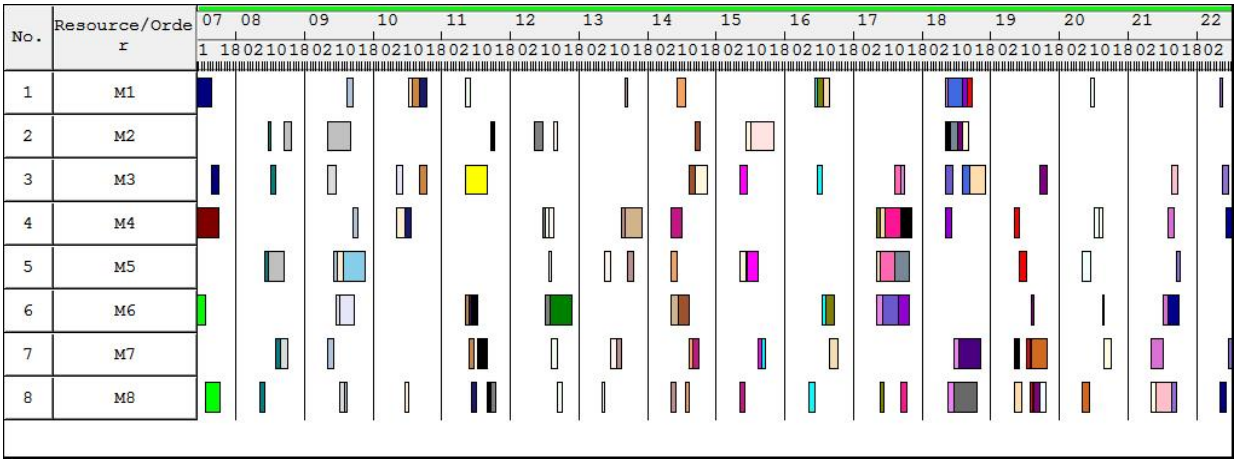


Figure B.13: Schedule generated through most-important-job-first dispatching rule for Test scenario 4 (total length is 14:20:53:59)

When comparing the sequence of job allocation and the schedule that was generated for each scenario, it is evident that the job with the highest priority was scheduled first, followed by the sequence of entries in Table B.13. This illustrates that the logic behind the *most-important-job-first* dispatching rule was adhered to, and that the output of the simulation scheduler for this rule can be assumed to be correct.

B.1.2.4 Earliest due date rule

The discussion of the schedules generated from the *earliest due date* dispatching rule for each test scenario, will now follow.

When the scheduler was run with the data defined in the test scenarios, the *earliest due date* dispatching rule generated a sequence in which jobs must be processed for each scenario as shown in Tables B.14, B.15 and B.16. The output of the dispatching rule was shown to be in the correct sequence, because the generated sequences started with the job that has the earliest due date, and it ends with the job that has the latest due date. These sequences were then used by the scheduler to generate a schedule for each scenario as illustrated in Figures B.14, B.15, B.16 and B.17, where the different machines are denoted by M1 – M8. The total length of the schedule is also provided in *"Days:Hours:Minutes:Seconds"*.

Table B.14: Earliest due date sequence for Test scenario 1

Job_ID	Job due date
3	2018/06/13
2	2018/06/15
5	2018/06/26
1	2018/06/30
4	2018/06/30

Table B.15: Earliest due date sequence for Test scenario 2

Job_ID	Job due date
4	2018/06/19
1	2018/06/20
2	2018/06/20
3	2018/06/20
5	2018/06/22

Table B.16: Earliest due date sequence for Test scenario 3

Job_ID	Job due date
4	2018/06/19
1	2018/06/20
2	2018/06/20
3	2018/06/20
5	2018/06/22

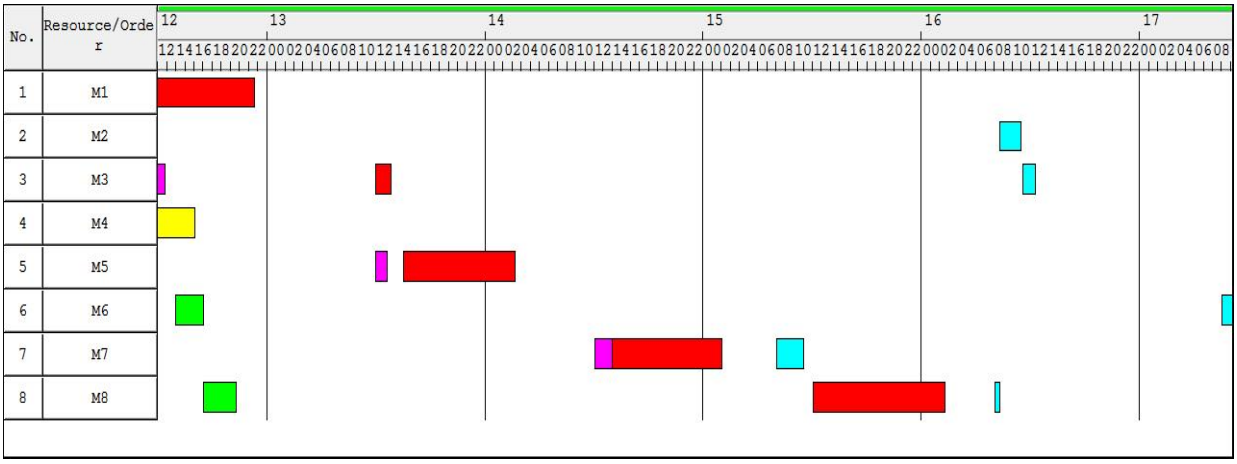


Figure B.14: Schedule generated through earliest due date dispatching rule for Test scenario 1 (total length is 4:22:02:14)

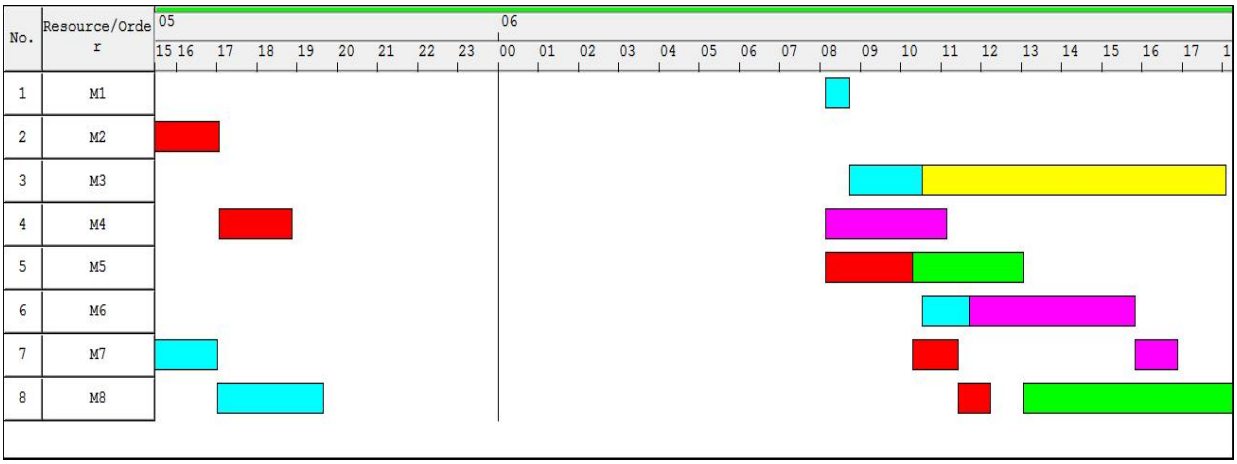


Figure B.15: Schedule generated through earliest due date dispatching rule for Test scenario 2 (total length is 1:02:58:03)

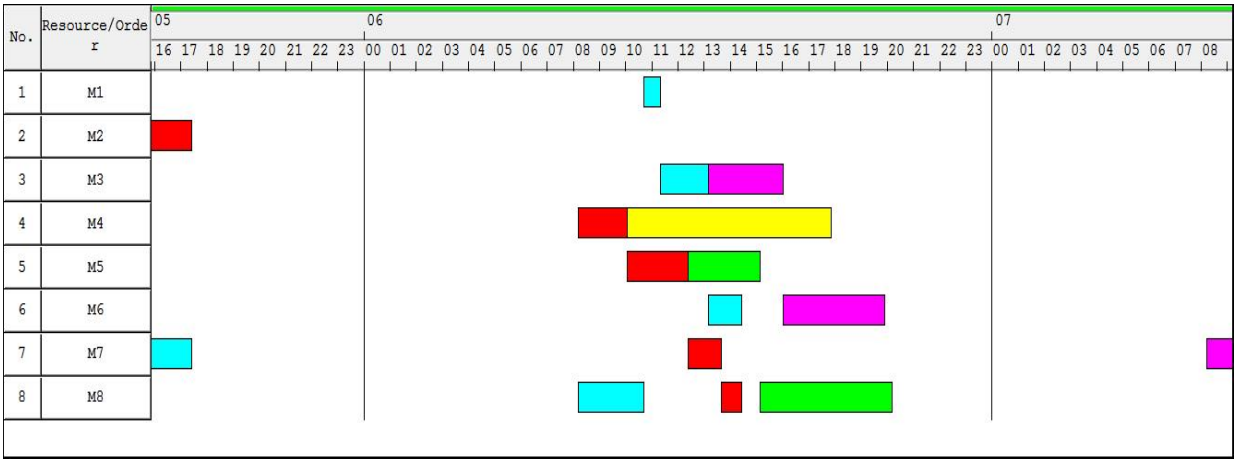


Figure B.16: Schedule generated through earliest due date dispatching rule for Test scenario 3 (total length is 1:17:02:09)

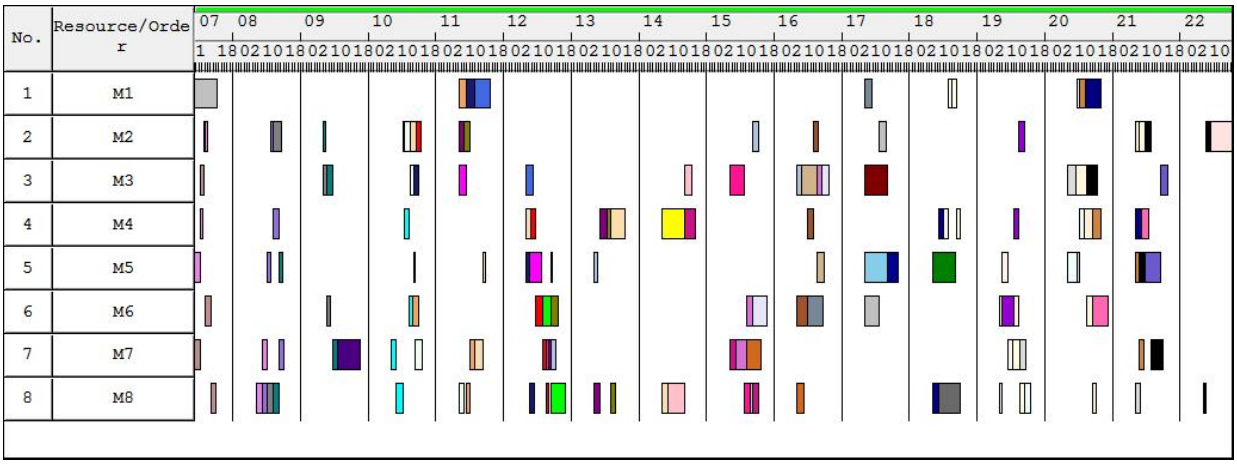


Figure B.17: Schedule generated through earliest due date dispatching rule for Test scenario 4 (total length is 15:02:17:49)

When comparing the sequence of job allocation and the schedules that were generated for each test scenario, it is evident that the job with the earliest due date was scheduled first, followed by the sequence of entries in Tables B.14, B.15 and B.16. This illustrates that the logic behind the *earliest due date* dispatching rule was adhered to in the first three test scenarios, and that the output of the simulation scheduler for this rule can be assumed to be correct. The sequence of Test scenario 4 was also reviewed, and it proved to be in the correct order.

B.1.2.5 Critical ratio rule

The discussion of the schedules generated from the *critical ratio* dispatching rule for each test scenario, will now follow. The critical ratio for each job was calculated using

$$\frac{Time\ until\ due\ date}{Remaining\ processing\ time}.$$

When the scheduler was run with the data defined in the test scenarios, the *critical ratio* dispatching rule generated a sequence in which jobs must be processed for each scenario as shown in Tables B.17, B.18 and B.19. The output of the dispatching rule was shown to be in the correct sequence, because the generated sequences started with the job that has the lowest critical ratio, and it ends with the job that has the highest critical ratio. These sequences were then used by the scheduler to generate a schedule for the scenarios as illustrated in Figures B.18, B.19, B.20 and B.21, where the different machines are denoted by M1 – M8. The total length of the schedule is also provided in *"Days:Hours:Minutes:Seconds"*.

