Integrated feedstock optimisation for multi-product polymer production

by

Marnus van Wyk

Dissertation presented for the degree of

Doctor of Philosophy

in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof JF Bekker

April 2022

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: April 2022

Abstract

A chemical complex can have multiple value chains, some of which may span across geographical locations. Decisions regarding the distribution of feedstock and intermediate feedstock to different production units can occur at different time intervals. This is highlighted as two problems, a feedstock distribution problem and an intermediate feedstock distribution problem. Unexpected events can cause an imbalanced value chain which requires timely decision-making to mitigate further adverse consequences. Scheduling methods can provide decision support during such events. The purpose of this research study is to develop an integrated decision support system which handles the two problems as a single problem and maximises profit in the value chain for hourly and daily decision-making. A high-level DSS architecture is presented that incorporates metaheuristic algorithms to generate production schedules for distribution of feedstock through the value chain. The solution evaluation process contains a balancing period to enable the application of metaheuristics to this type of problem and a novel encoding scheme is proposed for the hourly interval problem. It was found that metaheuristics algorithms can be used for this problem and integrated into the proposed decision support system.

Opsomming

'n Chemiese kompleks kan verskeie waardekettings hê, waarvan sommige oor geografiese gebiede strek. Besluite rakende die verspreiding van grondstowwe en intermediêre grondstowwe na verskillende produksie-eenhede kan op verskillende tydsintervalle plaasvind. Dit word uitgelig as twee probleme: 'n probleem met die verspreiding van grondstowwe en 'n intermediêre grondstowwe verspreidingsprobleem. Onverwagte gebeure kan 'n ongebalanseerde waardeketting veroorsaak wat tydige besluitneming benodig om verdere gevolge te versag. Beplanningsmetodes kan ondersteuning bied tydens sulke geleenthede. Die doel van hierdie navorsingstudie was om 'n geïntegreerde stelsel vir besluitnemingsondersteuning oor die twee probleme as een probleem te ontwikkel, wat wins in die waardeketting vir uurlikse en daaglikse besluitneming maksimeer. 'n Hoëvlak DSS-argitektuur word aangebied met metaheuristieke om produksieskedules vir verspreidingstowwe deur die waardeketting te genereer. Die oplossingsevalueringsproses bevat 'n balanseerperiode om die metaheuristiek op hierdie tipe probleme toe te pas, en 'n nuwe koderingskema word voorgestel vir die uurlikse intervalprobleem. Die gevolgtrekking word gemaak dat metaheuristieke vir hierdie probleem gebruik kan word en geïntegreer kan word in die voorgestelde ondersteuningsstelsel vir besluitneming.

Acknowledgements

I am grateful to the following people and institutions for their help and support during the completion of this dissertation:

- Prof. James Bekker, for his knowledge, guidance and patience throughout the study.
- Ivan Bester and Robin Jordi, for providing the high-performance computing infrastructure for this dissertation and helping me to get started.
- Holger Maul, Kobus Harmse and Simon Streicher, for the inspiration and support at various stages of this dissertation.
- The internal review committee that reviewed this dissertation and gave valuable input.
- Anne Erikson, for copy-editing this dissertation and making valuable comments.
- My parents, for their support and for encouraging me to study further.
- My wife and children, for their never-ending support, motivation and patience.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

ACO Ant colony optimisation

API Application programming interface

CD Critical distance

CEM Cross-entropy method

COW Clusters of workstations

DBMS Database management system

DRL Deep reinforcement learning

DSS Decision support system

EA Evolutionary algorithm

EERD Extended entity-relationship diagram

EWO Enterprise-wide optimisation

FF Feedforward

GA Genetic algorithm

GCF Greatest common factor

GDP Generalised disjunctive programming

GNN Graph neural network

GRASP Greedy randomised adaptive search procedure

GS Greedy search

HDPE High-density polyethylenes

HPC High-performance computing

HTH High-level teamwork hybrid

ISO International Organization for Standardization

JSON JavaScript Object Notation

LDPE Low-density polyethylene

LLDPE Linear low-density polyethylene

LS Local search

LSTM Long short-term memory

MARL Multi-agent reinforcement learning

MILP Mixed-integer linear programming

MINLP Mixed-integer non-linear programming

MLP Multilayer perceptron

MMSP Multi-product scheduling problem

MOO Multi-objective optimisation

MOOP Multi-objective optimisation problem

mSTN Maximal state-task network

NFL No free lunch

NN Neural network

NOW Networks of workstations

NSGA Non-dominated sorting genetic algorithm

NSGA-II Non-dominated sorting genetic algorithm II

NSGA-III Non-dominated sorting genetic algorithm III

OPD Object-process diagram

OPL Object-process language

OPM Object-process methodology

PBIL Population-based incremental learning

PESA Pareto envelope-based selection algorithm

PP Polypropylene

PSO Particle swarm optimisation

PVC Polyvinyl chloride

QLQ Quantity logic and quality

ReLU Rectified linear unit

RL Reinforcement learning

RNN Recurrent neural network

RTN Resources-task network

RTO Real-time optimisation

SA Simulated annealing

SAT Satellite Operations

SCO Secunda Chemicals Operations

SEC Specific energy consumption

SGA Simple genetic algorithm

SME Subject matter expert

SO Sasolburg Operations

SPEA Strength Pareto evolutionary algorithm

SPEA2 Strength Pareto evolutionary algorithm 2

SSO Secunda Synfuels Operations

STN State-task network

SUS Stochastic universal sampling

TS Tabu search

UOM Unit of measurement

UOPSS Unit-operation-port-state superstructure

VCM Vinyl chloride monomer

Chapter 1

Introduction

This chapter serves as an introduction to the research presented. It includes some background details about the organisation selected for the investigation, the problem description, research statement, objectives and information concerning the scope of the dissertation.

1.1 Background

Chemical manufacturers produce a range of products such as low-density polyethylene (LDPE), linear low-density polyethylene (LLDPE), vinyl chloride monomer (VCM) and polyvinyl chloride (PVC). The applications of these chemicals range from packaging film, plastic containers and storage bins to PVC pipes and PVC sheets. Chemical operations occur on a large scale in complex manufacturing plants or units, spread over multiple geographical locations. A regional operations location consists of several chemical plants connected by pipe networks with tanks acting as intermediate storage or as buffers. These regional operations each require supporting infrastructure to provide utilities such as *electricity*, *steam*, *fuel gas* and *utility water*. Due to the volumes involved, these utility providers will be located on site, increasing the complexity of the production ecosystem.

In chemical operations, *feedstock* is an industry term for the supply of raw material used at a chemical plant. Chemicals are produced from the hydrocarbons found in feedstock such as crude oil, coal and natural gas. Figure 1.1 illustrates a simplified *C_2 value chain*. Natural gas can be separated into C_2/C_3 rich gas which in turn, can be separated into resulting feedstocks such as ethane and propane. Cracking of these feedstocks will yield a slate of chemical feedstocks called *intermediate feedstocks*. Ethylene and propylene, being intermediate feedstocks (olefins), can be used

to produce a range of chemical products (polyolefins) which include polyethylene and polypropylene (Kannegiesser and Günther, 2011).

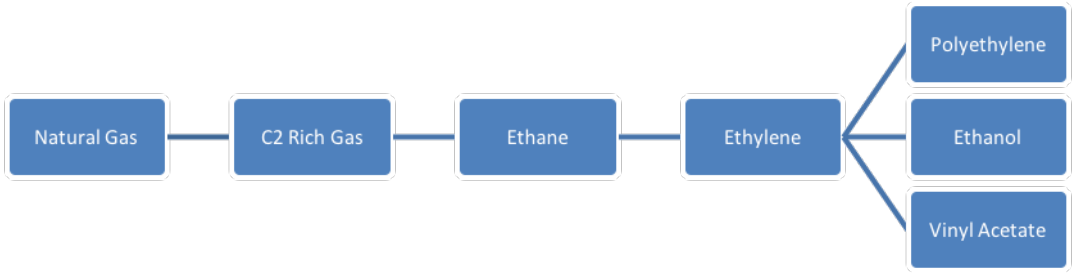


Figure 1.1: Simplified C_2 value chain

C_2 or ethane (C_2H_6), refers to the number of single-bonded carbon atoms, while the term *value chain* refers to a sequence of value-adding activities (production plants) that advance a product to the final customer (United States Department of Energy, 2018). Another such example in Sasol is the coal value chain which includes coal handling, processing and gasification (Conradie, 2007).

Customers buy ethylene, polymerised ethylene (LLDPE and LDPE or PVC) that can be used for injection moulding, blow moulding, rotational moulding, film extrusion, pipe extrusion and profile extrusion. Sasol produces 13 LLDPE grades, 14 LDPE grades and three main PVC products (Sasol, 2019).

1.1.1 Sasol Limited

As this study is carried out at Sasol, a brief background of the organisation will be given here. Sasol is an international integrated chemical and energy company, with over 30 000 employees in 32 countries that produce products in 17 chemical product groups, eight fuel and oil product groups and three gas product groups (Sasol, 2019).

Sasol – in the manufacturing sector – contributed just under 5% to the GDP of South Africa (Hurston, 2013), which was estimated at R4 trillion in 2015 (Statistics South Africa, 2015).

Sasol's Southern African operations current (2020) business model consist of four regional operating hubs, namely:

1. Secunda Synfuels Operations (SSO);
2. Secunda Chemicals Operations (SCO);
3. Sasolburg Operations (SO);
4. Satellite Operations (SAT) (Sasol, 2014a).

SCO and SO are the two chemical regional operating hubs which are located in Secunda and Sasolburg respectively; meaning the C_2 value chain is split between these two geographical points. Figure 1.2 indicates the geographical distance between these two regional operating hubs. SCO in Secunda is 136 km north-east of SO in Sasolburg.

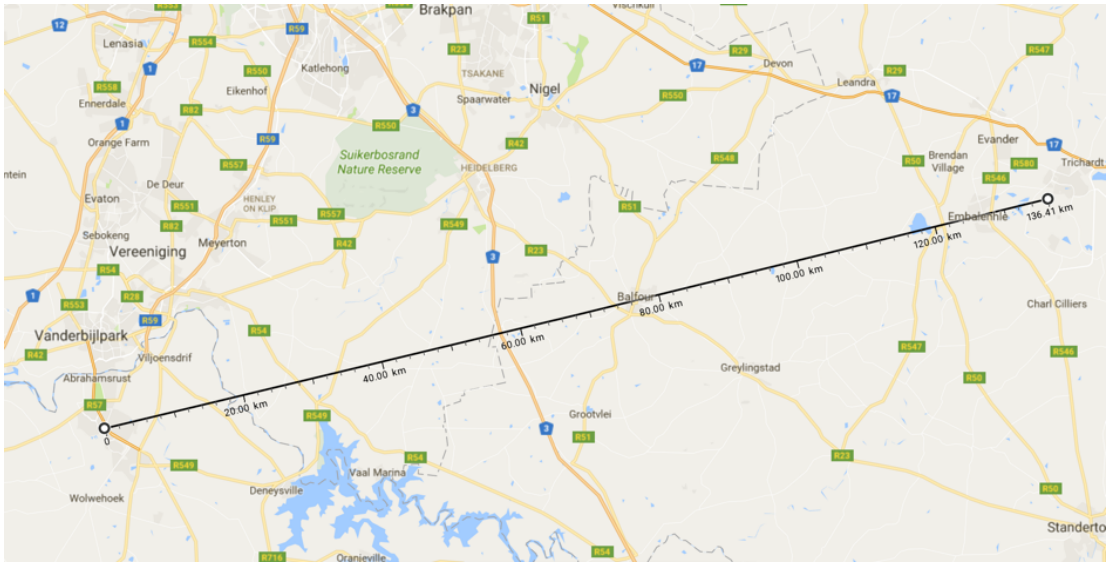


Figure 1.2: Distance between Sasolburg and Secunda (Google Maps, 2016)

Value chains can vary in complexity and larger value chains can consist of around 20 plants. A chemical complex can have multiple value chains, with some of them spanning across different geographical locations. Each region requires a management team and depending on the size and complexity of the region, multiple management teams may exist. A region may have between three and 150 plants. Decision-making in production and maintenance occurs at different time intervals. Decisions regarding the distribution of feedstock can occur on an hourly or daily time interval. Production and maintenance activities are executed according to set plans. The management teams ensure that these plans are effectively synchronised and adhered to, while communicating and aligning them with those of other regions.

1.1.2 C_2 value chain

A high-level overview of the C_2 value chain at Sasol is presented in Figure 1.3. The cryogenic distillation process at the cold separation units separates hydrogen, methane, C_2 rich gas and C_3 rich (i.e., propylene and propane) gas into different streams. The C_2 rich gas contains 60% ethylene (C_2H_4) and 40% ethane (C_2H_6). The C_2 rich gas enters the value chain from five cold separation units and is sent to

the ethylene recovery units for further processing. The ethylene recovery units ($P_{1,1}$ and $P_{1,2}$) recover the ethylene from the C_2 rich gas and compress the ethylene into the header of the ethylene pipeline. The ethane is sent to the ethane cracking units ($P_{2,1}$ and $P_{2,2}$) which has five furnaces at $P_{2,1}$ and three furnaces at $P_{2,2}$. The ethane cracking units crack ethane into ethylene and compress the ethylene into the header of the ethylene pipeline. To ensure continued operations, both ethane cracking units have ethane storage tanks ($T_{1,1}$ and $T_{1,2}$). The ethylene pipeline can be used as intermediate storage due to its length between the two regions. Ethylene can be liquefied into a storage tank ($T_{2,2}$) to balance demand from ethylene consumers. The liquefied ethylene can be vaporised slowly back into the pipeline when needed. The ethylene consumers ($P_{3,2}$, $P_{4,2}$, $P_{5,2}$, $P_{6,2}$ and $P_{3,1}$) are supplied from the pipeline and will be discussed individually (Sasol, 2014b, 2015a,b; United States Department of Energy, 2018).

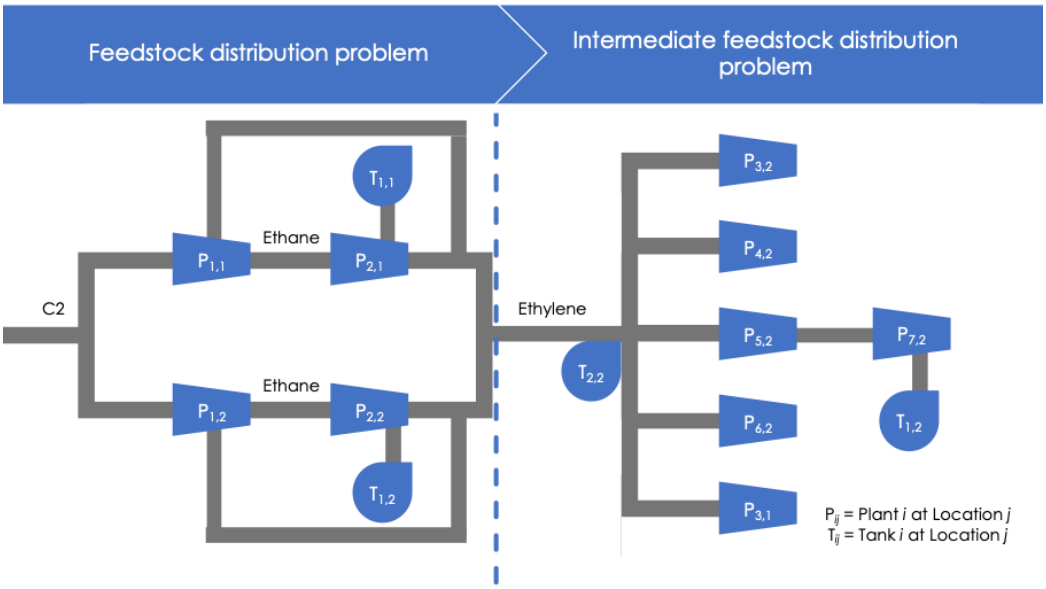


Figure 1.3: C_2 value chain

The LLDPE unit ($P_{3,2}$) manufactures linear low-density polyethylene using a fluidised bed gas reactor. The unit continuously produces 13 different LLDPE grades with sequence-dependent changeovers between grades. The product from the reactor is in a powder form which is extruded into pellets. The pellets, stored in silos, are bagged off and stored in warehouses (Sasol, 2014b, 2019).

The LDPE unit ($P_{4,2}$) manufacture low-density polyethylene using a tubular reactor. The unit continuously produces 14 different LDPE grades with sequence-dependent changeovers between grades. The product from the reactor is extruded

into pellets. The pellets, stored in silos, are bagged off and stored in warehouses (Sasol, 2014b, 2019).

The VCM unit ($P_{5,2}$) uses two different reactions to continuously produce ethylene dichloride (EDC or $C_2H_4Cl_2$), the one being a direct chlorination process and the other an oxyhydrochlorination process. The EDC is cracked into vinyl chloride monomer (VCM or C_2H_3Cl) and sent to the polyvinyl chloride (PVC or $(C_2H_3Cl)_n$) unit. The PVC unit ($P_{7,2}$) uses a suspension polymerisation batch process to manufacture three types of PVC products. Once the PVC is separated, it is dried, bagged off and stored in warehouses. The PVC unit has VCM storage tanks ($T_{3,1}$) that act as intermediate storage between the VCM and the PVC units (Sasol, 2014b, 2019).

Safripol ($P_{6,2}$), a customer of Sasol, purchases ethylene to manufacture high-density polyethylene (HDPE) and polypropylene (PP). A fixed-ratio ethylene supply contract between Sasol and Safripol is in place (Du Plessis, 2010).

PP units ($P_{3,1}$), manufacture 12 grades of homopolymers and eight grades copolymers and uses low volumes of ethylene from the pipeline (Sasol, 2015a, 2019).

The ethylene consumers' capacity exceeds the ethylene production capacity and requires critical decision-making to ensure customers' contractual obligations are met while converting ethylene into the maximum saleable product.

Management relies heavily on subject matter experts (SMEs) to improve the value chains and regions on an operational, tactical and strategic level. SMEs use a range of models to optimise certain sections of a value chain or region. These models are mostly single-objective multi-period linear and non-linear models and include Microsoft Excel-based models, mathematical programming models, stochastic programming models and simulations. Several software packages have been implemented to assist with these time-sensitive production scheduling decisions and production planning decision. These systems proved to be inflexible and inconsistent under certain operational circumstances with most SMEs reverting to the original Microsoft Excel-based models.

1.2 Problem description

The feedstock distribution of the value chain or region is affected by the feedstock it receives, feedstock in storage, the maintenance activities of individual plants and the day-to-day operations of individual plants to maximise production, minimise losses and balance the utilities' demand.

The intermediate feedstock distribution to plants producing saleable products is affected in the same way as the feedstock distribution. Although this is common in

value chain configurations, the algorithms should be developed for each value chain configuration.

The plants producing final products schedule the production to meet the demand of the orders with the amount of feedstock allocated to them. The schedule can be affected by maintenance activities, the day-to-day operations and the sequence in which the products or grades need to be produced. These plants are sequence-dependent multi-product batch plants or sequence-dependent multi-grade continuous plants. Tanks, pipelines and silos or *intermediate storage* reduce distribution complexity and increase flexibility.

Various scenarios question the operating philosophy of steady-state operation. Examples of these could be:

1. What is the most profitable schedule for the next 90 days?
2. What is the most profitable schedule for the next 14 days?
3. How should the feedstock be distributed in the next 90 days when a unit has unplanned downtime?
4. How should the feedstock be distributed in the next 14 days when a unit has unplanned downtime?
5. What is the trade-off between the profitability of the schedule and energy consumption?

Each activity in the value chain has a cost and energy consumption associated with it, while the manufactured product contributes towards sales. The profit of the value chain can be calculated by subtracting the cost of manufacturing from sales. Two decisions can have different costs associated with them and the profit indicates which decision will have a lesser impact on profitability. Subsequently, the two decisions can have different energy requirements that could influence the final decision. This will be discussed in detail in a future chapter.

The problem description can be separated into two problems and is illustrated in Figure 1.4,

1. a feedstock *distribution* problem;
2. an intermediate feedstock *distribution* problem.

Figure 1.4: Two problems in the problem description

The two problems are similar and these problems have been addressed individually within Sasol in various ways and with varying degrees of success, from manual slot allocation by schedulers to SMEs that monitor, model and advise management. They do not currently integrate with other models and are not easily scalable to include other models. The time horizons play a significant role. An upset in the value chain requires time-sensitive decision-making with hourly oversight on the production schedule while the production plan is based on a daily production rate for three months. Modelling the two horizons proves to be difficult as the complexity increases linearly. Using the same method or model for both time horizons will increase the alignment between the decision-making on an hourly or daily basis. Figure 1.5 illustrates the scale of the time horizon for the two problems.

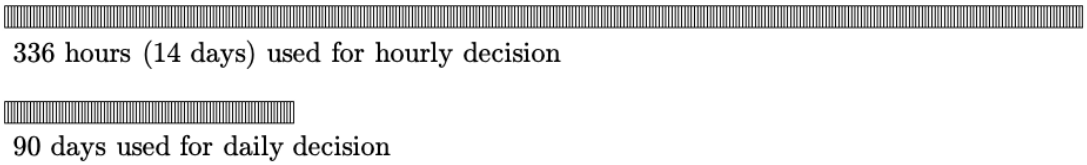


Figure 1.5: Size of the time horizons

This raised the *need for a decision support system (DSS) that integrates the distribution problems for the two time horizons.* The research question is to determine the required architecture and methods to realise such a DSS. The research task is thus as follows:

Develop an integrated decision support system that maximises profit in the Sasol C_2 value chain for hourly and daily decision-making.

It is important to define the term decision support system. The Oxford English Dictionary (2011) defines a decision support system as: "A computer program or other system used to aid in decision-making.".