Table B.17: Critical ratio sequence for Test scenario 1

Job_ID	Processing time (hh:mm:ss)	Date difference	Critical ratio
1	49:59:48	588:44:41	11.8
3	6:11:19	180:44:41	29.2
2	3:55:27	228:44:41	58.3
4	8:30:21	588:44:41	69.2
5	3:52:48	492:44:41	127.0

Table B.18: Critical ratio sequence for Test scenario 2

Job_ID	Processing time (hh:mm:ss)	Date difference	Critical ratio
4	8:05:04	320:29:58	39.6
2	7:38:33	344:29:58	45.1
1	7:37:33	344:29:58	45.2
3	7:34:08	344:29:58	45.5
5	7:32:09	392:29:58	52.1

Table B.19: Critical ratio sequence for Test scenario 3

Job_ID	Processing time (hh:mm:ss)	Date difference	Critical ratio
4	7:46:06	320:06:39	41.2
2	7:46:06	344:06:39	44.3
1	7:46:05	344:06:39	44.3
3	7:46:05	344:06:39	44.3
5	7:46:08	392:06:39	50.5

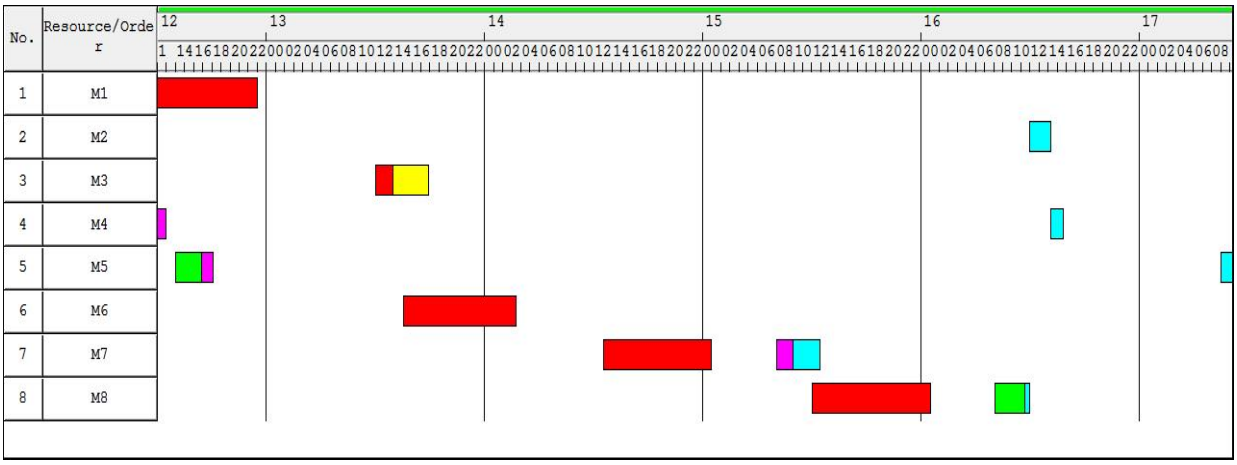


Figure B.18: Schedule generated through critical ratio dispatching rule for Test scenario 1 (total length is 4:21:53:46)

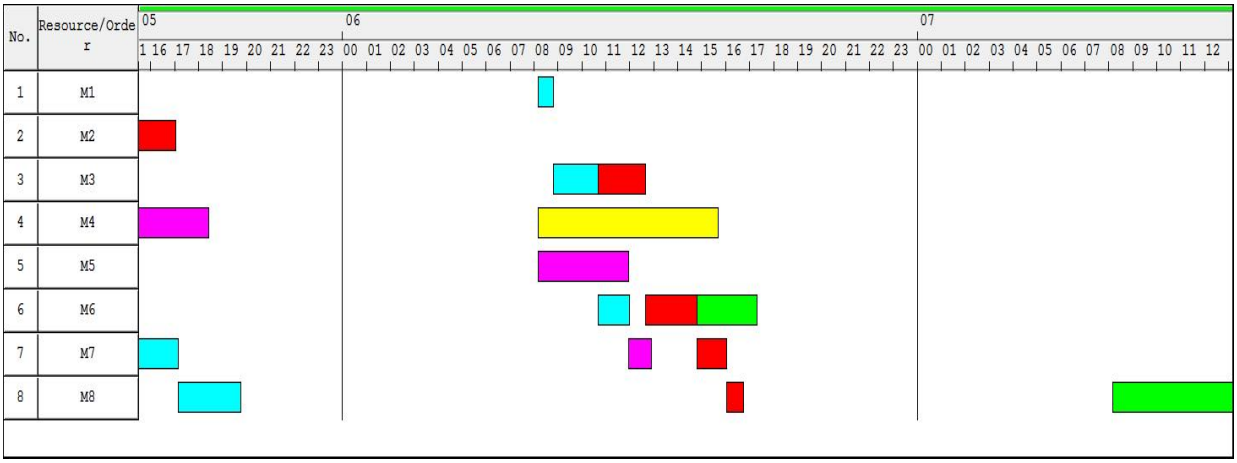


Figure B.19: Schedule generated through critical ratio dispatching rule for Test scenario 2 (total length is 1:21:43:15)

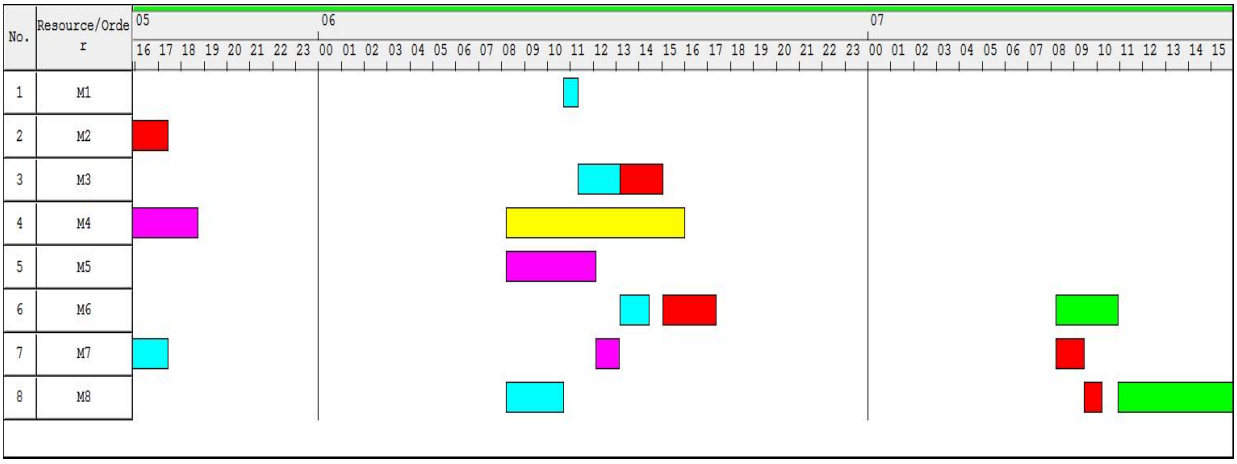


Figure B.20: Schedule generated through critical ratio dispatching rule for Test scenario 3 (total length is 1:23:46:05)

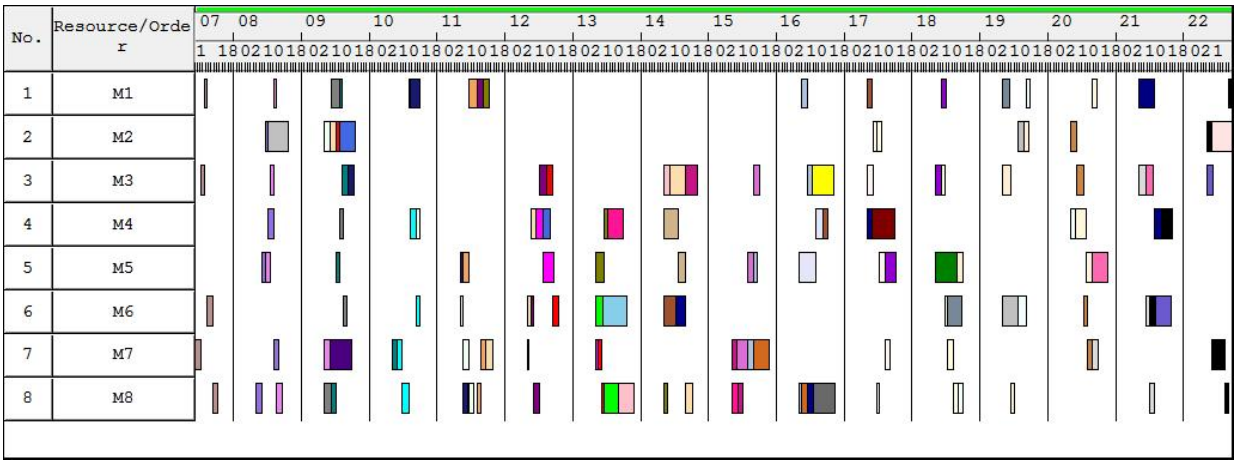


Figure B.21: Schedule generated through critical ratio dispatching rule for Test scenario 4 (total length is 15:06:09:51)

When comparing the sequence of job allocation and the schedules that were generated for each scenario, it is evident that the job with the lowest critical ratio was scheduled first, followed by the sequences shown in Tables B.17, B.18 and B.19. This illustrates that the logic behind the *critical ratio* dispatching rule was adhered to in each scenario, and that the output of the scheduler for this rule can be assumed to be correct.

B.1.2.6 Minimum slack time rule

The discussion of the schedules generated from the *minimum slack time* dispatching rule for each test scenario, will now follow.

When the scheduler was run with the data defined in the test scenarios, the *minimum slack time* dispatching rule generated a sequence in which jobs must be processed for each scenario as shown in Tables B.20, B.21 and B.22. The output of the dispatching rule was shown to be in the correct sequence, because the generated sequences started with the job that has the minimum slack time, and ended with the job that has the maximum slack time. These sequences were then used by the scheduler to generate a schedule for the scenarios as illustrated in Figures B.22, B.23, B.24 and B.25, where the different machines are denoted by M1 – M8. The total length of the schedule is also provided in *"Days:Hours:Minutes:Seconds"*.

Table B.20: Minimum slack time sequence for Test scenario 1

Job_ID	**Slack time (hh:mm:ss)**
2	15:00:00
3	40:40:00
4	251:00:00
5	337:00:00
1	1128:00:00

Table B.21: Minimum slack time sequence for Test scenario 2

Job_ID	**Slack time (hh:mm:ss)**
5	387:13:00
3	1011:46:00
2	1017:41:00
4	1576:09:00
1	1696:09:00

Table B.22: Minimum slack time sequence for Test scenario 3

Job_ID	**Slack time (hh:mm:ss)**
5	387:13:00
3	1011:46:00
2	1017:41:00
4	1576:09:00
1	1696:09:00

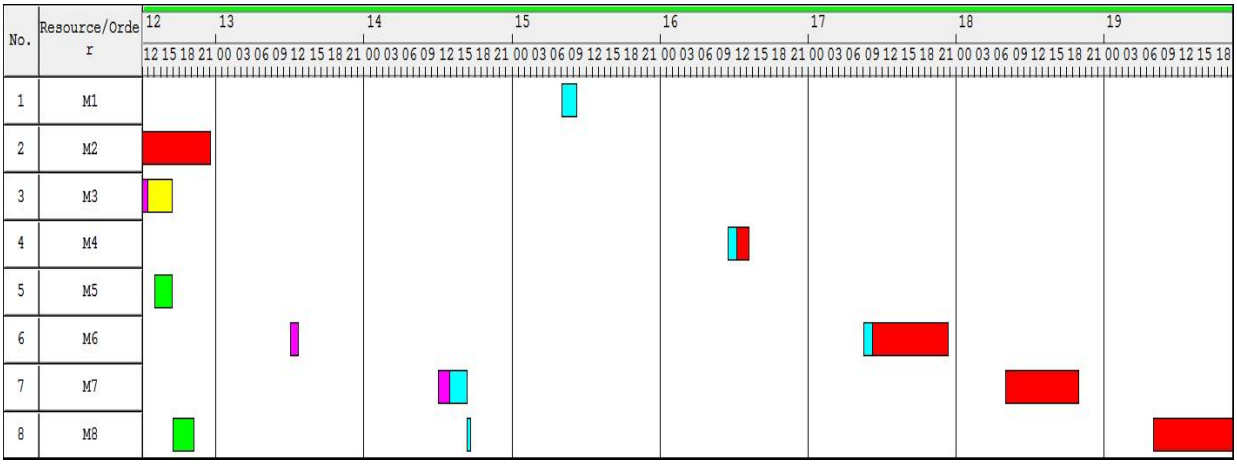


Figure B.22: Schedule generated through minimum slack time dispatching rule for Test scenario 1 (total length is 7:08:15:51)