1.3 Objectives

To address the research task, the following objectives were set:

1. *Acquire* knowledge in the scientific literature related to the problem statement;

2. *Research* various time representations that can address the two problems;
3. *Acquire* knowledge on single-objective and multi-objective optimisation by reviewing scientific literature;
4. *Select* single-objective algorithms and acquire specific knowledge on the selected algorithms;
5. *Select* a bi-objective optimisation algorithm and acquire specific knowledge on the selected algorithm;
6. *Construct* the selected single-objective algorithms to find near-optimal distribution for the feedstock and intermediate feedstock distribution problems;
7. *Construct* the selected bi-objective algorithm to find Pareto optimal solutions for the feedstock and intermediate feedstock distribution problems;
8. *Construct* a DSS for integrated near-optimisation by the algorithms;
9. *Develop* a web-based system to provide an interface to the algorithms;
10. *Verify* the code and *evaluate* the DSS;
11. *Conduct* experiments on different scenarios as previously stated:

    What is the most profitable schedule for the next 90 days?

    What is the most profitable schedule for the next 14 days?

    How should the feedstock be distributed in the next 90 days when a unit has unplanned downtime?

    How should the feedstock be distributed in the next 14 days when a unit has unplanned downtime?

    What is the trade-off between the profitability of the schedule and energy consumption?
12. *Analyse* and *synthesise* the results and provide recommendations for future work.

1.4 Scope

The scope of this dissertation is limited to the two problems indicated in Figure 1.4 in the problem description. Safripol's inclusion will be limited to the ethylene supply contract. The PP units form part of the C_3 value chain and only ethylene consumption will be considered in the dissertation.

No proprietary information or data will be disclosed and all process data collected from historians will be normalised. Only information available in the public domain will be included in the dissertation.

Margins are derived from public data listed in Appendix A and operating limits estimated from production capacities available in the public domain (Sasol, 2015c).

The current-state data can be entered manually or collected automatically from the historian. Manual data entry can be used to test specific scenarios. The collected data will be stored in a separate database on the webserver. The algorithms will run on the webserver and the output will be published on a website.

Since the study focuses on a complex, practical problem, it is anticipated that the solutions by the DSS can only claim to be satisfactory or near-optimal at best. Throughout the dissertation, the researcher will accept results as being near-optimal.

1.5 Research methodology

To fulfil the objectives of this dissertation, the proposed research methodology is described here.

In fulfilment of **Objectives 1** and **2**, a detailed literature review will be conducted to acquire knowledge on the parts of the two problems and the two time horizons. Discussions with subject matter experts at Sasol will unify literature with expert knowledge. This includes managers, engineers, accountants and other knowledgeable employees related to the field of study.

A review of the scientific literature, focused on single-objective optimisation and multi-objective optimisation in **Objective 3**, will provide the understanding to select single-objective algorithms in **Objective 4** and a bi-objective algorithm in **Objective 5**. To complete **Objective 4** and **Objective 5**, a study will be done on existing literature to acquire a comprehensive understanding of the selected algorithms.

Literature will be consulted to aid in selecting an encoding scheme, objective function and constraint strategies, in partial fulfilment of **Objective 6** and **Objective 7**. The algorithms used in this dissertation will be constructed from pseudo code to complete **Objective 6** and **Objective 7**.

The algorithms constructed in **Objective 6** and **Objective 7** will be combined into an integrated decision support system to fulfil **Objective 8**.

A web-based system will be developed to fulfil **Objective 9**, allowing engineers to find near-optimal solutions on the distribution of feedstock in the C_2 value chain using the algorithms constructed.

Tests will be done to verify the code used in this study and SMEs will evaluate DSS to achieve **Objective 10**.

To fulfil **Objective 11**, the algorithms will be executed in different scenarios and compared. The results will be analysed with subject matter experts at Sasol.

The results will be reflected on and future work recommendations will be made to satisfy **Objective 12**.

1.6 Structure of the document

The document is structured as follows.

Chapter 1 – ***Introduction***

This chapter introduces Sasol and the complexity around chemical operations. The problem description is discussed. The research methodology is then addressed and used as the basis of the document structure.

Chapter 2 – ***Scheduling literature review***

This chapter begins with a short introduction to scheduling, followed by a discussion of the scheduling literature on similar problems in the process industry. The chapter will end with a discussion of literature on time representation.

Chapter 3 – ***Optimisation algorithms: selected literature review***

The chapter begins with important aspects of single-objective optimisation and a review of a number of algorithms, followed by a brief introduction to multi-objective optimisation and a review of the selected algorithms. The chapter concludes with a section on hybrid metaheuristics with specific focus on classification, grammar, parallel metaheuristics and machine learning.

Chapter 4 – ***Construction and implementation of the algorithms***

This chapter describes the construction and implementation of the algorithms, starting with the variables used in the study and the encoding scheme implementation. This is followed by a description on the evaluation process required for the problem with constraint handling, balancing the variables and calculating the objective function highlighted. Finally, there is a discussion of the hybrid and parallel algorithms implemented for this study.

Chapter 5 – *System design and implementation*

The proposed DSS architecture is presented in this chapter with how a subject matter expert will interface with the system.

Chapter 6 – *Verification and evaluation*

In this chapter, the verification and evaluation of the decision support system is presented. The chapter starts by specifying the tests that are needed to verify the code used in this study. The performance of the algorithms is then compared with the others in relation to the questions listed earlier in Chapter 1. An analysis of the parameters used with the best performing algorithms is then presented, followed by the results from parameter tuning. Thereafter, an evaluation of the DSS by subject matter experts and subsequently, an experiment based on a recommendation from the SMEs concludes the chapter.

Chapter 7 – *Conclusion and recommendations*

The chapter discusses the conclusion arrived at from the study and suggests possibilities for future work. A summary on the work covered in the study is presented and how the work aligned with the scope and objective in **Chapter 1**.

1.7 Chapter summary

A background was given on chemical manufacturing with the focus on Sasol. The C_2 value chain concept was introduced with the complexities of the production ecosystem. The difficulties with decision-making on the value chain was described and that introduced the research task. Objectives that were set to achieve the research task were presented and a proposed research methodology to achieve the objectives was introduced. The scope of the dissertation was presented and the structure of the remainder of the document was set out. In the following two chapters, a review of literature relevant to the study will be discussed, commencing with a review of literature relevant to scheduling in the process industry in **Chapter 2**.

Chapter 2

Scheduling literature review

Chapter 1 served as an introduction to the background and problem being addressed in this dissertation. This chapter starts with a short introduction to scheduling by highlighting the classification to identify different scheduling problems. To achieve Objective 1, a review is carried out of literature in the process industry that is relevant to the feedstock distribution problems in Figure 2.1. Castro *et al.* (2018) identified major challenges with scheduling in the process industry and two of the challenges that apply to this study are then discussed. Finally, decisions in different time horizons are introduced into the problem description and literature on time representation is reviewed to achieve Objective 2.

Figure 2.1: Two problems in the problem description

2.1 Classification of scheduling

Production scheduling has been researched extensively, with historical surveys and reviews of papers spanning over decades. Graves (1981) proposed a broad classification to production scheduling problems, namely:

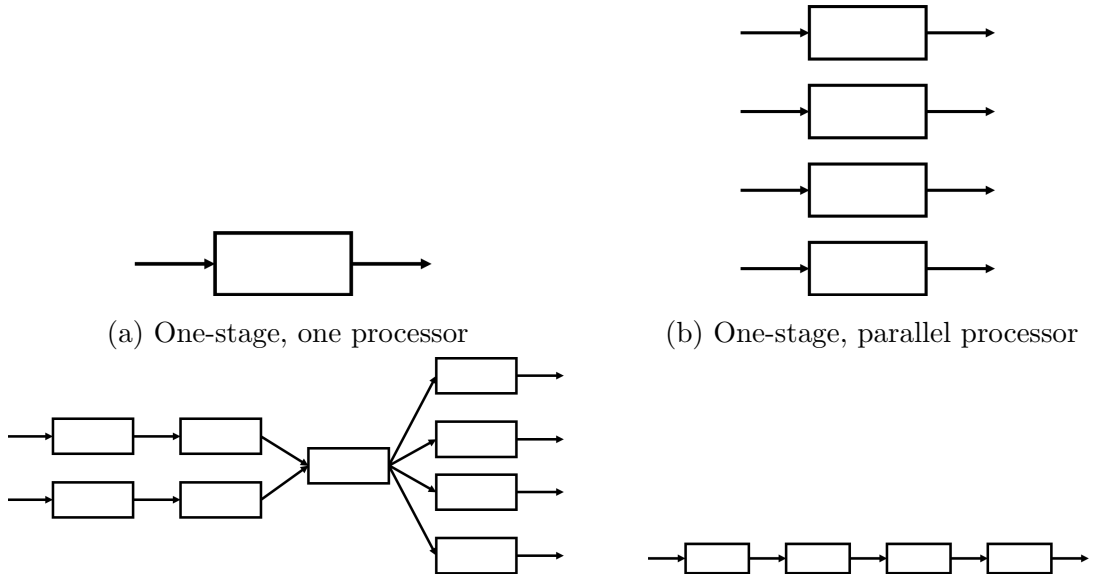
1. Requirements generation;
2. Processing complexity;
3. Scheduling criteria.

The first classification is in terms of a *closed shop* or *open shop*. In a closed shop, production is forecast to meet demand and orders are replenished from inventory. In

an open shop, no inventory is kept and production batches are equal to the ordered quantity. Most environments are combinations of open and closed shops and pure open or pure closed environments are rare.

The second classification is in terms of production processing complexity or production processing structure. The processing complexity plays a defining role in the production scheduling problem and can be categorised as follows:

1. One-stage, one processor, illustrated in Figure 2.2a;
2. One-stage, parallel processor, illustrated in Figure 2.2b;
3. Multi-stage, flow shop, illustrated in Figure 2.2c;
4. Multi-stage, job shop, illustrated in Figure 2.2d.



(a) One-stage, one processor

(b) One-stage, parallel processor

(c) Multi-stage, flow shop

(d) Multi-stage, job shop

Figure 2.2: Processing structure

The final classification is in terms of schedule performance and schedule cost. Schedule performance is primarily focused on the measures that can reduce time. Common measures can include minimising the makespan of producing the product slate or minimising the time between the completion date and the due date of the order. Schedule cost is primarily focused on the economical measures of a schedule. Common measures can include minimising the total operating cost of production or minimising the cost of changeovers. It is not uncommon to use a combination of both to evaluate the schedule (Graves, 1981).

2.2 Scheduling in the process industry

The classification has since been expanded to describe the complex configurations found in process scheduling, this includes but is not limited to multi-plant, multipurpose, multi-supplier, multilevel, multi-product, multigrade and sequence-dependent. Some examples of work found in the literature are discussed next, in the context of this study.

Maravelias (2012) proposed a classification for chemical production scheduling. The term 'production environment' is proposed and can have three processing types namely: sequential, network, and hybrid. With network processing, two popular representations are state-task network (STN) and resources-task network (RTN).

The researcher could not find a comprehensive classification for scheduling in the process industry in literature (Harjunkoski *et al.*, 2014; Kallrath, 2002; Maravelias, 2012). Castro *et al.* (2018) highlight that scheduling in the process industry is relativity new, having started in 1978. A chronological review of literature relevant to the feedstock distribution problems in Figure 2.1 will now follow.

Bell (1980) presents a two-stage ethylene production process model to address a decoupling inventory problem with storage capacity constraints. Being an inventory problem, the model will determine the optimum storage capacity and the value of purchasing additional feedstock. This production process is similar to the *intermediate feedstock distribution* problem in Figure 2.1. Further work on the problem introduced a *balance period* at the end of each period (Bell, 1983). The balance period gave a degree of flexibility in the short run by varying the pressures in the connecting pipes before liquefying into storage or evaporating out of storage. Bell *et al.* (1990) labelled the problem as "the international polymer problem".

In 1988, a survey in a wide range of industries including the manufacturing of polymers concluded that the requirements for production scheduling across all the industries researched were similar in their process structures, product sequencing constraints and scheduling objectives (Musier and Evans, 1989).

In 1997, Reisman *et al.* (1997) reviewed 184 flow shop scheduling papers and found that less than 3% dealt with realistic production environments. This is consistent with the gap identified by Maccarthy and Liu (1993) that scheduling models defined in literature do not always apply in practice due to unique systemic complexities.

Kallrath (2002), gives an overview of the state-of-the-art planning and scheduling problems in the process industry. The author highlights special features and concepts relevant to the process industry and briefly discusses metaheuristics. The study recommends using state-of-the-art technology based on mixed-integer optimisation

for solving real-world planning problems and concedes that for scheduling, there is no commonly accepted state-of-the-art approach and most are still based on heuristics.

Jackson *et al.* (2003) proposed multi-period non-linear programming formulation for production planning of a multi-plant polymers site. The formulation includes non-linear process models to either meet demand or maximise profit. Due to the size of the model, decomposition methods must be used to expand the model further.

Wassick (2009), details a world-scale integrated chemical production complex and discusses the challenges with planning and scheduling. He refers to the integrated site as a localised multi-echelon supply chain and presents a solution using a discrete-time RTN model for waste disposal scheduling.

Zyngier and Kelly (2012) proposed a paradigm "unit-operation-port-state superstructure (UOPSS)" for modelling advanced planning and scheduling systems that has a more natural and simpler ability to account for problems. The variables can be categorised as quantity logic and quality (QLQ) in the model and can be applied for both batch and continuous processes.

- Quantity – Rate, yield, flow, pressure, temperature, batch or lot-size.
- Logic – Startup, shutdown, setup or switchover.
- Quality – Density, ratio, components, properties, conditions.

From 2011 to 2016, Marchetti *et al.* (2013, 2016) studied the C_3 feedstock distribution in a polypropylene production facility. A single and multi-product, multi-period problem was presented with the objective to maximise the overall profit while satisfying the constraints. Three non-linear models were formulated and for the multi-product formulation, medium- and long-term test cases were presented. In the test cases, it was found that reducing the production cost by means of lower production rates was more profitable. The C_3 feedstock distribution study focused on a similar value chain to that in this study with the difference of it being a different configuration, on C_3 and not C_2, and it included the scheduling of production with the constraint of a sequenced production wheel.

Harjunkoski *et al.* (2014) did an extensive review of the existing scheduling models and methods developed for process industries. The authors highlighted six major types of modelling; namely, scheduling algorithms (exact or heuristic), metaheuristics such as evolutionary algorithms, timed automata, integrated modelling and solution methods such as constraint programming, mathematical programming, and hybrid methods. Metaheuristics provide good solutions in a reasonable amount of time while satisfying the constraints. The difficulty is with representing the constraints

and the use of constraint handling strategies. They stated that metaheuristics have attractive features over exact and deterministic methods. These include features and constraints that can be modelled more easily and using less variables than in exact and deterministic methods. Some of the more difficult issues are the representation of constraints. Constraint handling strategies can be used and should be carefully structured so as not to inhibit the search for solutions.

Brunaud *et al.* (2020) compared four main modelling frameworks, STN, maximal state-task network (mSTN), RTN, and UOPSS with the focus on quality-based changeovers. In general, the UOPSS outperformed the other frameworks and the formulation was found to be easier to implement, extend, and scale.

Castro *et al.* (2018) reviewed generalised disjunctive programming (GDP) as the current state of the art as a modelling approach to process scheduling. GDP can combine both discrete and continuous logic using Boolean variables. The author also showed two other approaches, STN and RTN that have matured in discrete-time and continuous-time process scheduling. They lists four major challenges in the scheduling area. Two of these, which are relevant to this dissertation, will be discussed.

The *first challenge* is coordination planning and scheduling models over different periods; short term (minutes to days), medium term (months) and long term (years) for sites that are spread over multiple geographical locations. This challenge could be extended further down to the control domain where the time frame is even shorter, down to seconds and minutes. This challenge could also be extended horizontally to include all upstream and downstream units. This research field has been labelled as enterprise-wide optimisation (EWO) with the activities planning, scheduling, real-time optimisation (RTO) and control included (Grossmann, 2012). The full scope of the challenge is not currently possible with mixed-integer linear programming (MILP) and mixed-integer non-linear programming (MINLP) models with most research focused on a subset of the scope (Castro *et al.*, 2018; Grossmann, 2005, 2012; Grossmann and Furman, 2009; Guillén-Gosálbez and Grossmann, 2008; Marchetti *et al.*, 2013, 2016; Wassick, 2009). The problem in this study is multi-period and the decision-making is both in the short term, hours and days. The *time representation* can increase the number of variables required and making decisions with different time intervals requires a different approach. This will be discussed in the next section.

The *second challenge* is linked to the first, to solve large-scale MILP and MINLP models effectively with efficient algorithms with modern computer architectures. Scheduling methods has evolved over the last 40 years and so have computers. Exact methods have yet to address the first challenge with modern computer architecture. Most production scheduling problems are stochastic and dynamic in nature

while being *NP*-complete (Graves, 1981; Musier and Evans, 1989). This increases the complexity of developing mathematical models while expecting optimal results. Approximation methods can be used as an alternative to finding near-optimal results in polynomial time for large industrial-sized problems. Depending on the accuracy required, the solutions generated from metaheuristics are generally acceptable.

According to Baumann and Trautmann (2014), the performance of MILP with large-scale problems is insufficient and the hybrid methods increase performance. The next section will focus on scheduling literature in the process industry but not on the feedstock distribution problems in Figure 2.1, to highlight algorithms used.

2.3 Optimisation methods used for scheduling in the process industry

Since production scheduling problems are often stochastic and dynamic in nature while being *NP*-complete, approximation methods can be used as an alternative to finding near-optimal solutions. Optimisation methods used for scheduling in the process industry, with relevance to the second challenge, will now be discussed.

Wang *et al.* (2000) compared a genetic algorithm (GA) to an MINLP algorithm with different horizons in the polymer industry. The MINLP performs slightly better up to a ten-day horizon, whereafter the GA performs better. The results of the GA are comparable with the MINLP. The problem contained a batch reaction process with the final processing being continuous. Mathematical modelling can lead to non-convex, large MINLP. The study highlighted two points, firstly, it is possible to combine batch and continuous processes and secondly, GAs can perform better with larger problems and are easier to formulate.

Sadegheih (2006) used a GA and simulated annealing (SA) to optimise production schedules for a flow shop scheduling problem. The SA needed longer computation times to achieve the same results as the GA. The conclusion highlighted that scenario particulars can be embedded in the objective function without impacting the optimisation routine. The expertise of human schedulers can be accounted for as rules to mimic reality.

In 2007, a review of literature on production scheduling with neural networks found 18 multilayer perceptron (MLP) neural networks applied to problems such as single-machine scheduling, job-shop scheduling, batch processes and flexible manufacturing. Akyol and Bayhan (2007) also listed Hopfield-type networks, competitive-type networks and hybrid approaches. They recommended using evolutionary algorithms with neural networks to improve flexibility and make them more effective.

He and Hui (2007) applied a GA to a large-size multi-product scheduling problem (MMSP). MILP solvers can find an optimal solution in a short period of time but with large-size problems, heuristic techniques are used to reduce the size.

Ramteke and Srinivasan (2011) proposed a real-coded chromosome multi-objective GA for short-term scheduling at polymer plants. The GA found near-optimal solutions to three different scheduling problems with sequencing constraints. The author states the disadvantages of using mathematical programming for both discrete-time and continuous-time formulations of short-term scheduling and proposes evolutionary algorithms, specifically GAs, to address the disadvantages.

Martínez Jiménez (2012) addressed six scheduling problems with different complexities and constraints with multi-agent reinforcement learning (MARL). The results achieved with the MARL did not outperform the GA on solution quality but on larger problems it completed the execution in a much shorter time.

Liu *et al.* (2016) used a non-dominated sorting genetic algorithm II (NSGA-II) to minimise two objectives, namely tardiness and total electricity consumption, on a scheduling problem. They further extended the NSGA-II with two additional steps to reduce the total non-processing electricity consumption more effectively.

Zhang *et al.* (2020) applied an NSGA-II to minimise three objectives, namely makespan, total energy consumption, and peak input power. The authors proposed a hierarchical multi-strategy genetic algorithm based on a non-dominated sorting method that reduced the energy consumption by approximately 15%.

Hubbs *et al.* (2020) applied deep reinforcement learning (DRL) on a production scheduling problem with a two-stage continuous chemical reactor and a packaging line and found the results outperformed the human schedulers.

Deterministic approaches guarantee the global optimality while metaheuristics like tabu search (TS) cannot. Lin and Miller (2004) applied TS to eight problems found in literature that were solved using MILP and MINLP. The author performed 100 runs on each of the problems and located the global optimum for one of the problems. For the other problems, the results came within 0,1% of the global optimum and the rest within 1%. By adjusting the parameters, the author managed to decrease the execution time and improve the quality of the solutions.

This section addressed literature related to feedstock distribution as shown in Figure 2.1 and the use of metaheuristics or hybrid metaheuristics for scheduling in the process industry. In the next section, the time representation from the first challenge will be discussed.

2.4 Time representation

There are generally two approaches with time-based scheduling; precedence-based and time-grid-based. Precedence-based models are concerned with the sequencing of certain tasks or batches. For time-grid-based models, Velez and Maravelias (2013) proposed the following classification; continuous-time and discrete-time, single- (common) and multiple-grid models and lastly uniform and non-uniform.

2.4.1 Continuous-time and discrete-time

With the discrete-time approach, the time horizon is divided equally into the greatest common factor (GCF) intervals. In real-world problems, the number of intervals become a multiplier for the number of variables needed, which lead to large combinatorial problems. With the continuous-time approach, the events take place in the domain of time; this flexibility allows for varying events on different levels. The continuous-time approach can lead to more complicated mathematical models compared to similar discrete-time models. Sundaramoorthy and Maravelias (2011) presented a critical review of discrete-time and continuous-time formulations and refuted beliefs held in the process systems engineering literature that computational performance, solution robustness, solution quality and generality are superior with continuous-time models. Chemical production scheduling problems cannot be solved optimally and it is important to consider the solution quality and execution time. Discrete-time formulations performed better with longer time horizons, had smaller optimality gaps with industrial-scale problems, often produced better solutions in a reasonable time and had practical advantages such as interoperability and modelling. The interoperability refers to the ability to integrate scheduling models using small GCF intervals rather than planning models using larger GCF intervals.