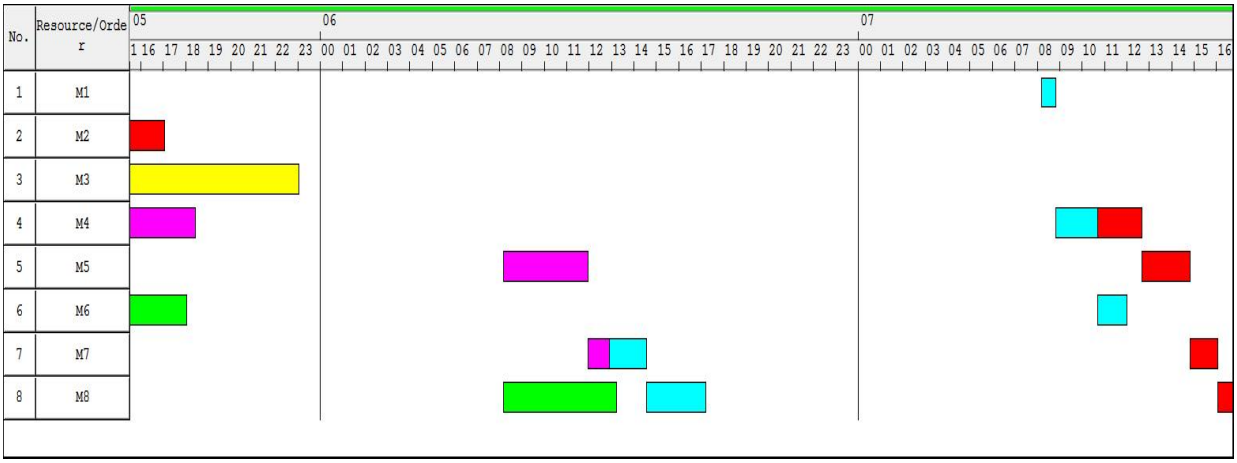


Figure B.23: Schedule generated through minimum slack time dispatching rule for Test scenario 2 (total length is 2:00:23:21)

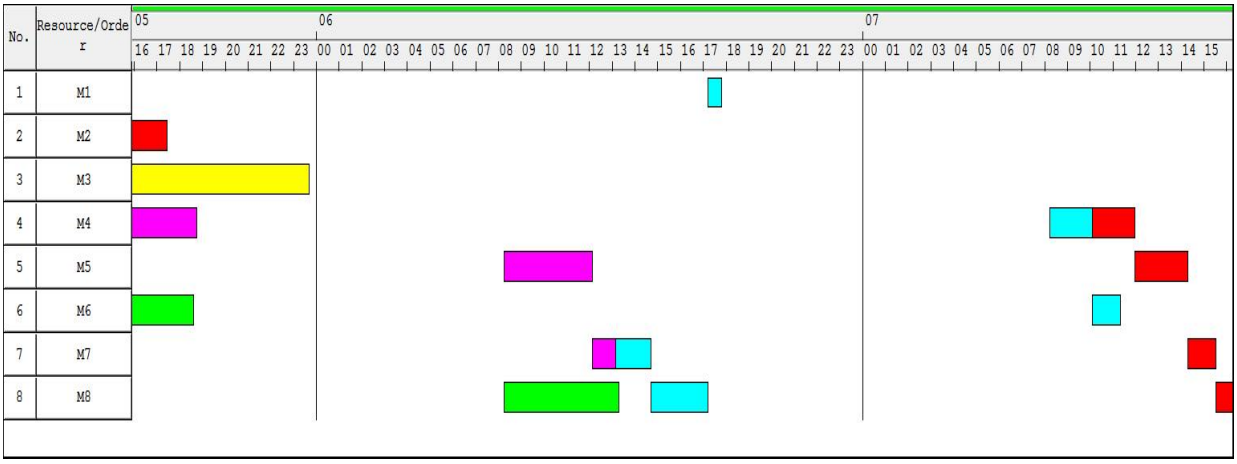


Figure B.24: Schedule generated through minimum slack time dispatching rule for Test scenario 3 (total length is 2:00:04:44)

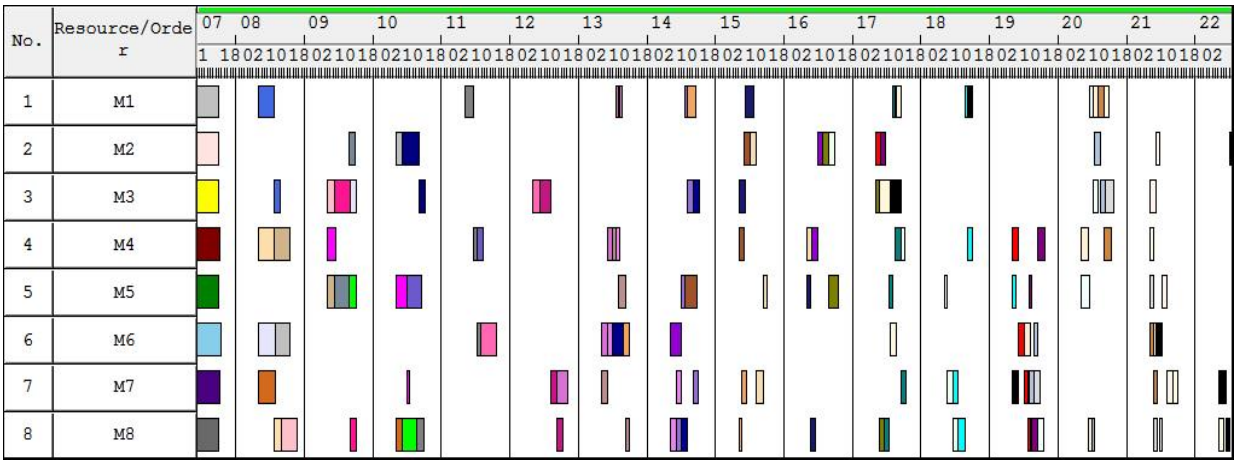


Figure B.25: Schedule generated through minimum slack time dispatching rule for Test scenario 4 (total length is 15:03:51:36)

When comparing the sequence of job allocation and the schedules that were generated for each scenario, it is evident that the job with the minimum slack time was scheduled first, followed by the sequence of entries in Tables B.20, B.21 and B.22. This illustrates that the logic behind the *minimum slack time dispatching* rule was adhered to in each scenario, and that the output of the simulation scheduler for this rule can be assumed to be correct.

B.1.3 Selection of new schedule

After the schedule for each dispatching rule is generated by the simulation scheduler, the outcome of these schedules according to the KPIs can be compared. The comparison

between the different schedules was built into the scheduler where a report is generated that displays the confidence intervals of the outcome of each dispatching rule for the different KPIs.

From these confidence intervals, the manager can then select which dispatching rule would be the most preferable for minimising the desired KPI. The confidence intervals and ANOVA tables for each KPI will now be discussed with reference to the different test scenarios. The experiments in the results represent the application of a specific dispatching rule. The dispatching rule that is associated with each experiment is listed in Table B.23.

Table B.23: Dispatching rule and experiment association

Dispatching rule	**Experiment number**
Shortest processing time	1
First-come-first-serve	2
Most-important-job-first	3
Earliest due date	4
Critical ratio	5
Minimum slack time	6

B.1.3.1 Test scenario 1 schedule selection

The results from the application of each dispatch rule in Test scenario 1 will now be discussed and the best-performing dispatching rule will be identified per KPI.

Average flow time

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.26 and B.27, of the *average flow time* KPI, it can be deduced that the best-performing dispatching rule was the *earliest due date* rule. The ANOVA table suggests that there is statistical difference between this rule and the other dispatching rules. When the outcome of the experiments was analysed it could be concluded that the *earliest due date* rule was indeed the best-performing rule. This is due to the fact that the rule, together with the *first-come-first-serve* and the *critical ratio* rules, provided the shortest schedule (as illustrated in Figures B.6, B.14 and B.18). However, what set the earliest due date rule apart, was that *J5*, which only had one operation, was processed earlier than in the other two schedules. This ultimately caused the *average flow time* of jobs to be the shortest of all the dispatching rules.

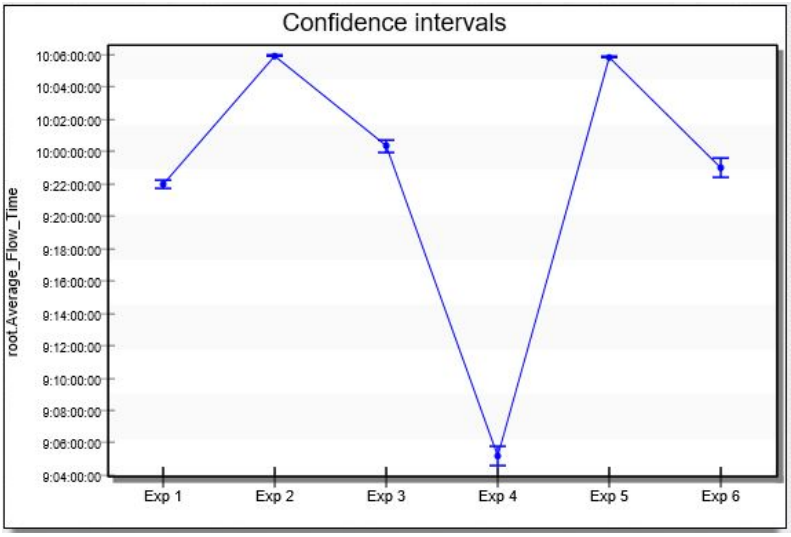


Figure B.26: Confidence intervals for the average flow time KPI for Test scenario 1

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0	0	0	0	0.003
Exp 2		0	0	0.017	0
Exp 3			0	0	0.001
Exp 4				0	0
Exp 5					0

Figure B.27: ANOVA results for the average flow time KPI for Test scenario 1

Average queue time

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.28 and B.29, of the *average queue time* KPI, it can be deduced that the best-performing dispatching rule was the *earliest due date* rule. The ANOVA table suggests that there is statistical difference between this rule and the other dispatching rules. When the outcome of the experiments was analysed it could be concluded that the *earliest due date* rule was indeed the best-performing rule. This is due to the fact that the rule, together with the *first-come-first-serve* and the *critical ratio* rules, provided the shortest schedule (as illustrated in Figures B.6, B.14 and B.18). However, what set the earliest due date rule apart, was that *J5*, which only had one operation, was processed earlier than in the other two schedules. This meant that *J5* did not have to queue before processing, and therefore the *average queue time* of jobs in the system is shorter.

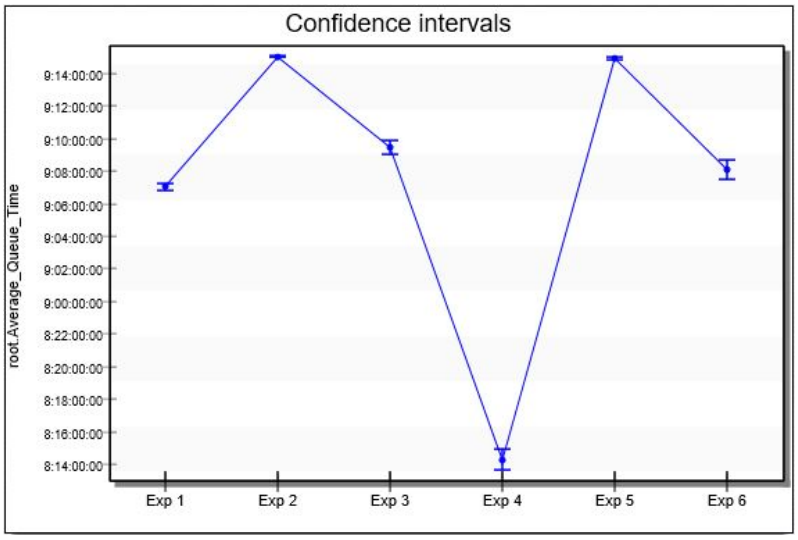


Figure B.28: Confidence intervals for the average queue time KPI for Test scenario 1

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0	0	0	0	0.003
Exp 2		0	0	0.032	0
Exp 3			0	0	0.001
Exp 4				0	0
Exp 5					0

Figure B.29: ANOVA results for the average queue time KPI for Test scenario 1

Average job tardiness

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.30 and B.31, of the *average job tardiness* KPI, it can be deduced that the best-performing dispatching rule was the *shortest processing time* rule. The ANOVA table suggests that there is statistical difference between this rule and the other dispatching rules. When the outcome of the experiments was analysed it could be concluded that the *shortest processing time* rule was indeed one of the best-performing rules. This is due to the fact that the rule, together with the *most-important-job-first*, *earliest due date*, and *minimum slack time* rules, provided the shortest schedule (as illustrated in Figures B.2, B.10, B.14 and B.22). All four of these dispatching rules completed the jobs before or just after their respective due dates. However, when taking the stochastic nature of the processing times into account, the *shortest processing time* rule performed the best.

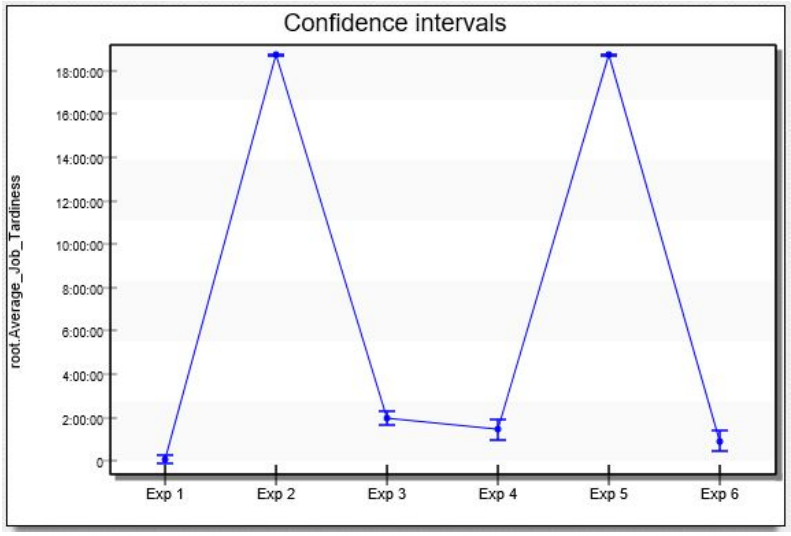


Figure B.30: Confidence intervals for the average job tardiness KPI for Test scenario 1

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0	0	0	0	0.002
Exp 2		0	0	0.732	0
Exp 3			0.051	0	0
Exp 4				0	0.097
Exp 5					0

Figure B.31: ANOVA results for the average job tardiness KPI for Test scenario 1

Average job lateness

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.32 and B.33, of the *average job lateness* KPI, it can be deduced that the best-performing dispatching rule was the *earliest due date* rule. The ANOVA table suggests that there is statistical difference between this rule and the other dispatching rules. When the outcome of the experiments was analysed it could be concluded that the *earliest due date* rule was indeed one of the best-performing rules. This is due to the fact that the rule provided the shortest schedule (as illustrated in Figure B.14), which then caused the *average lateness* of the jobs to be the least of all the other dispatching rules.

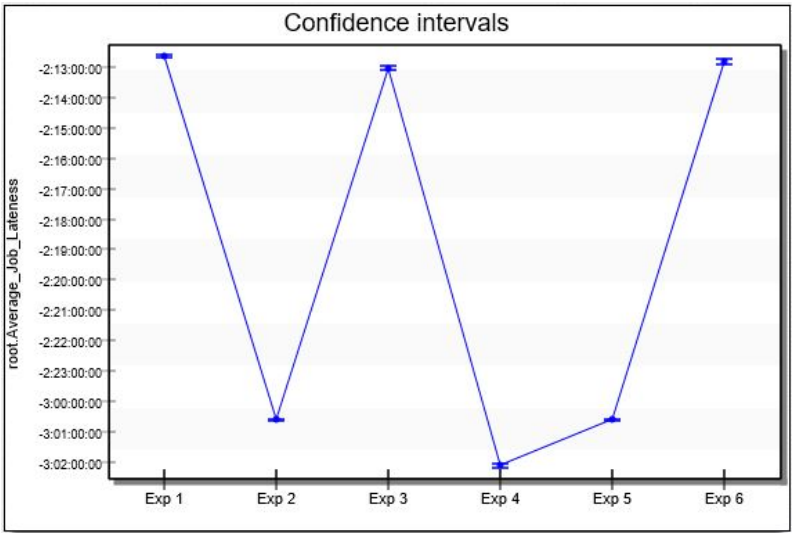


Figure B.32: Confidence intervals for the average job lateness KPI for Test scenario 1

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0	0	0	0	0.001
Exp 2		0	0	0.957	0
Exp 3			0	0	0
Exp 4				0	0
Exp 5					0

Figure B.33: ANOVA results for the average job lateness KPI for Test scenario 1

Makespan

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.34 and B.35, of the *makespan* KPI, it can be deduced that the best-performing dispatching rules were the *earliest due date* and *critical ratio* rules. The ANOVA table suggests that there is no statistical difference between these two dispatching rules and therefore either one of them can be chosen. When the outcome of the experiments was analysed it could be concluded that these two rules were indeed the best-performing rules. This is due to the fact that these rules generated two of the shortest schedules of all the dispatching rules (as illustrated in Figures B.14 and B.18), which minimised the *makespan* of the system.

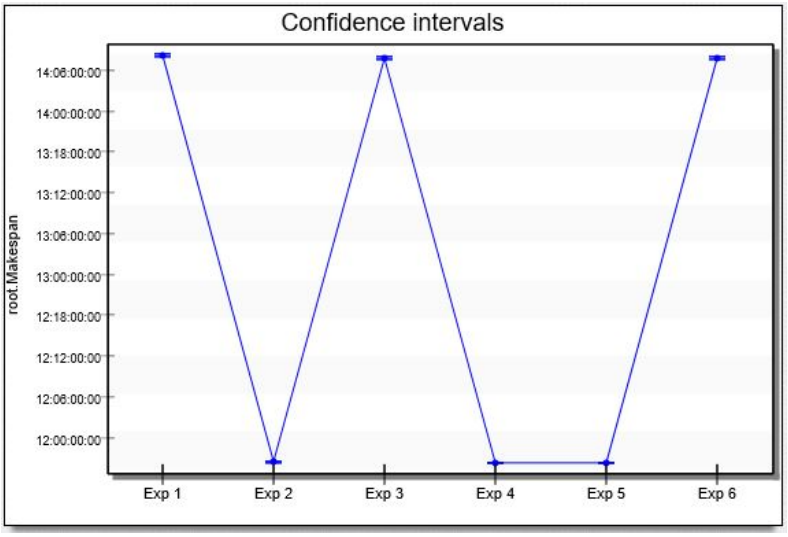


Figure B.34: Confidence intervals for the makespan KPI for Test scenario 1

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0	0.029	0	0	0.052
Exp 2		0	0	0	0
Exp 3			0	0	0.817
Exp 4				0.201	0
Exp 5					0

Figure B.35: ANOVA results for the makespan KPI for Test scenario 1

B.1.3.2 Test scenario 2 schedule selection

The results from the application of each dispatch rule in Test scenario 2 will now be discussed and the best-performing dispatching rule will be identified per KPI.

Average flow time

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.36 and B.37, of the *average flow time* KPI, it can be deduced that the best-performing dispatching rule was the *earliest due date* rule. The ANOVA table suggests that there is statistical difference between this rule and the other rules, therefore the *earliest due date* rule must be chosen. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact that the rule generated the shortest schedule of all the dispatching rules (as illustrated in Figure B.15), which caused the *average flow time* of jobs to be the shortest of all the dispatching rules.

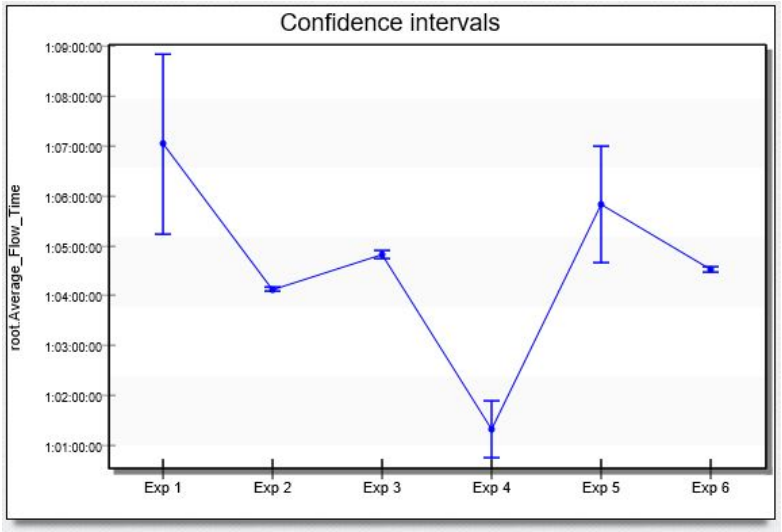


Figure B.36: Confidence intervals for the average flow time KPI for Test scenario 2

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0.003	0.018	0	0.253	0.008
Exp 2		0	0	0.006	0
Exp 3			0	0.089	0
Exp 4				0	0
Exp 5					0.03

Figure B.37: ANOVA results for the average flow time KPI for Test scenario 2

Average queue time

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.38 and B.39, of the *average queue time* KPI, it can be deduced that the best-performing dispatching rule is again the *earliest due date* rule. The ANOVA table suggests that there is statistical difference between this rule and the other rules, therefore this rule must be chosen. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact that the rule generated the shortest schedule of all the dispatching rules (as illustrated in Figure B.15), which caused the *average queue time* of jobs to be the shortest of all the dispatching rules.

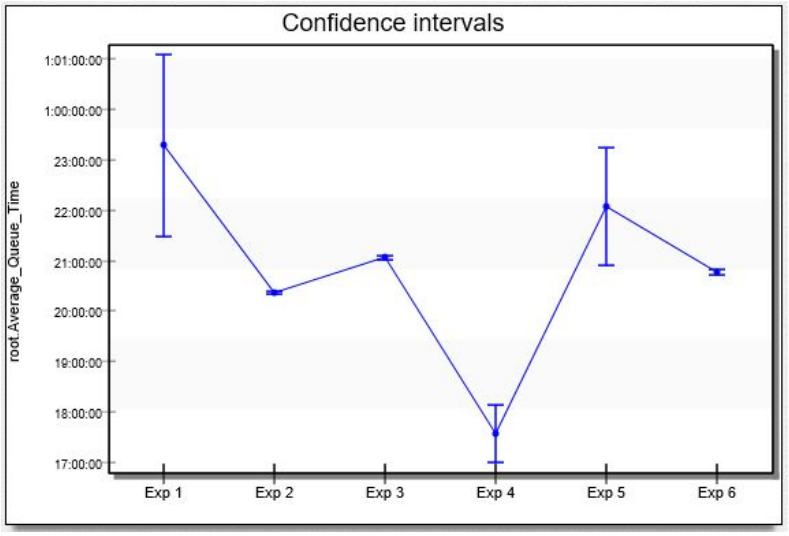


Figure B.38: Confidence intervals for the average queue time KPI for Test scenario 2

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0.003	0.018	0	0.253	0.008
Exp 2		0	0	0.006	0
Exp 3			0	0.086	0
Exp 4				0	0
Exp 5					0.03

Figure B.39: ANOVA results for the average queue time KPI for Test scenario 2

Average job tardiness

When referring to the confidence intervals, as illustrated in Figure B.40, of the *average job tardiness* KPI, it can be deduced that there were no jobs that were completed after their due date. This means that an ANOVA table could not be created to compare the outcomes of the different dispatching rules. The confidence intervals therefore suggest that any of the dispatching rules could be selected if the *average job tardiness* KPI was to be minimised.

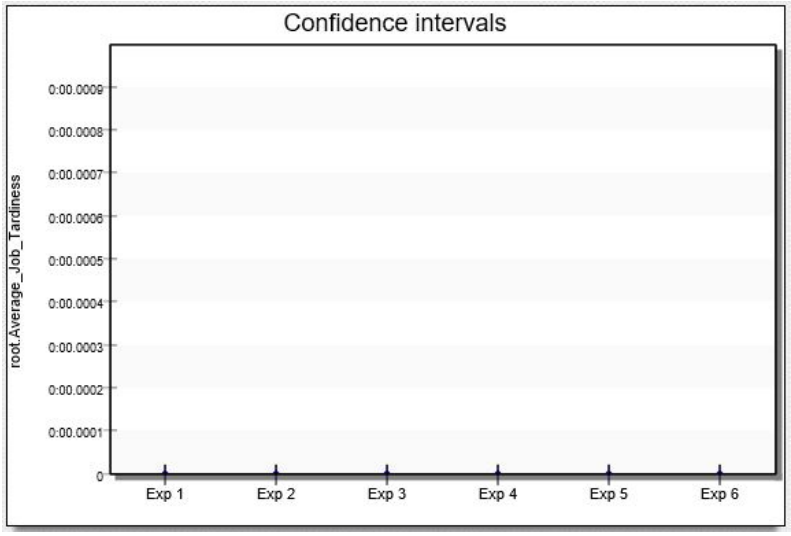


Figure B.40: Confidence intervals for the average job tardiness KPI for Test scenario 2

Average job lateness

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.41 and B.42, of the *average job lateness* KPI, it can be deduced that the best-performing dispatching rule is the *earliest due date* rule. The ANOVA table suggests that there is statistical difference between this rule and the other rules, therefore this rule must be selected. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact that the rule generated the shortest schedule of all the dispatching rules (as illustrated in Figure B.15), which caused the *average lateness* of the jobs to be the least of all the dispatching rules.

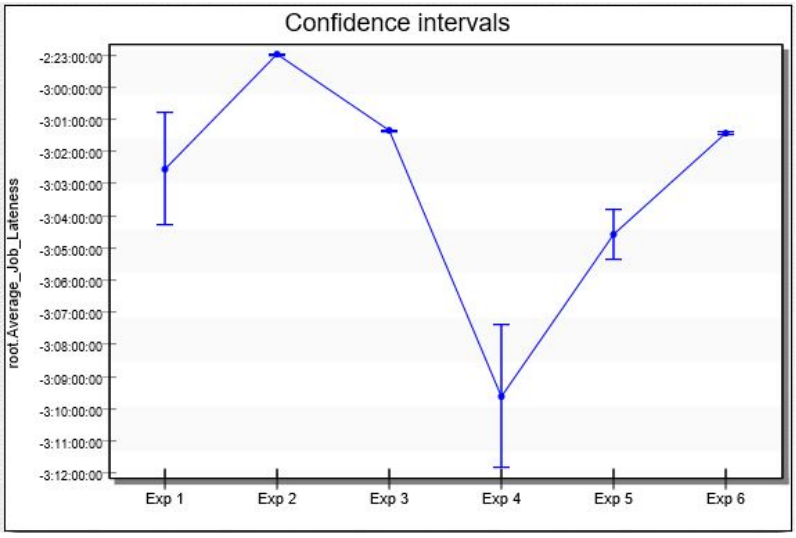


Figure B.41: Confidence intervals for the average job lateness KPI for Test scenario 2

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0	0.176	0	0.035	0.201
Exp 2		0	0	0	0
Exp 3			0	0	0.009
Exp 4				0	0
Exp 5					0

Figure B.42: ANOVA results for the average job lateness KPI for Test scenario 2

Makespan

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.43 and B.44, of the *makespan* KPI, it can be deduced that the best-performing dispatching rule is the *earliest due date* rule. The ANOVA table suggests that there is statistical difference between this rule and the other rules, therefore this rule must be selected. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact that the rule generated the shortest schedule of all the dispatching rules (as illustrated in Figure B.15), which caused the *makespan* to be the shortest of all the dispatching rules.