2.4.2 Single- (common) and multiple grids

Single-grid formulations have all steps, task or facilities on one time grid. This can introduce complexities when a process occurs on different timescales. A process with a fermentation step on a daily scale and a purification step on an hourly scale will require the model to use an hourly scale when modelling the process (Velez and Maravelias, 2013). This increases the model's complexity proportionally to the number of time points on the horizon. Multiple time grids can have steps, tasks, units or facilities on different time grids to allow for different GCF intervals and reducing the number of time points required in the model.

2.4.3 Uniform and non-uniform grids

A uniform grid has all the time points spaced equally on the grid. Non-uniform grids can have different GCF intervals on one grid. Velez and Maravelias (2013) suggest the term non-uniform should only be used for discrete-time models as continuous-time models always have unequal interval points.

2.4.4 Time representation selection

With multi-period formulations, it is important to include all the decision variables for all the periods or intervals on the entire planning horizon. Solving each interval individually will produce a solution but this might not be optimal over the whole planning horizon (Talbi, 2009).

The two horizons required for the problems were listed in **Chapter 1** as a 90-day horizon with a daily interval and a 14-day horizon with an hourly interval. For this study, a discrete-time model will be used with daily and hourly as the two GCF intervals. Two discrete-time single uniform grids are illustrated in Figure 2.3.

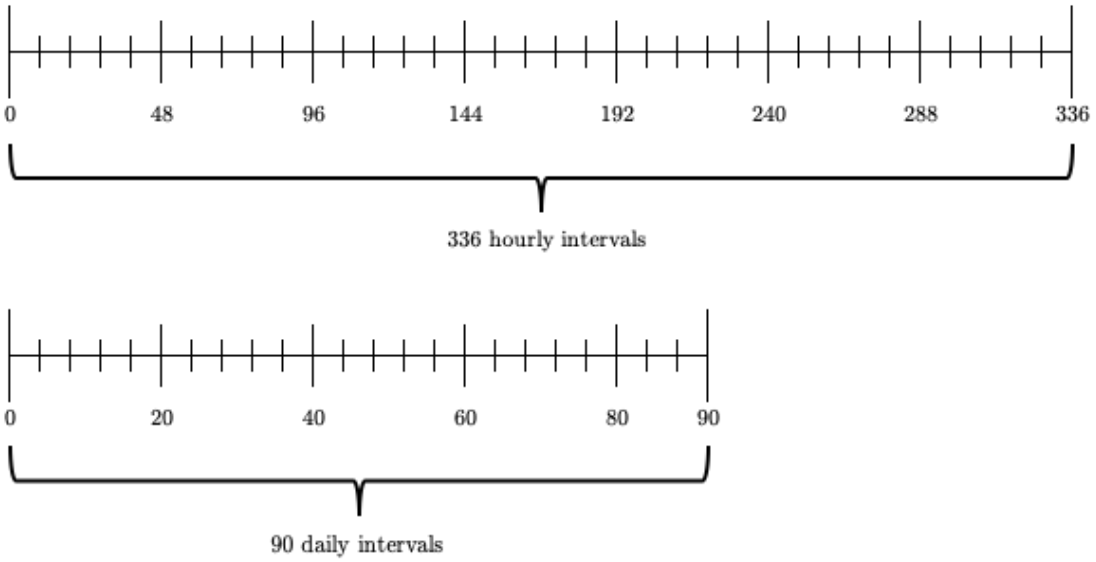


Figure 2.3: Discrete-time single uniform grids

The size of the problem can make it difficult or even impossible to solve (Floudas and Lin, 2004). Having 336 time points for the hourly representation, excluding the decision variables, is a large problem but the size of the problem can be reduced. Different time-based decomposition options will now be discussed.

2.4.5 Time-based decomposition

To reduce the size of the problem, Lagrangean decomposition, Benders decomposition, bi-level decomposition or time-based decomposition can be used (Grossmann, 2012). Two time-based decompositions that are possible for this study include:

Rolling horizon – A strategy where the first interval is solved in detail and the remaining intervals aggregated. After the first interval is solved, the problem is solved again. In the next iteration, the first interval is fixed with the decision made in the first interval, the second interval is focused on in detail with the remaining intervals aggregated. The problem size is reduced with each iteration. This approach can also be done in reverse and the number of detailed intervals in one iteration can be extended (Grossmann, 2012; Stobbe *et al.*, 2000).

Moving-window – Each interval is solved individually before moving to the next interval. This approach originated from the model predictive control field and can lead to sub-optimality (Harjunkoski *et al.*, 2014).

A plant can only increase or decrease its production rate by a limited amount within an hour. This is an important consideration for an hourly interval and less so for the daily interval. It reduced the number of variables for the hourly interval but constrained the options based on the previous interval. Using a rolling horizon will require a discrete-time single non-uniform grid implementation that can move the time scale at each iteration. This will be computationally expensive and complex to implement. Alternatively, a moving-window approach will reduce the search space to one or more intervals at a time and can provide better solutions in certain conditions. The lack of forward-looking over all the intervals can cause infeasible or suboptimal results.

Using a finer grid for the first three days and a coarser grid for the remaining 11 days can reduce the problem size. Optimising the finer grid first and relaying the starting point to the coarser grid or optimising the two grids independently will both lack a complete view over all the intervals. The proposed approach is to optimise both together by using a discrete-time single non-uniform grid. Figure 2.4 illustrates the 72-hour grid and the 11-day grid. The 83 time points are significantly less than the original 336 time points barring the fact that the 11-day grid has a significantly larger search space than the 72-hour grid.

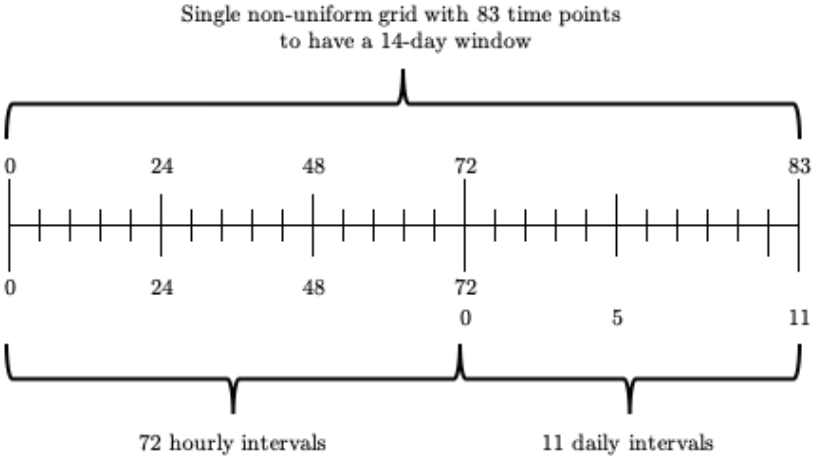


Figure 2.4: Proposed discrete-time single non-uniform grid

The formulation of the two uniform grids in Figure 2.3 and the proposed non-uniform grid in Figure 2.4 will be discussed in a future chapter.

2.5 Synthesis of literature review on scheduling

This section presents a reflection on the scheduling literature reviewed for the process industry. Feedstock optimisation is concerned with balancing the feedstock distribution between multiple units with multiple stages to maximise throughput with the maximum profit.

Similar problems in the process industry with different configurations which have been addressed using mathematical programming have met with varying success. The successes are only for certain configurations, operational conditions and time intervals. Large and complex problems require decomposition, relaxation or approximation techniques that make them less accurate. The literature pointed to the same difficulties highlighted in the problem description of the dissertation.

The researcher has not found similar problems addressed with metaheuristics in the literature and most researchers in this research field continue to favour mathematical programming (Velez and Maravelias, 2015) and continuous-time formulations (Sundaramoorthy and Maravelias, 2011). Genetic algorithm (GA) is a popular choice in this field of study and had comparable or better results in larger problems. From the literature, it seems possible to formulate or model both problems in Figure 2.1 as one problem and use metaheuristics and hybrid metaheuristics to find near-optimal results. The unit-operation-port-state superstructure (UOPSS) modelling using quantity logic and quality (QLQ) can be adapted for metaheuristics in the process industry.

Lastly, a discrete-time representation can be used with a daily and hourly interval. The discrete-time representation will increase the size of the problem, nevertheless it has more practical advantages. To address the size of the problem with the hourly representation, a non-uniform representation is proposed.

2.6 Chapter summary

This chapter started with a short introduction to scheduling by highlighting the classification to identify different scheduling problems. Thereafter, a review of literature in the process industry that is relevant to the feedstock distribution problems in Figure 2.1 is carried out. Castro *et al.* (2018) identified major challenges with scheduling in the process industry and two of the challenges that apply to this study were then discussed. Finally, the literature on time representation was reviewed and considerations identified. The succeeding chapter will review literature on single-objective optimisation, multi-objective optimisation and hybrid metaheuristics with a specific focus on classification, grammar, machine learning and, parallel metaheuristics.

Chapter 3

Optimisation algorithms: selected literature review

Chapter 2 provided a review of scheduling in the process industry and ended with a review of time representations of scheduling models.

Many optimisation problems can be formulated into a single-objective problem thus allowing for single-objective optimisation. This chapter gives an introduction to important aspects of single-objective optimisation to achieve Objective 3, followed by a review of a selection of algorithms that have been developed over the last few decades to achieve Objective 4. Thereafter, a brief introduction to multi-objective optimisation and a review of non-dominated sorting genetic algorithm II (NSGA-II) to achieve Objective 5. Finally, a review of hybrid metaheuristics literature with a specific focus on classification, grammar, parallel metaheuristics and machine learning is presented.

3.1 A brief overview of single-objective optimisation

Single-objective optimisation is concerned with finding the best solution using a single criterion. The best solution could be the minimum of the criterion or the maximum of the criterion. The criterion is referred to as a fitness or objective function and can be expressed in cost, profit, time or another performance or consumption value. Multiple metrics can be defined as a single criterion such as cost or profit.

Formulation of optimisation problems can be broadly classified as exact and approximate: an exact problem formulation uses a mathematical model and can usually be solved in finite time with a guaranteed optimum value for the objective. A linear programming formulation is such an example:

$$\begin{aligned} \min \quad & c'x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

where x is the variable to be determined, c and b are constant vectors with c' indicating the coefficients for the purpose of forming a matrix product (Talbi, 2009).

Many problems can be formulated mathematically but they cannot be solved in finite time because they are complex often having many constraints; in that case, metaheuristics can be used to find good solutions in reasonable time.

Since the scheduling problem is complex, metaheuristics will be considered to obtain near-optimal solutions. Some features of metaheuristics are subsequently discussed.