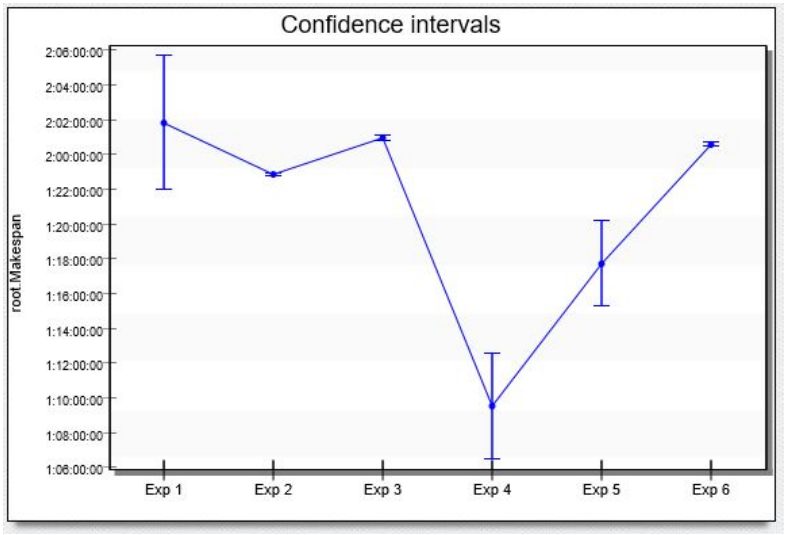


Figure B.43: Confidence intervals for the makespan KPI for Test scenario 2

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0.122	0.638	0	0.001	0.513
Exp 2		0	0	0	0
Exp 3			0	0	0.001
Exp 4				0	0
Exp 5					0

Figure B.44: ANOVA results for the makespan KPI for Test scenario 2

B.1.3.3 Test scenario 3 schedule selection

The results from the application of each dispatch rule in Test scenario 3 will now be discussed and the best-performing dispatching rule will be identified per KPI. Due to the deterministic processing times that were used to ensure that the processing times are equal, the results will not include any ANOVA tables or confidence intervals which compare the outcomes of the different dispatching rules. The expected values of KPIs for each dispatching rule, are illustrated in Figure B.45.

	root.DispatchingRule	root.Average_Flow_Time	root.Average_Queue_Time	root.Average_Job_Tardiness	root.Average_Job_Lateness	root.Makespan
Exp 1	1	1:05:33:34.2000	21:47:28.2000	0.0000	-2:22:17:24.9133	2:00:06:06.0000
Exp 2	2	1:04:51:38.2000	21:05:32.2000	0.0000	-2:22:53:06.6477	1:22:32:52.0000
Exp 3	3	1:04:16:50.6000	20:30:44.6000	0.0000	-3:01:27:08.2513	2:00:11:56.0000
Exp 4	4	1:03:48:45.0000	20:02:39.0000	0.0000	-3:04:45:58.6425	1:17:02:09.0000
Exp 5	5	1:07:20:16.6000	23:34:10.6000	0.0000	-3:02:50:03.8911	1:23:46:05.0000
Exp 6	6	1:04:11:05.0000	20:24:59.0000	0.0000	-3:01:27:31.0637	2:00:04:44.0000

Figure B.45: Expected values of KPIs for each dispatching rule

Average flow time

When referring to Figure B.45, it can be deduced from the results of the *average flow time* KPI, that the best-performing dispatching rule was the *earliest due date* rule, which had the minimum value. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact that the rule generated the shortest schedule of all the dispatching rules (as illustrated in Figure B.16), which caused the *average flow time* of the jobs to be the shortest of all the dispatching rules.

Average queue time

When referring to Figure B.45, it can be deduced the results of the *average queue time* KPI, that the best-performing dispatching rule was the *earliest due date*, which had the minimum value. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact that the rule generated the shortest schedule of all the dispatching rules (as illustrated in Figure B.16), which caused the *average queue time* of the jobs to be the shortest of all the dispatching rules.

Average job tardiness

When referring to Figure B.45, it can be deduced the results of the *average job tardiness* KPI, that there were no jobs that were completed after their respective due dates. The expected values therefore suggest that any of the dispatching rules could be selected if the

average job tardiness KPI was to be minimised.

Average job lateness

When referring to Figure B.45, it can be deduced the results of the *average job lateness* KPI, that the best-performing dispatching rule was the *earliest due date* rule, which had the minimum value. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact that the rule generated the shortest schedule of all the dispatching rules (as illustrated in Figure B.16), which caused the *average lateness* of the jobs to be the shortest of all the dispatching rules.

Makespan

When referring to Figure B.45, it can be deduced the results of the *makespan* KPI, that the best-performing dispatching rule was the *earliest due date* rule, which had the minimum value. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact that the rule generated the shortest schedule of all the dispatching rules (as illustrated in Figure B.16), which caused the *makespan* of the jobs to be the shortest of all the dispatching rules.

B.1.3.4 Test scenario 4 schedule selection

The results from the application of each dispatching rule in Test scenario 4 will now be discussed and the best-performing dispatching rule will be identified per KPI.

Average flow time

When referring to the confidence intervals and ANOVA table, as illustrated in Figures B.46 and B.47, of the *average flow time* KPI, it can be deduced that the best-performing dispatching rule was the *minimum slack time* rule. The ANOVA table suggests that there is statistical difference between this rule and the other dispatching rules. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact that the schedule generated by this rule, as illustrated in Figure B.25, ensured that jobs with fewer operations could be processed earlier and exit the system earlier. Therefore the *average flow time* of jobs in the system due to this rule was shorter than the other dispatching rules.

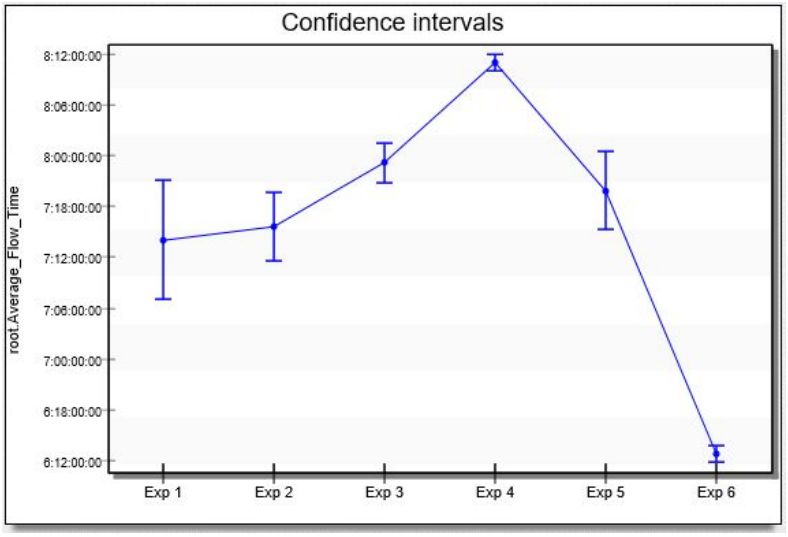


Figure B.46: Confidence intervals for the average flow time KPI for Test scenario 4

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0.703	0.016	0	0.161	0
Exp 2		0.002	0	0.156	0
Exp 3			0	0.195	0
Exp 4				0	0
Exp 5					0

Figure B.47: ANOVA results for the average flow time KPI for Test scenario 4

Average queue time

When referring to the confidence intervals and ANOVA table, as illustrated in Figures B.48 and B.49, of the *average queue time* KPI, it can be deduced that the best-performing dispatching rule was again the *minimum slack time* rule. The ANOVA table suggests that there is statistical difference between this rule and the other dispatching rules. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact that the schedule generated by this rule, as illustrated in Figure B.25, ensured that jobs with fever operations could be processed earlier and exit the system earlier. Therefore, these jobs spent less time waiting in a queue, which caused the *average queue time* of jobs in the system to be shorter for this rule than for the other dispatching rules.

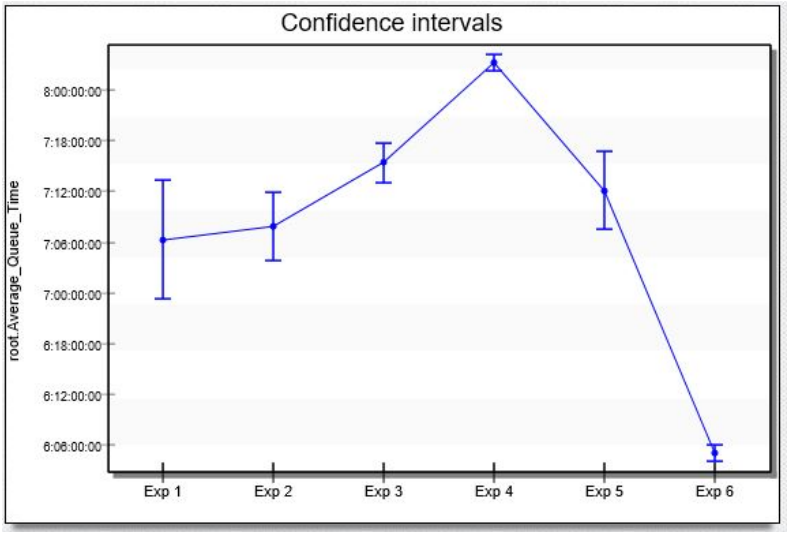


Figure B.48: Confidence intervals for the average queue time KPI for Test scenario 4

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0.703	0.016	0	0.16	0
Exp 2		0.002	0	0.156	0
Exp 3			0	0.195	0
Exp 4				0	0
Exp 5					0

Figure B.49: ANOVA results for the average queue time KPI for Test scenario 4

Average job tardiness

When referring to the confidence intervals and ANOVA table, as illustrated in Figures B.50 and B.51, of the *average job tardiness* KPI, it can be deduced that when either the *earliest due date* or *critical ratio* rule was implemented, there were no jobs that were completed after their respective due dates. The confidence intervals therefore suggest that either one of the rules can be selected when the *average job tardiness* KPI needs to be minimised. When the outcome of the experiments was analysed it could be concluded that these rules were indeed the best-performing rules. This is due to the fact both of these rules focus on the due dates of the jobs and ensure that the jobs with the earliest due date, are processed first. This will in turn then minimise the *average tardiness* of the jobs in the system. Experiments 4 and 5 do not feature in the ANOVA table, because for both experiments there were no tardy jobs and therefore no variation occurred.

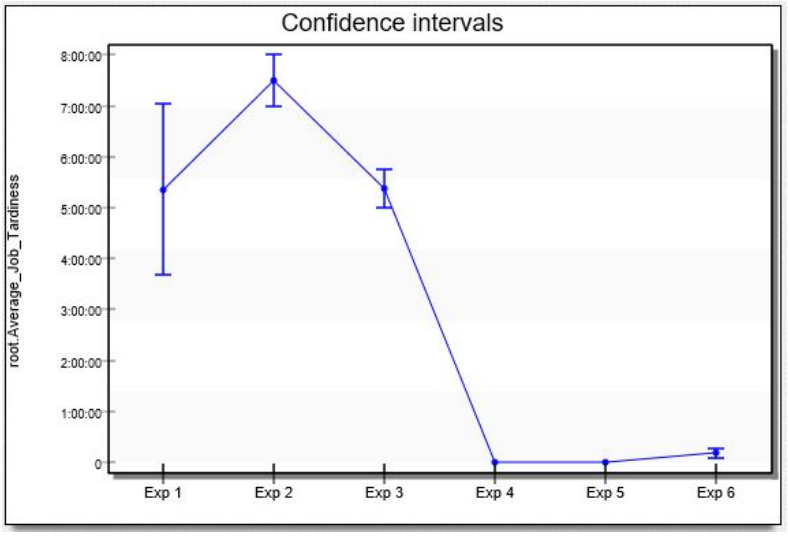


Figure B.50: Confidence intervals for the average job tardiness KPI for Test scenario 4

	Exp 2	Exp 3	Exp 6
Exp 1	0.018	0.984	0
Exp 2		0	0
Exp 3			0

Figure B.51: ANOVA results for the average job tardiness KPI for Test scenario 4

Average job lateness

When referring to the confidence intervals and ANOVA table, as illustrated in Figures B.52 and B.53, for the *average job lateness* KPI, it can be deduced that the best-performing dispatching rule was the *critical ratio* rule. When the outcome of the experiments was analysed it could be concluded that this rule was indeed the best-performing rule. This is due to the fact the rule focuses on the remaining processing time and the due dates of the jobs, which will ensure that the jobs with the earliest due date and longest remaining processing time are processed first. This will in turn then minimise the *average lateness* of the jobs in the system.

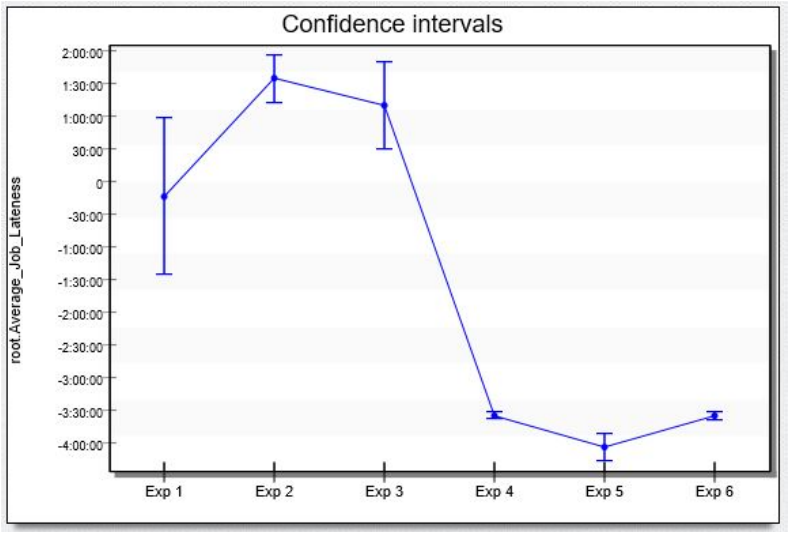


Figure B.52: Confidence intervals for the average job lateness KPI for Test scenario 4

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0.006	0.045	0	0	0
Exp 2		0.277	0	0	0
Exp 3			0	0	0
Exp 4				0	0.998
Exp 5					0

Figure B.53: ANOVA results for the average job lateness KPI for Test scenario 4

Makespan

When referring to the confidence intervals and ANOVA table, as illustrated in Figures B.54 and B.55, for the *makespan* KPI, it can be deduced that the best-performing dispatching rules were the *first-come-first-serve* and *critical ratio* rules. The ANOVA table suggests that there is no statistical difference between these two rules and therefore either of them can be selected. When the outcome of the experiments was analysed it could be concluded that these rules were indeed the best-performing rules. This is because the rules generated two of the shortest schedules of all the dispatching rules (as illustrated in Figures B.9 and B.21), which caused the *makespan* of the jobs to be the shortest of all the dispatching rules.

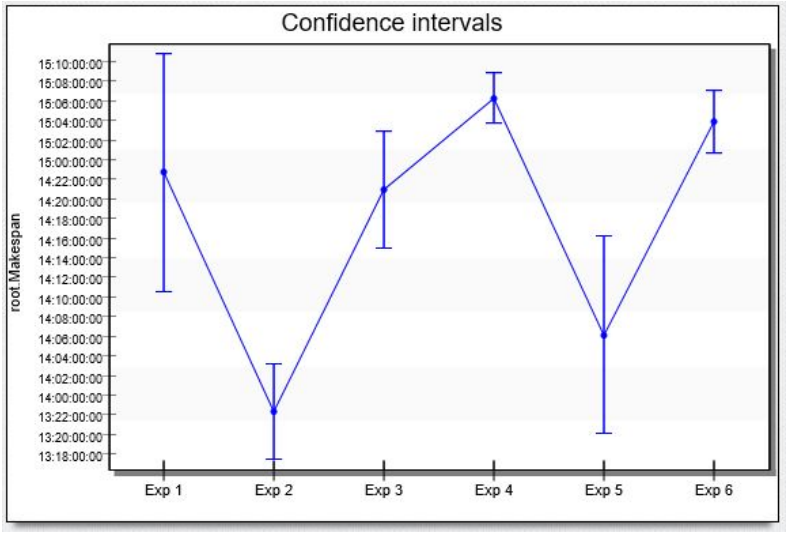


Figure B.54: Confidence intervals for the makespan KPI for Test scenario 4

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0.001	0.787	0.216	0.036	0.401
Exp 2		0	0	0.155	0
Exp 3			0.005	0.013	0.039
Exp 4				0	0.221
Exp 5					0

Figure B.55: ANOVA results for the makespan KPI for Test scenario 4

B.1.4 Synthesis

From the analysis of Test scenarios 1, 2 and 3, it became evident that the best-performing dispatching rule was almost always the *earliest due date* rule, while Test scenario 4 showed more realistic outputs. This can possibly be attributed to the small dataset of five jobs with 16 operations, that was used in the first three test scenarios. It was therefore decided to create another test scenario, called Test scenario 5, again with five jobs which have alternating expected processing times. The processing times will alternate between long processing times and short processing times, *i.e.* J_1, J_3 and J_5 will have long expected processing times, while J_2 and J_4 will have short expected processing times. This test scenario will then be run to determine if a dispatching rule other than the *earliest due date* rule could perform best. The results of this scenario will be discussed next.

Average flow time

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.56 and B.57, for the *average flow time* KPI, it can be deduced that the best-performing dispatching rule was the *most-important-job-first* rule. The ANOVA table suggests that there is statistical difference between this rule and the other rules, therefore the *most-important-job-first* rule must be chosen when the *average flow time* KPI is minimised.

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0	0	0	0	0.026
Exp 2		0	0	0	0
Exp 3			0	0	0
Exp 4				0	0
Exp 5					0

Figure B.57: ANOVA results for the average flow time KPI for Test scenario 5

Average queue time

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.58 and B.59, for the *average queue time* KPI, it can be deduced that the best-performing dispatching rule is again the *most-important-job-first* rule. The ANOVA table suggests that there is statistical difference between this rule and the other rules, therefore this rule must be chosen when minimising the *average queue time* KPI.

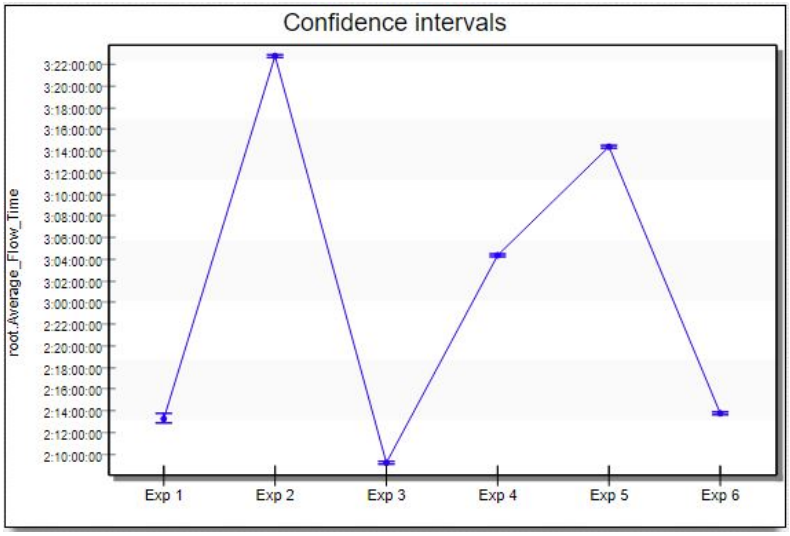


Figure B.56: Confidence intervals for the average flow time KPI for Test scenario 5

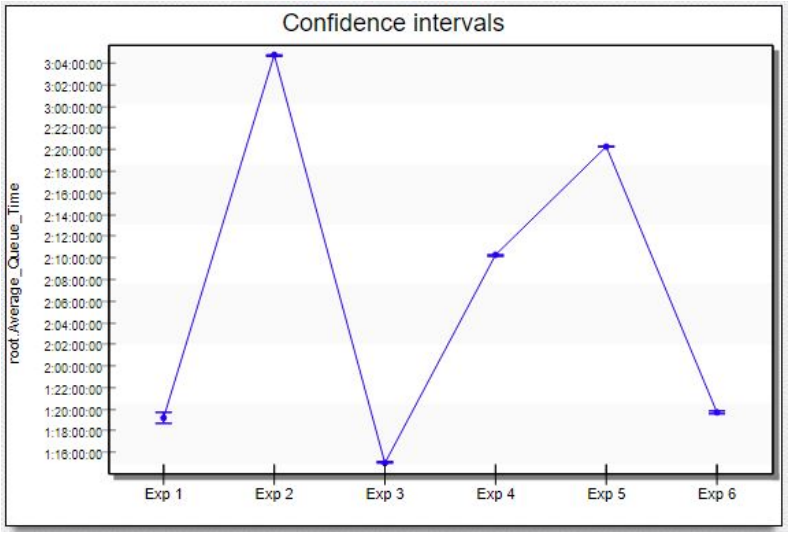


Figure B.58: Confidence intervals for the average queue time KPI for Test scenario 5

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0	0	0	0	0.034
Exp 2		0	0	0	0
Exp 3			0	0	0
Exp 4				0	0
Exp 5					0

Figure B.59: ANOVA results for the average queue time KPI for Test scenario 5

Average job tardiness

When referring to the confidence intervals, as illustrated in Figure B.60, for the *average job tardiness* KPI, it can be deduced that only one dispatching rule caused jobs to be tardy. This means that an ANOVA table could not be created to compare the outcomes of the different dispatching rules. The confidence intervals therefore suggest that any of the dispatching rules, except the *first-come-first-serve* rule, could be selected if the *average job tardiness* KPI was to be minimised.

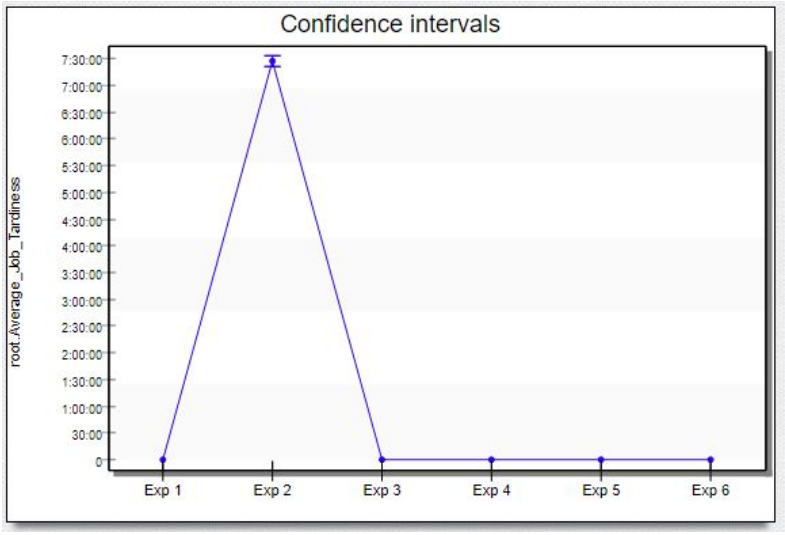


Figure B.60: Confidence intervals for the average job tardiness KPI for Test scenario 5

Average job lateness

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.61 and B.62, for the *average job lateness* KPI, it can be deduced that the best-performing dispatching rule is the *minimum slack time* rule. The ANOVA table suggests that there is statistical difference between this rule and the other rules, therefore this rule must be selected when minimising the *average job lateness* KPI.

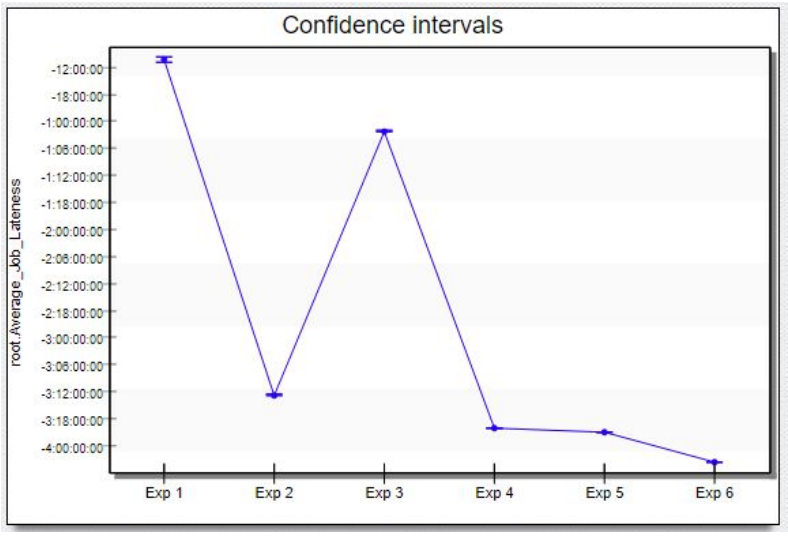


Figure B.61: Confidence intervals for the average job lateness KPI for Test scenario 5

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0	0	0	0	0
Exp 2		0	0	0	0
Exp 3			0	0	0
Exp 4				0	0
Exp 5					0

Figure B.62: ANOVA results for the average job lateness KPI for Test scenario 5

Makespan

When referring to the confidence intervals and the ANOVA table, as illustrated in Figures B.63 and B.64, for the *makespan* KPI, it can be deduced that the best-performing dispatching rule is the *minimum slack time* rule. The ANOVA table suggests that there is statistical difference between this rule and the other rules, therefore this rule must be selected when minimising the *makespan* KPI.