3.1.1 Metaheuristics genealogy

Metaheuristics represent a group of optimisation techniques developed over the last few decades (Silver, 2004; Talbi, 2009). In practice, there is an emphasis on good solutions within a reasonable time rather than delivering the best solution possible. The term *metaheuristic* was first published in 1986 by Fred Glover and referred to a master strategy that interacts with other heuristics. These high-level algorithms delivered better results than typical heuristics (Glover and Laguna, 1997a). The word "algorithm" (in Latin *algorismus*), is derived from the name of *al-Khwarizmi*, a ninth-century Persian mathematician who wrote about algebraic methods (Weise, 2009).

Depending on the complexity of the problem, exact methods can obtain optimal solutions. Exact methods will search the solution area and find the optimal solution by subdividing the problem into smaller problems. These methods include dynamic programming, branch and bound and constraint programming. Exact methods are limited to less complex problems whilst for more complex problems, approximation methods will deliver a high-quality solution in a reasonable time. Approximation methods cannot guarantee finding a global optimal solution (Talbi, 2009).

Williamson and Shmoys (2012) refer to an old engineering saying, *Fast, Cheap or Reliable. Choose two.* Similarly with optimisation; if exact methods do not exist, algorithms cannot have, (1) optimal solutions, (2) in polynomial time, (3) for any instance. Approximation methods allow for relaxation of these requirements. Approximation methods can be divided into two classes; namely, *approximation* and *heuristic* algorithms. Approximation algorithms provide sub-optimal solutions in

polynomial time. Due to the sub-optimality, approximation algorithms are not very useful in real-life applications. Heuristics can usually find good solutions to large and complex problems in a reasonable amount of time. Heuristics can be classified into two groups, namely *specific heuristics* and *metaheuristics*. Specific heuristics relax the third requirement (for any instance) and are developed for specific problems. Metaheuristics can find good solutions and can be applied to a wide variety of problems in a reasonable amount of time. The aim is to find a general-purpose algorithm that finds a close to optimal solution in a reasonable time. Metaheuristics can be used as a starting point in finding underlying heuristics to solve specific problems.

3.1.2 Single-solution and population-based solutions

A local search (LS) algorithm starts with a feasible solution and searches the neighbourhood for an improved solution. The improved solution is selected and the search for an even better solution starts.

This process is reiterated until no improved solution can be found. This is considered a local optimum and a fundamental weakness in local search methods. The first local search algorithm was developed in 1947 and subsequently, more than eight improved algorithms exist that are based on local search which include simulated annealing (SA) and tabu search (TS) (Talbi, 2009). This group of methods is referred to as **single-solution based** methods.

Population-based methods started in the early 1960s and many are inspired by natural processes. These include genetic algorithm (GA), particle swarm optimisation (PSO) and ant colony optimisation (ACO). Population-based algorithms generate a population of solutions and evolve improved solutions using different strategies. Single-solution based metaheuristics intensify the search in an area whereas population-based metaheuristics diversify the search in the whole area (Glover and Laguna, 1997a; Talbi, 2009).

3.1.3 Metaheuristics decision variable representation

An important aspect of metaheuristics representation, is the representation of decision variables as a vector with an associated objective function. This will define the search space of the problem. Some of the classical encodings include binary values, discrete values, real values and permutations (Talbi, 2009).

Binary encoding is when the decision variables can be expressed in a string containing 0's and 1's. Some typical problems include the knapsack problem, satisfiability problem and 0-1 integer programming problems. A binary vector can be presented

as follows:
$x \in [0, 1, 0, 1, 0, 0, 0, 1]$

Discrete value encoding is where the decision variables can be expressed in discrete values. Some typical problems include location problems and assignment problems. A discrete value vector can be presented as follows:
$x \in [6, 3, 6, 1, 2, 0, 9, 5]$

Real value encoding is where the decision variables can be expressed in real values. Some typical problems include continuous optimisation, parameter identification and global optimisation. A real value vector can be presented as follows:
$x \in [1.34, 2.34, 5.61, 2.16, 9.02, 2.53, 9.12, 6.25]$

In permutation encoding, the decision variables can only appear once in the vector. Some typical problems include sequencing problems, travelling salesman problems and scheduling problems. A permutation value can be presented as follows:
$x \in [6, 3, 4, 1, 2, 8, 7, 5]$

The representation plays a fundamental role in the efficiency of any metaheuristic. Real numbers can be encoded into binary vectors but may generate non-efficient metaheuristics due to disparity (Talbi, 2009). It can also increase the length of the chromosome leading to an increase in computation cost (Zacharia *et al.*, 2013). The additional encoding and decoding at each iteration will further increase the computational cost (Rahman *et al.*, 2015). Nguyen and Bagajewicz (2010) warn that real number vectors have more difficulty converging with large genetic algorithms than with binary representations. A real value representation that is limited to a 0,5 increment can provide sufficient accuracy for the problem and limits the complexity introduced by real numbers.

3.1.4 Metaheuristics objective function

The objective function is used to determine the quality of the solution. It is important to take care of defining the objective function properly to avoid unacceptable solutions (Talbi, 2009). Objective functions in the process industry are frequently profit-related, which maximises the flow of products or the return on investment (Zyngier and Kelly, 2012).

The objective function implemented in this study will be discussed in detail in the next chapter.

3.1.5 Metaheuristics constraint handling

A difficulty that is introduced with industry problems, is that most of them have to deal with constraints (Hansen *et al.*, 2010). Lin and Miller (2004), Talbi (2009)

and Weise (2009) give a good overview of typical constraint-handling strategies which include:

- Reject Strategies – Discarding infeasible solutions from the search space. This is possible when the discarded search space is very small.
- Penalising Strategies – Infeasible solutions are penalised using a penalty function but are still considered during the search process.
- Repairing Strategies – Transforming infeasible solutions into feasible solutions.
- Decoding Strategies – Indirectly encoding the representation to feasible solutions.
- Preserving Strategies – Ensuring values used in vectors and operators generate feasible solutions by embedding problem-specific knowledge.
- Additional Objectives – Formulate constraints as additional objectives.

A combination of constraint-handling strategies is required in this study. Initial solution variables can only be selected from feasible sets and will be discussed in the next chapter. Supply and demand (consumption) constraints are used to ensure the value chain is balanced. Bell (1983) proposed a balancing period; the intent is to allow a temporary imbalanced network and use the connecting pipelines, tanks and flaring to balance the network. The strategies for the supply and demand constraints are applied in a sequence until the constraint is satisfied and if none of the methods resolves the violation, the solution vector is rejected. Figure 3.1 illustrates the C_2 value chain, $P_{1,1}$, $P_{2,1}$, $P_{1,2}$ and $P_{2,2}$ *supply* ethylene and $P_{3,2}$, $P_{4,2}$, $P_{5,2}$, $P_{6,2}$ and $P_{3,1}$ *demand* ethylene. To match the supply and demand precisely is not possible without buffer storage and two typical scenarios will now be discussed.

When *supply* exceeds *demand*, the supply constraint violation is addressed by:

1. Increasing the rate of the consuming plants within the feasible region;
2. Storing the excess product in the pipeline;
3. Liquefying the feed to a liquid and storing it in a tank;
4. Flaring the excess supply.

When *demand* exceeds the *supply*, the demand constraint violation is addressed by:

1. Reducing the rate of the consuming plants within the feasible region;
2. Using product in the pipeline;
3. Vaporising liquid to gaseous feed from tank inventory;
4. Shutting down a unit.

Liquefying feed, vaporising stored feed, flaring feed and shutting down a unit can be used at a cost. This cost penalty is represented in the objective function.
Other constraints in this study are used with a rejection strategy:

1. Contractual feed obligation (Ratio) – The fixed supply ratio with $P_{6,2}$ must be adhered to;
2. Pipeline pressures (Min/Max) – The pipelines connecting $P_{\mathrm{i,j}}$ and $T_{\mathrm{i,j}}$ have a minimum and maximum that cannot be exceeded;
3. Tank levels (Min/Max) – $T_{1,1}$, $T_{1,2}$ and $T_{2,2}$ have a minimum and maximum that cannot be exceeded.

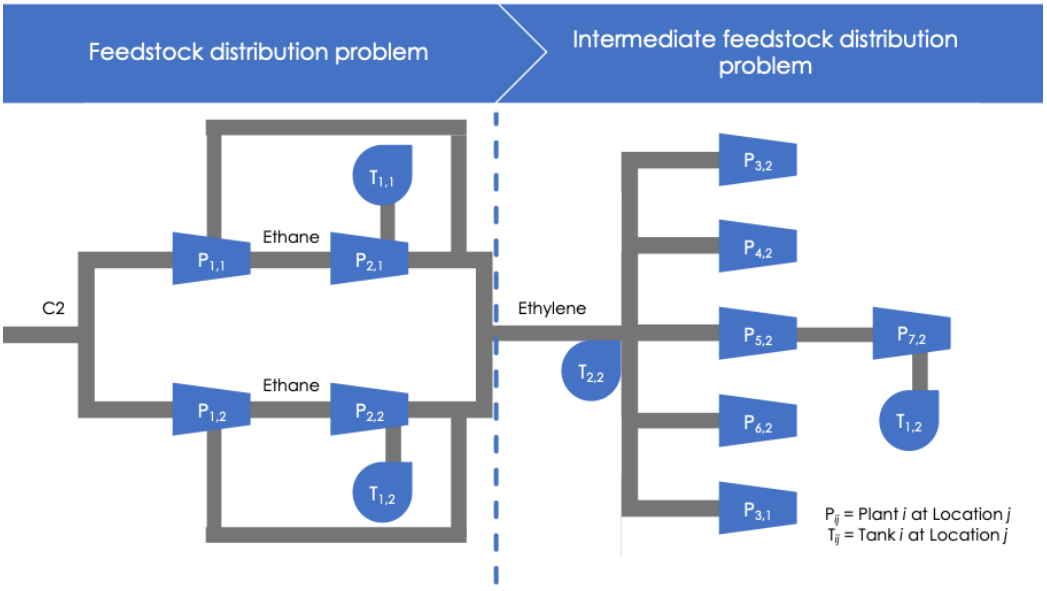


Figure 3.1: The C_2 value chain

The constraints that are embedded in the representation will be discussed in the next chapter.

3.1.6 Termination conditions

Global optimality is not guaranteed and the search process can continue until some termination criteria are met. Maximum time and maximum iterations as termination criteria are easy to implement and are widely used. The maximum iteration stopping criterion is used in this study where applicable.

3.2 Single-objective optimisation algorithms

Brownlee (2011) lists over 40 algorithms that can be used in single-objective optimisation. The list is not complete but includes a comprehensive list of "Clever Algorithms" that can be used. Fred Glover, the father of Tabu search, and other authors recently stated that the development of some algorithms in the last five decades is not important or has made only marginal additions to the field. They further state that some metaphor-based algorithms are not science and are even harmful to the field in general (Sörensen *et al.*, 2018). As highlighted in Section 2.5, similar problems addressed by metaheuristics has not been found and therefore only well-documented metaheuristics that have been researched extensively were selected for the study. The algorithms reviewed in this section have been researched extensively since first publication and are widely acknowledged in the field (Kallrath, 2002).

3.2.1 Local search

One of the oldest algorithms in metaheuristics is local search and it is the basis of many single-solution based algorithms (Talbi, 2009). It is one of the simplest algorithms to implement and can return good solutions (Blum *et al.*, 2008; Williamson and Shmoys, 2012). It starts with an initial solution and searches the neighbouring solutions for a better solution. When the search has completed the neighbourhood search, the best neighbouring solution is accepted and used for the next search iteration. The search ends when no neighbouring solution improves the current solution. The interest in this study, should the LS algorithm start in a good neighbourhood, is that it will improve the result in a reasonable time. A pseudo representation of the LS algorithm can be seen in Algorithm 3.2.1.

Algorithm 3.2.1: Local search pseudocode

```
   Input:
   Output: S_best
1  S_current ← ConstructInitialSolution();
2  S_best ← S_current;
3  while ¬ StopCondition() do
4  |  CandidateList ← ∅;
5  |  CandidateList ← CreateNeighbouringSolutions(S_candidate);
6  |  S_candidate ← CandidateList(S_bestNeighbour);
7  |  if S_best ≤ S_candidate then
8  |  |  S_best ← S_candidate;
9  |  else
10 |  |  StopCondition();
11 |  end
12 end
13 return S_best;
```

3.2.2 Tabu search

The tabu search algorithm was first coined in 1986 by Fred Glover (Glover and Laguna, 1997a). In the same year, Pierre Hansen developed what is called the steepest ascent / mildest descent algorithm (Hansen, 1986). The word "tabu" originated from Polynesia which relates to this algorithm as, things that are tabu should not be visited and should be left alone (Weise, 2009). TS belongs to the local search class and was based on three approaches namely surrogate constraints and cutting plane approaches with a strong influence from steepest ascent / mildest descent (Glover and Laguna, 1997b). Glover cites 72 example applications in scheduling, production, inventory, investment, location, allocation supply and other (Glover and Laguna, 2013).

The TS algorithm searches the neighbourhood for solutions and escapes the local optimum by allowing tabu moves. The memory ability with TS, allows the algorithm to continue searching the space and not to become stuck in a region. Intensification and diversification strategies are employed to ensure a promising region is searched thoroughly while exploring the entire feasible region.

The TS algorithm generates a set of neighbour solutions. The best solution in the set is used as the candidate solution and examined if the solution exists in the tabu list. TS discards solutions that have been previously visited. TS manages a short-term memory of recent solutions and is called a *tabu list*. The candidate solution is compared to the best solution and if found to be a better solution, replaces the best solution. The candidate solution is then added to the tabu list. The tabu list is kept below the maximum tabu list size by removing the oldest entry. The candidate solution is set as the starting point for the next iteration even if the candidate solution

was worse than the best solution (Lin and Miller, 2004; Talbi, 2009; Weise, 2009). A pseudo representation of the TS algorithm can be seen in Algorithm 3.2.2.

Algorithm 3.2.2: Tabu search pseudocode

Input: $TabuList_{size}$, $iterations_{max}$
Output: S_{best}

```
1  S_best ← ConstructInitialSolution();
2  TabuList ← ∅;
3  while ¬ i < iterations_max do
4  |   CandidateList ← ∅;
5  |   CandidateList ← CreateNeighbouringSolutions(S_candidate);
6  |   S_bestNeighbour ← LocateBestCandidate(CandidateList);
7  |   if S_candidate ≤ S_bestNeighbour then
8  |   |   S_candidate ← S_bestNeighbour;
9  |   |   if S_best ≤ S_bestNeighbour then
10 |   |   |   S_best ← S_bestNeighbour;
11 |   |   end
12 |   else if S_bestNeighbour ∉ TabuList then
13 |   |   S_candidate ← S_bestNeighbour;
14 |   |   TabuList ← S_bestNeighbour;
15 |   else
16 |   |   StopCondition();
17 |   end
18 |   if TabuList ≥ TabuList_size then
19 |   |   DeleteListItem(TabuList);
20 |   end
21 end
22 return S_best;
```

3.2.3 Simulated annealing

The simulated annealing algorithm was inspired by a physical process in metallurgy, on how metal crystals reconfigure and reach equilibria after being cooled from a high temperature (Glover and Laguna, 1997a; Weise, 2009). Kirkpatrick *et al.* (1983) developed the SA algorithm in 1983 and applied it to a wide variety of combinatorial optimisation problems. In mid-1980, Černý (1985) independently published a thermodynamic approach to the travelling salesman problem. The SA algorithm forms part of the LS family in single-solution based metaheuristics and can accept movements to non-improvement neighbours (Talbi, 2009).

An initial solution is generated and the neighbourhood is searched for improved neighbours. Improving neighbours are always accepted and non-improving neighbours are accepted depending on the degradation of the objective function and the current temperature. The probability that non-improving moves are accepted

decreases as the algorithm progresses or until the equilibrium condition has been reached. ΔE or $f(s') - f(s)$ represent the difference between the current solution and the neighbour solution respectively. The temperature (T) defines the annealing schedule which a fraction of the time or iterations expended so far. The Boltzmann distribution is used as a probability distribution and can be stated as $P(\Delta E, T) = e^{-\frac{f(s')-f(s)}{T}}$. A pseudocode representation of the SA algorithm can be seen in Algorithm 3.2.3.

Algorithm 3.2.3: Simulated annealing pseudocode

Input: $cooling$, $temp_{init}$
Output: S_{best}
$S_{current} \leftarrow$ `CreateInitialSolution()`;
$S_{best} \leftarrow S_{current}$;
$temp_{curr} \leftarrow temp_{init}$;
while $\neg\ temp_{curr} > temp_{0.01}$ **do**
  CandidateList $\leftarrow \emptyset$;
  CandidateList $\leftarrow$ `CreateNeighbouringSolutions(`$S_{candidate}$`)`;
  $S_{bestNeighbour} \leftarrow$ `LocateBestCandidate(CandidateList)`;
  $temp_{curr} \leftarrow$ `CalculateTemperature(`$cooling$, $temp_{curr}$`)`;
  if $S_{candidate} \leq S_{bestNeighbour}$ **then**
    $S_{candidate} \leftarrow S_{bestNeighbour}$;
    if $S_{best} \leq S_{bestNeighbour}$ **then**
      $S_{best} \leftarrow S_{bestNeighbour}$;
    end
  else if `Exp(` $\frac{S_{candidate} - S_{bestNeighbour}}{temp_{curr}}$ `)` $>$ `Rand()` **then**
    $S_{candidate} \leftarrow S_{bestNeighbour}$;
  end
end
return S_{best};

3.2.4 Genetic algorithm

The genetic algorithm imitates the biological selection of evolutionary reproduction and belongs to the larger class of evolutionary algorithms (EAs) (Glover and Laguna, 1997a). John Holland was inspired by the mechanism of natural selection which led to the development of the original GA in 1962 and the development was continued during the 1960s and 1970s by Holland, his colleagues and students (Mitchell, 1995). During the 1980s they applied GA to machine learning and optimisation (Talbi, 2009). Holland describes in his framework the method for evolving from one population of *chromosomes* to a new population by replacing the current population. Traditionally, a chromosome represented the candidate solution for the problem in *bits* format but nowadays other types of representation are used (Talbi, 2009).

Kasat *et al.* (2003) presented a range of GA applications in the polymer science and engineering domain. Relevant to this study is the scheduling of production producing expandable polystyrene using a GA. Wang *et al.* (2000) found that the computing times and quality of solutions obtained by the GA were comparable with the mixed-integer non-linear programming (MINLP) models.

Population-based metaheuristics such as the GA start from an initial population of solutions represented in chromosomes. Each chromosome consists of genes that represent the decision variables in a solution. The fitness of each individual in the population is calculated. Using a selection strategy, offspring are generated using crossover, mutation and inversion. The offspring are introduced to the population and the weakest solutions are removed from the population. A pseudo representation of the GA algorithm can be seen in Algorithm 3.2.4.

Genetic algorithms are known to require parameter tuning and the use of GA as a black box algorithm could result in sub-optimal results (Hansen *et al.*, 2010; Lobo and Goldberg, 2004). The following sections cover important aspects of genetic algorithms that require careful consideration.

3.2.4.1 Chromosomes

A chromosome is a vector with the decision variables or genes represented as described in subsection 3.1.3. This can be binary encoding, discrete value encoding, real value encoding or permutation encoding. More complex chromosomes can be used by using a combination of different encodings.

3.2.4.2 Population

The population, consisting of a list of chromosomes, is the core of the algorithm. It determines the memory size, the convergence speed in sequential GAs and affects the speed of search in parallel GAs (El-Milhoub *et al.*, 2006). The size of the population and how the initial population is generated play a big role. A big population may have more space for the algorithm to explore but would impair the efficiency of the algorithm. A smaller population would not have space to explore and would not be effective at finding near-optimal solutions. The trade-off between big and small population sizes has been studied from multiple points of view (Diaz-Gomez and Hougen, 2007; El-Milhoub *et al.*, 2006; Elmihoub *et al.*, 2004; Gotshall and Rylander, 2000; Hansen *et al.*, 2010; Maaranen *et al.*, 2007).

One of the first researchers that studied the parameters for EAs devised test functions and concluded that a population size of 50-100 would result in a good performance. These parameter settings and other have since been used by many

researchers (De Jong, 1975). Gotshall and Rylander (2000) proposed a method for determining the optimal population size for GAs and conducted experiments on dissimilar problems with positive results. Harik *et al.* (1999) and Elmihoub *et al.* (2004) used the gambler ruin model to estimate the population size of GAs. Both reported reaching a solution of a particular quality for small, medium and hybrid GAs.

De Jong (2007) since published a 30-year perspective on parameter settings and stated that due to the no free lunch (NFL) theorem, there is no single algorithm that will outperform all others. Unless adequate restrictions are placed on the problems, that optimal parameter setting will not exist. Lobo and Goldberg (2004) presented a parameterless method for crossover-based GAs. The method is slower than for a GA with optimal parameter settings but it removes the effort of finding the optimal parameters. De Jong (2007) states that parameterless methods or no externally visible parameters are the ultimate goal.

Levine (1997) recommends using a steady-state GA. The selection of two parents can generate one or two offspring, evaluate their fitness and reintroduce them into the population while killing off random or weaker individuals in the population. This provides elitism and uses less memory (Lones, 2011). This also addresses the problem stated by De Jong (2007) who stated that the population size should be set independently from the offspring population size.

3.2.4.3 Mutation

The mutation operator helps maintain diversity in the population by doing a random bit-flip in the chromosome. Various methods can be used for other representations.

For floating-point representations, it is rare to use something other than Gaussian mutation (Lones, 2011). Lones (2011) suggests *integer randomisation mutation* for when the representation treats integers as members of a set and *random walk mutation* when integers represent a metric space. Luke further cautions against the use of *point mutation* as if this is not constructed properly, it could result in limiting rather than diversifying the population. A common strategy to include elitists in GAs is by excluding the best chromosome from mutation (Haupt and Haupt, 2003).

Research varies extensively; some point to fixed mutation rates while others advocate for adaptive mutation rates (Blum *et al.*, 2005b; Carr, 2014; De Jong, 2007, 1975; Grobler *et al.*, 2010; Hansen *et al.*, 2010; Talbi, 2009). Until a high performing adaptive method has been found and tested with various problems, it will be up to the researcher to experiment with different methods and rates.