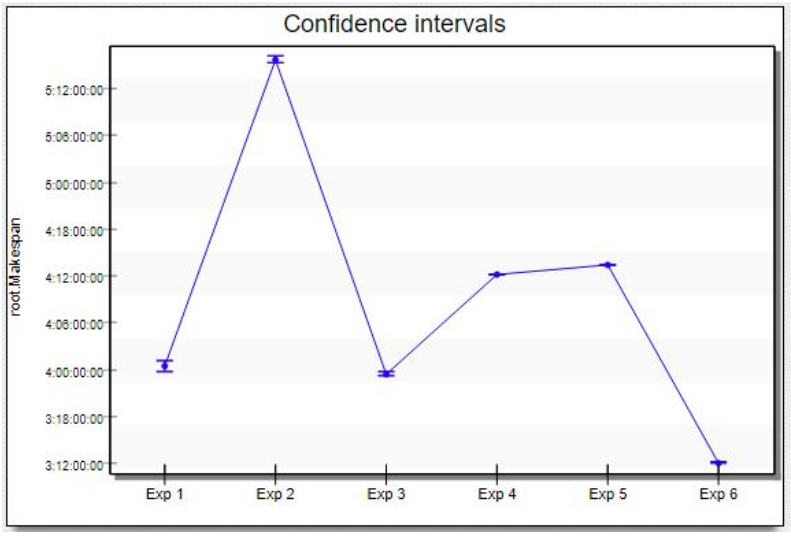


Figure B.63: Confidence intervals for the makespan KPI for Test scenario 5

	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Exp 1	0	0.006	0	0	0
Exp 2		0	0	0	0
Exp 3			0	0	0
Exp 4				0	0
Exp 5					0

Figure B.64: ANOVA results for the makespan KPI for Test scenario 5

From the results of Test scenario 5, it became evident that the *earliest due date* dispatching rule did not perform as well as some of the other rules. It also suggests that the

small dataset could have skewed the results so that the *earliest due date* rule performed best every time. However, Test scenario 5, which is the same size as the first three test scenarios in terms of the number of jobs and operations, showed more realistic output. This then proves that the scheduler is working correctly.

This concludes the validation of the scheduling process, which included dispatching rule validation and best-performing schedule selection. Next, the validation of the sensors will be discussed.

B.2 Validation of sensor operation

This section will serve to describe the validation of the sensors for the sensorised shop-floor. The validation will test whether the sensors implement the operation status changes correctly.

For the validation test, a scenario was set up where a job is logged on the information system and is linked to a specific RFID card. This job will then be allocated several operations, and the sensors will be used to log the flow of the job between the machines. The information system will be accessed after each status change that is completed by a sensor, to determine if the status change was completed correctly on the information system.

The test scenario that was created contains one job (named J_6) which has four operations. The operations that were given to the job are illustrated in Table B.24. The job has a `Job_ID` of "7" and all the operations will start with an `OperationStatus_ID` of "1" which means the operation is *Pending*. When the job arrives at the machine for its first operation (which is the machine with `MachineID` of "1"), the `OperationStatus_IDFK` should change to "2" which means that the job is *Waiting*. Therefore, the correct button on the sensor is selected and the RFID card corresponding to the job is swiped. The result of the process is illustrated in Table B.25. This process suggests that the sensor succeeded in changing the operation status of the job in the cloud-based information system. All the sensors that were developed in this project, use the same coding with the exception of the sensor number that differs. Each sensor was given its own number which should correspond with the machine number that it is used on.

The web pages that were created will also display the result of the status change to the user of the system. The change as seen by the user is illustrated in Figure B.65.

OperationStatus
Pending
Pending
Pending
Pending

(a) User view of data before operation status change

OperationStatus
Waiting
Pending
Pending
Pending

(b) User view of data after operation status change

Figure B.65: User view of the change in operation status

Operation _ID	Job _IDFK	Operation Number	Machine _IDFK	Processing Time	Earliest StartTime	Latest StartTime	Operation Status_IDFK
18	7	1	1	0:10:00	5/8/2018	5/8/2018	1
19	7	2	3	0:30:00	5/10/2018	5/11/2018	1
20	7	3	5	0:45:00	5/12/2018	5/13/2018	1
21	7	4	8	0:25:00	5/15/2018	5/17/2018	1

Table B.24: Data entered into `tblOperations` for sensor validation

Operation _ID	Job _IDFK	Operation Number	Machine _IDFK	Processing Time	Earliest StartTime	Latest StartTime	Operation Status_IDFK
18	7	1	1	0:10:00	5/8/2018	5/8/2018	2
19	7	2	3	0:30:00	5/10/2018	5/11/2018	1
20	7	3	5	0:45:00	5/12/2018	5/13/2018	1
21	7	4	8	0:25:00	5/15/2018	5/17/2018	1

Table B.25: Data in `tblOperations` after sensor test

B.3 Chapter summary

This chapter described the validation and testing of the scheduling process, as well as the sensors' operations. For the validation of the scheduling process, test scenarios were defined. The expected solution when applying each dispatching rule to the test scenario was determined and then compared to the schedule generated by the simulation scheduler. For the validation of the sensors, the sensor was tested to determine whether it could successfully and accurately change the operation status of a chosen job on the information system.

Appendix C

Operational testing

This chapter describes the operational testing of the developed system. The testing will include logging disruptions to the information system, after which the generated schedules will be analysed to determine whether the scheduler could deal with the disruption. The operational testing will also include simultaneous testing of the sensors.

C.1 Operational testing of the simulation scheduler

The operational testing of the simulation scheduler is carried out to incorporate disruptions into the system and then determine what effect they have on the schedules. The disruptions that will be tested include:

- logging a machine as *broken*,
- logging an operator as *absent*, and
- adding a new job with many operations.

For the operational testing, it was decided to use input data for a test scenario with ten jobs entered into the system. Each of the ten jobs has similar expected processing times. First, a reference schedule needed to be generated where no disruptions occurred. Therefore, the schedules that include the disruptions will be compared to the reference schedule. For explanation purposes, only one the schedule of the shortest processing time dispatching rule will be showed; however, the results of all the dispatching rules will be

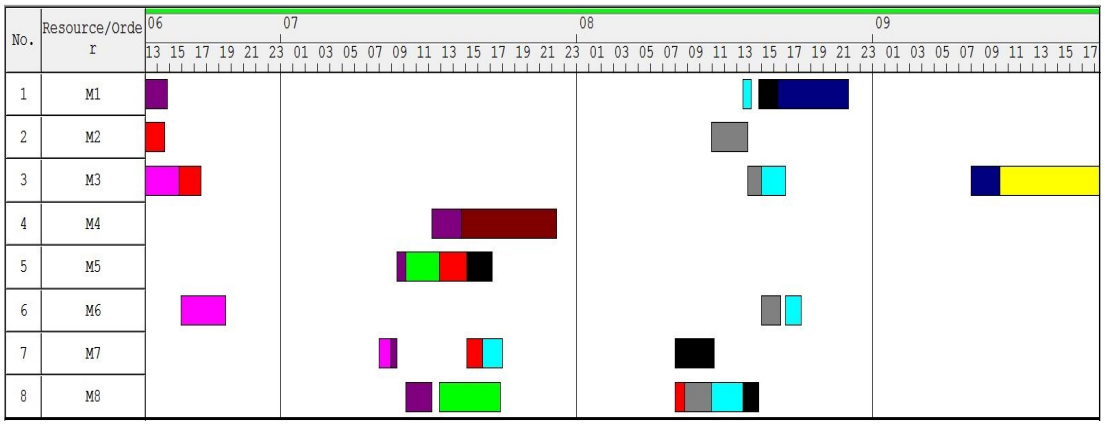


Figure C.1: Reference schedule with no disruptions (duration 3:05:27:30)

compared. The reference schedule that was generated is illustrated in Figure C.1, where the length of the schedule is also provided in *"Days:Hours:Minutes:Seconds"*. The results of this scenario, where there are no disruptions, are illustrated in Table C.1. The inclusion and discussion of the disruptions will now follow.

C.1.1 Log machine as broken

There are two parts that can be tested when logging machines as *Broken*. Firstly, how the schedule changes when a single machine fails, and secondly, how the schedule changes when identical machines fail.

When only one machine is logged as *Broken*, it is expected that the scheduler will start with that machine as unavailable and not utilise the machine before it is operational again. The jobs that were allocated to the failed machine will therefore be allocated to the identical machine, if applicable. When the machine is operational again, the scheduler should determine the outstanding jobs that were allocated to the identical machine. These jobs should then be reallocated equally between the two operational machines. If there is no identical machine, the jobs will wait in a queue in front of the failed machine. Figure C.2 is an illustration of how the reference schedule changed when a milling machine ($M3$) failed and remained broken for a period of four days, and where the length of the schedule

Table C.1: Results when no disruptions occurred

Dispatching Rule	**Average Flow Time**	**Average Queue Time**	**Average Job Tardiness**	**Average Job Lateness**	**Makespan**
Shortest processing time	2:08:42:08	2:00:55:30	0	-1:14:41:26	3:19:21:41
First-come-first-serve	2:11:35:23	2:03:48:45	0	-1:12:28:16	4:00:21:11
Most-important-job-first	2:05:16:57	1:21:30:19	0	-1:03:27:16	4:10:44:40
Earliest due date	2:14:06:28	2:06:19:50	0	-2:05:34:22	3:17:29:17
Critical ratio	2:11:26:21	2:03:39:43	0	-2:05:47:17	3:17:05:24
Minimum slack time	2:03:15:56	1:19:29:18	0	-2:03:37:52	3:17:16:59

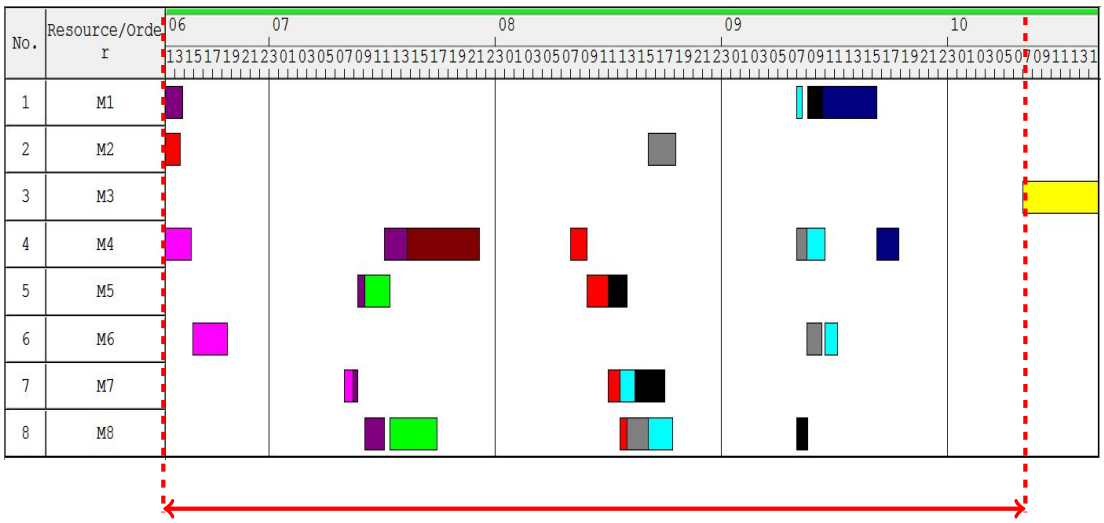


Figure C.2: Schedule generated when a milling machine was broken for four days (duration 3:01:57:30)

is also provided in *"Days:Hours:Minutes:Seconds"*. From the figure it is evident that the scheduler did not use the machine for the time it was broken, and the operations that were allocated to $M3$ shifted to the identical milling machine ($M4$). The scheduler performed as expected. Table C.2 provides the results of the different performance indicators for this disruption. From the table it is evident that the scheduler changed the schedule to accommodate the disruption, because the values are longer than those where no disruption occurred.

Table C.2: Results when a milling machine was broken for four days

Dispatching Rule	**Average Flow Time**	**Average Queue Time**	**Average Job Tardiness**	**Average Job Lateness**	**Makespan**
Shortest processing time	2:18:41:48	2:10:55:10	0	-1:14:14:57	4:11:33:21
First-come-first-serve	2:19:56:22	2:12:09:44	0	-1:05:41:23	4:12:22:17
Most-important-job-first	2:23:40:39	2:15:54:01	0	-1:00:34:25	5:12:05:40
Earliest due date	3:00:23:15	2:16:36:37	0	-2:03:51:06	4:12:35:01
Critical ratio	2:21:37:54	2:13:51:16	0	-2:04:18:15	4:09:22:06
Minimum slack time	3:00:21:13	2:16:34:35	0	-2:00:49:33	4:18:35:22

When two identical machines are logged as *Broken*, it is expected that the scheduler will start with both machines as unavailable and not utilise either machine before they are operational again. The jobs that were allocated to them will therefore start queueing in front of the two machines. Typically, both machines would not remain broken for the same length of time, therefore, when one of the machines becomes operational, the scheduler should allocate all of the jobs for both machines to the available machine. However, when the second machine also becomes operational again, the remaining jobs at the first operational machine should be divided equally between the identical machines. Figure C.3 is an illustration of how the reference schedule changed when both milling machines ($M3$ and $M4$) failed and remained broken for a period of four and three days respectively, and where the length of the schedule is also provided in *"Days:Hours:Minutes:Seconds"*. From the figure it is evident that the scheduler did not use either machine for the time they were both broken. After three days, when $M4$ became operational, it started processing operations. However, $M3$ only became operational after four days, and then it started with processing. Due to both $M3$ and $M4$ being unavailable for at least three days, the operations on the other machines also had to wait. Therefore, the scheduler did perform as expected. Table C.3 provides the results of the different performance indicators for this disruption. From the table it is evident that the scheduler changed the schedule to accommodate the disruption, because the values are longer than those where no disruption

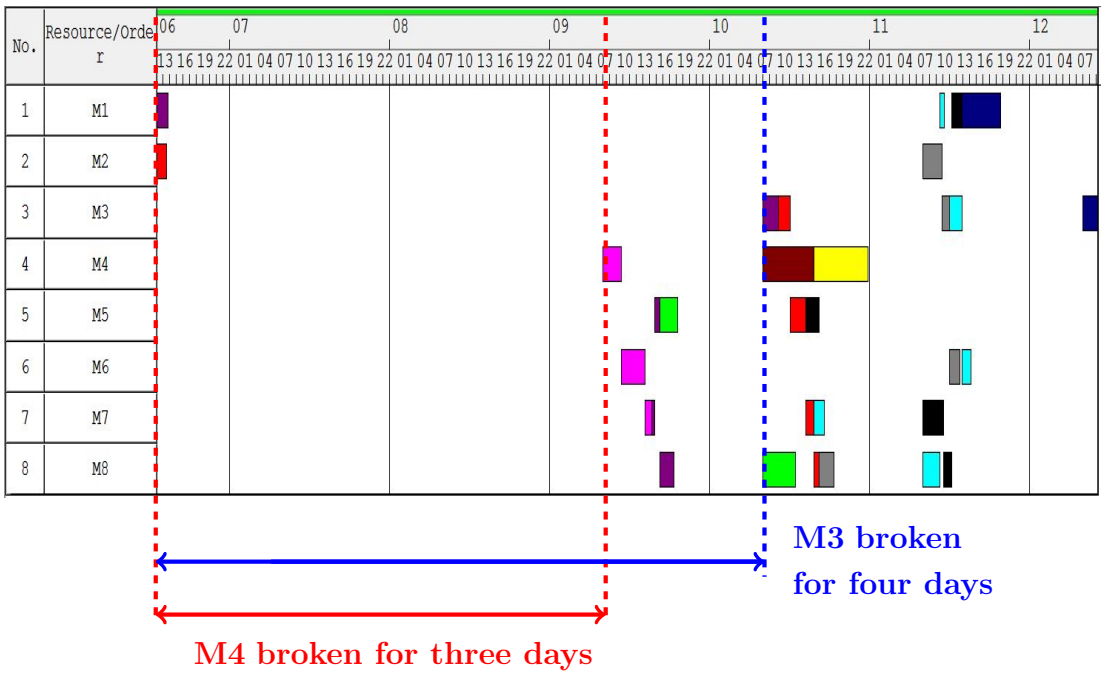


Figure C.3: Schedule generated when both turning machines failed (duration 5:21:18:51)

occurred.

Table C.3: Results when both milling machines were broken

Dispatching Rule	**Average Flow Time**	**Average Queue Time**	**Average Job Tardiness**	**Average Job Lateness**	**Makespan**
Shortest processing time	5:00:53:05	4:17:06:27	03:16	-1:09:30:54	6:16:18:46
First-come-first-serve	5:18:57:53	5:11:11:15	0	-1:04:16:09	7:02:02:40
Most-important-job-first	4:09:15:21	4:01:28:43	0	-21:19:23	6:16:42:03
Earliest due date	5:12:18:00	5:04:31:22	0	-1:22:35:37	6:17:42:04
Critical ratio	5:07:25:33	4:23:38:55	0	-1:22:15:31	6:16:53:38
Minimum slack time	5:05:33:54	4:21:47:16	0	-1:19:36:52	6:17:26:03

C.1.2 Log operator as absent

When an operator is logged as *Absent*, it is expected that the scheduler will start with the machine where the operator is stationed as unavailable and not utilise the machine before the operator is available again. The same process will apply as when the machine is broken, where the jobs allocated to that machine are moved to the identical machine, if applicable. When the operator is present again, the scheduler will divide the remaining jobs allocated to the identical machine between the two machines. If there is no identical machine, the jobs will just queue in front of the applicable machine. Figure C.4 is an illustration of how the reference schedule changed when the operator located at a grinding machine ($M6$) was absent for three days, and where the length of the schedule is also provided in *"Days:Hours:Minutes:Seconds"*. From the figure it is evident that the scheduler did not use the machine for the time the operator was absent, and the operations that were allocated to $M6$ shifted to the identical grinding machine ($M5$). In the scheduling system, the scheduler views a failed machine and an absent operator at a machine as the same instance; however, these events are logged separately for management information purposes. Table C.4 provides the results of the different performance indicators for this disruption. From the table it is evident that the scheduler changed the schedule to accommodate the disruption, because the values are longer than those where no disruption occurred.

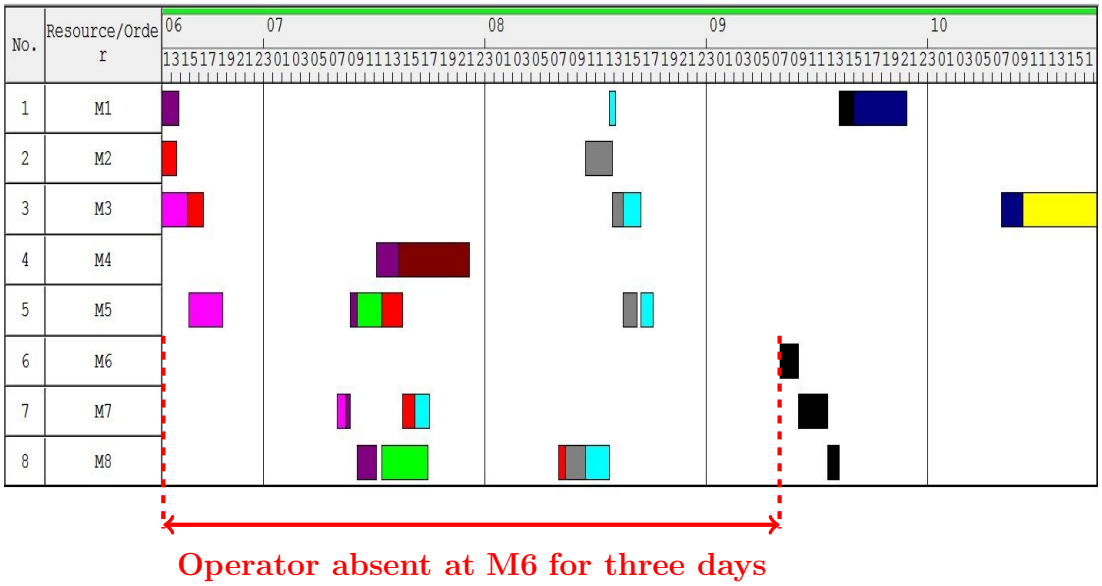


Figure C.4: Schedule generated when the operator at a grinding machine is absent for three days (duration 4:05:12:30)

Table C.4: Results when the operator at a grinding machine is absent for three days

Dispatching Rule	Average Flow Time	Average Queue Time	Average Job Tardiness	Average Job Lateness	Makespan
Shortest processing time	2:15:51:37	2:08:04:59	0	-1:13:13:14	4:08:10:33
First-come-first-serve	2:20:18:54	2:12:32:16	0	-1:10:45:43	4:15:42:47
Most-important-job-first	2:10:10:37	2:02:23:59	0	-1:02:52:37	4:14:42:08
Earliest due date	2:19:32:40	2:11:46:01	0	-2:03:50:21	4:09:18:38
Critical ratio	2:16:29:49	2:08:43:10	0	-2:04:38:39	4:01:58:42
Minimum slack time	2:10:33:03	2:02:46:25	0	-2:01:47:47	4:10:51:31

C.1.3 Add new job

When a new job with several operations is added to the information system, it is expected that the schedule generated by the scheduler will incorporate the new job into the new schedule. The job that was added has an expected processing time that is considerably longer than those of the other jobs. Figure C.5 illustrates that the new job was successfully

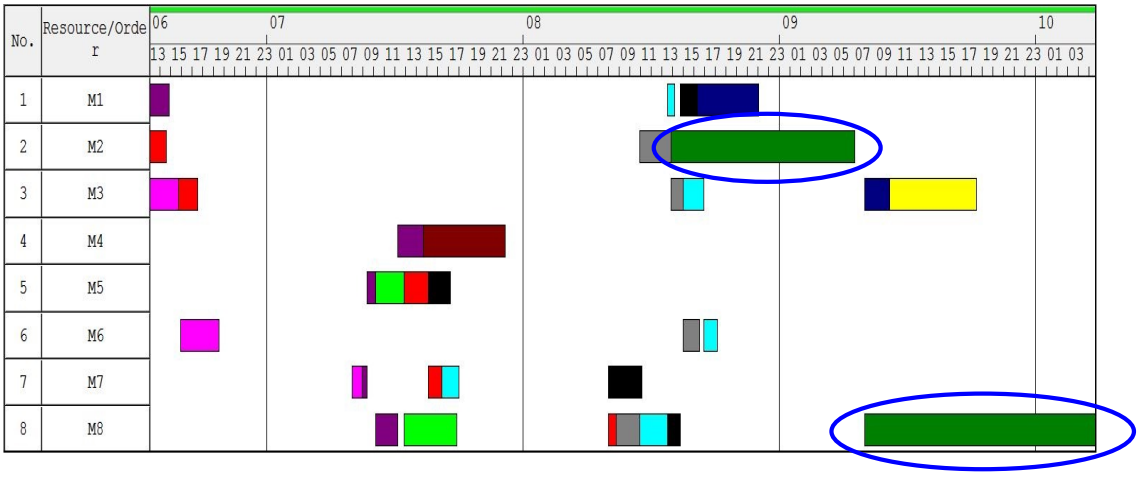


Figure C.5: Schedule generated when a new job is added (duration 3:16:48:23)

added to the schedule, where the operations of the new job are circled and the length of the schedule is provided in *"Days:Hours:Minutes:Seconds"*. Each of these disturbances that were tested proved that the scheduler adapted and generated a new schedule that incorporated each disturbance. The scheduler can therefore be assumed to be working correctly. Table C.5 provides the results of the different performance indicators for this disruption. From the table it is evident that the scheduler changed the schedule to accommodate the disruption, because the values are longer than those where no disruption occurred. This concludes the operational testing of the scheduler and the following section will discuss the operational testing of the sensors.

Table C.5: Results when a new job is added

Dispatching Rule	**Average Flow Time**	**Average Queue Time**	**Average Job Tardiness**	**Average Job Lateness**	**Makespan**
Shortest processing time	2:14:04:22	2:03:30:10	1:47:53	-1:09:02:35	4:20:12:59
First-come-first-serve	2:16:18:39	2:05:44:28	1:24:41	-1:07:28:00	4:15:31:25
Most-important-job-first	2:18:21:16	2:07:47:04	0	-22:10:08	5:10:44:40
Earliest due date	3:14:07:30	3:03:33:18	0	-1:21:11:28	5:00:17:23
Critical ratio	2:16:44:37	2:06:10:25	39:08	-1:22:24:15	4:09:09:22
Minimum slack time	3:03:21:10	2:16:46:58	0	-1:18:35:47	5:10:51:31

C.2 Operational testing of the sensors

For the operational testing of the sensors, a test data set was logged on the cloud-based information system. The test data consisted of five jobs, each with several operations. After the data set was logged, each job could be linked with a specific RFID card which will uniquely identify each job. Thereafter, the RFID cards were methodically swiped at each machine where the respective job had an operation. After each card swipe, the information system was consulted to determine whether the status change took place correctly. After

all the jobs were taken through the whole system, it could be concluded that the sensors reported all the status changes correctly.

There were also two considerations that needed to be taken into account, the first of which is that the information package sent from the different sensors should not interact or interfere with each other. Another consideration is that if an RFID card of a job is swiped at a sensor where it does not have an operation, the sensor should send the information package, but the Raspberry Pi should not log any status changes. Both these considerations were tested and the sensors performed as expected. This concludes the operational testing of the sensors.

C.3 Chapter summary

This chapter describes the operational testing of the developed system. The testing includes logging disruptions on the information system, after which the generated schedules are analysed to determine whether the scheduler could deal with the disruption. The operational testing also includes simultaneous testing of the sensors.