For this study, the researcher compared two mutation methods; namely, no mutation and mutation using local search. The mutation using local search selects a

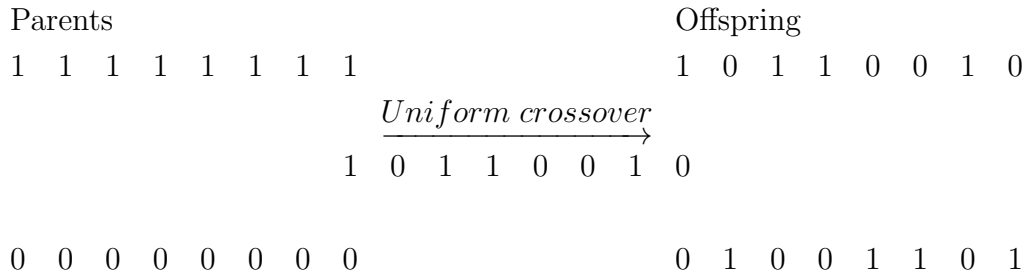


Figure 3.2: Uniform crossover with a binary representation

random variable, if the variable is smaller than the mutation variable, a local search is applied on the selected chromosome. The mutation variable is set at 0,07 for this study.

3.2.4.4 Crossover

Crossover or recombination enables two parent chromosomes to be recombined into new chromosomes so that the new chromosome inherits characteristics of the parents. Hansen *et al.* (2010) recommend not using one-point crossover due to positional bias. Multiple crossovers would address the positional bias. To remove any bias, a uniform crossover could be used by generating a Bernoulli distributed 0s and 1s vector and crossing over on the 1s. Nguyen and Bagajewicz (2010) found better solutions using uniform crossover with computation times that is relative to other methods and it scales well with large problems. Due to the size of the problem in the study, a uniform crossover was used.

Figure 3.2 illustrates a uniform crossover with a binary representation. A mask is generated with 0 and 1 from crossover where 0 will remain and 1 will crossover into the offspring (Hansen *et al.*, 2010). The daily interval solution vector has a length of 1 980 and the hourly interval solution vector a length of 7 392. Using a crossover procedure that scales with the problem is important or it will not be effective. Figure 3.3 illustrates a sample from the vector in the daily interval solution vector representation.

3.2.4.5 Selection

The selection mechanism provides for stronger individuals to become parents. This allows for the improvement of the next generation. Weaker individuals also have a chance to become a parent and are not necessarily discarded.

Roulette wheel selection is the most common selection strategy (Talbi, 2009). Hansen *et al.* (2010) discourages the use of a simple roulette wheel selection and recommends tournament selection or stochastic universal sampling (SUS). SUS has zero

Parents

17.5	25.5	11.5	4.5	4	4.5	17.5	16.5

15.5	28	9.5	2.5	3	5	19	11

Uniform crossover

1	0	1	1	0	0	1	0

Offspring

17.5	28	11.5	4.5	3	5	17.5	11

15.5	25.5	9.5	2.5	4	4.5	19	16.5

Figure 3.3: Uniform crossover with an example from study

bias but can only be used in sequential algorithms (Baker, 1987). Baker (1987) proposed a different selection method namely remainder stochastic independent sampling for parallel algorithms. Jinghui Zhong *et al.* (2005) has found that in an experiment using a simple genetic algorithm (SGA), tournament selection outperformed the SGA using roulette wheel selection. Lones (2011) recommends tournament selection as it is a simple tunable technique that works well with parallel algorithms. Tournament selection was used based on most research recommending the method.

Algorithm 3.2.4: Genetic algorithm pseudocode

Input: $Population_{size}$, $Generation_{size}$, $P_{mutation}$
Output: S_{best}
Population ← `InitialisePopulation(`$Population_{size}$`)`;
`EvaluatePopulation(`Population`)`;
S_{best} ← `GetBestSolution(`Population`)`;
for $i \leq Generation_{size}$ **do**
  $Parent_1$ ← `TournamentSelection(`Population`)`;
  $Parent_2$ ← `TournamentSelection(`Population`)`;
  Children ← ∅;
  $Child_1$, $Child_2$ ← `UniformCrossover(`$Parent_1$, $Parent_2$`)`;
  Children ← `Mutate(`$Child_1$, $P_{mutation}$`)`;
  Children ← `Mutate(`$Child_2$, $P_{mutation}$`)`;
  `EvaluatePopulation(`Children`)`;
  Population `Replace(`Population, Children`)`;
  S_{best} ← `EvaluatePopulation(`Population`)`;
end
return S_{best};

3.2.5 Greedy search

The greedy search (GS) algorithms are approximation algorithms that systematically search the solution space in a best-first approach. Wilt *et al.* (2010) compared greedy search families; namely, best-first, hill-climbing, and beam search and found that massive search space requires beam search. Best-first often had comparable results in a timed comparison. Pekny (2002) refers to the greedy algorithm as an algorithm that can complete the search in a reasonable time but with very low-quality solutions. The interest in the GS algorithm for this study is similar to the LS algorithm, in that should the GS algorithm start in a good neighbourhood, it will improve the result in a reasonable time.

The GS algorithm starts at a solution and evaluates the nearest-neighbour; if the neighbour improves the current solution, the move is accepted. If the move is worse than the current solution, the next neighbour is considered. This process continues until a local optimum has been reached. A pseudo representation of the GS algorithm can be seen in Algorithm 3.2.5.

Algorithm 3.2.5: Greedy search pseudocode

```
   Input:
   Output: S_best
 1 S_best ← ConstructInitialSolution();
 2 while ¬ StopCondition() do
 3 |   S_candidate ← CreateNeighbourSolution(S_best);
 4 |   if S_best ≤ S_candidate then
 5 |   |   S_best ← S_candidate;
 6 |   else
 7 |   |   StopCondition();
 8 |   end
 9 end
10 return S_best;
```

3.2.6 Summary of single-objective optimisation algorithms

The previous sections provided a review of a selection of single-objective optimisation algorithms. The algorithms reviewed would end either by reaching a local optimum or reaching the stopping criterion or criteria. This would be after exploring and exploiting the search domain sufficiently. Exploiting the neighbourhood further could be done by supplementing the algorithm with other algorithms. This is called hybrid metaheuristics, which will be reviewed later in this chapter. The following section provides a brief overview of multi-objective optimisation.

3.3 A brief overview of multi-objective optimisation

The petrochemical and chemical sector accounts for 30% of the energy use in industry worldwide (Saygin *et al.*, 2011). The energy demand of a chemical processing plant can be enormous, and operating a plant at a slightly lower rate that uses less energy might not result in a proportional decrease in profit. With many practical industrial problems, multiple criteria or objectives can be used to evaluate a solution. The trade-off between *maximising profit* while *minimising energy consumption* provides additional information when making decisions on the production plan.

Multi-objective optimisation (MOO) is concerned with finding the best set of solutions which are defined as the *Pareto optimal solutions* and, consequently a Pareto set. The Pareto optimal solutions represent a compromise between conflicting objectives with no solution being possible to improve without deteriorating at least one other objective (Talbi, 2009). When metaheuristics are applied on a multi-objective optimisation problem (MOOP), the goal becomes to obtain an approximation Pareto set. MOO is classified further into bi-objective optimisation and many-objective optimisation. Bi-objective optimisation refers to problems containing two objectives and many-objective optimisation to those containing a large number of objectives, typically more that three or four objectives (Copado-Méndez *et al.*, 2012; Mane and Narasinga Rao, 2017; Talbi, 2009). Since the problem in this study contains two objectives, the term bi-objective optimisation is used when referring to the problem in this study and MOO to describe the problem in general. An MOOP formulation can be defined as

$$MOOP = \begin{cases} \min \quad \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})) \\ \text{subject to} \quad \mathbf{x} \in S \end{cases}$$

where the integer $n \geq 2$ is the number of objectives, $\mathbf{x}$ the feasible set of decision variables of the vector representation $\mathbf{x} = (x_1, \ldots, x_k)$ and $F(\mathbf{x})$ the vector of objectives to be optimised (Talbi, 2009).

Zitzler *et al.* (2000) compared six multi-objective evolutionary algorithms on test problems with the strength Pareto evolutionary algorithm (SPEA) outperforming the other algorithms such as the non-dominated sorting genetic algorithm (NSGA). The comparison highlighted the importance of elitism in MOO. Deb *et al.* (2002) introduced NSGA-II with several improvements to address the shortcomings such as elitism. Zitzler *et al.* (2001) introduced the second version of SPEA; namely, strength

Pareto evolutionary algorithm 2 (SPEA2), which included an improved fitness assignment strategy with a new density-based selection and archived truncation techniques. They compared SPEA2, the Pareto envelope-based selection algorithm (PESA) and NSGA-II, with both the NSGA-II and SPEA2 having the best performance overall. It was noted that they behave similarly on different problems, with NSGA-II in some cases having a broader spread while SPEA2 has a better distribution of points. The NSGA-II was selected for this study as it is an extension of the GA that was reviewed earlier in this chapter.

3.3.1 Non-dominated sorting genetic algorithm II

The NSGA-II was proposed by Deb *et al.* (2002) as an improvement on the NSGA proposed in Srinivas and Deb (1994) for multi-objective optimisation. The improvements to the NSGA-II focused on the high computational complexity of the non-dominated sorting, the lack of elitism, and the need for specifying the sharing parameter in the NSGA. The non-dominated sorting genetic algorithm III (NSGA-III) was later introduced which does not make use of any explicit selection operator and makes use of reference directions to maintain diversity (Deb and Jain, 2014; Seada and Deb, 2015). For this study, the NSGA-II was chosen as the reference point approach in the NSGA-III is not required for this problem. The NSGA-II procedure is illustrated in Figure 3.4 and will now be described.

An initial population P_t is generated and the children Q_t are created with a crossover procedure as described in Section 3.2.4.4. To maintain diversity, mutation is done on the children as described in Section 3.2.4.3. Each chromosome in the population R_t is then evaluated with each objective. The population is sorted and the first non-dominating set F_1 is calculated. Any chromosome that is dominated by another is moved to the next set F_2. This process continues until all the chromosomes in the population are assigned to a non-dominating set. Once completed, the crowding distance is calculated for each chromosome with the following two equations:

$$cd^1 = 0 + \frac{f^k_{n+1} - f^k_{n-1}}{f^k_{max^1} - f^k_{min^1}}$$

$$cd^2 = cd^1 + \frac{f^k_{n+1} - f^k_{n-1}}{f^k_{max^2} - f^k_{min^2}}.$$

Chromosomes of the best non-dominating set F_1 are assigned to the new population P_{t+1}. The succeeding sets are then added until P_{t+1} is filled. If the complete set cannot be accommodated in P_{t+1}, the chromosomes are selected based on the crowding distance to fill P_{t+1}. Finally, crossover and mutation are performed to generate

the children and after evaluating the fitness of the children, the process repeats until the stopping condition is met.

A pseudocode representation of the NSGA-II algorithm can be seen in Algorithm 3.3.1.

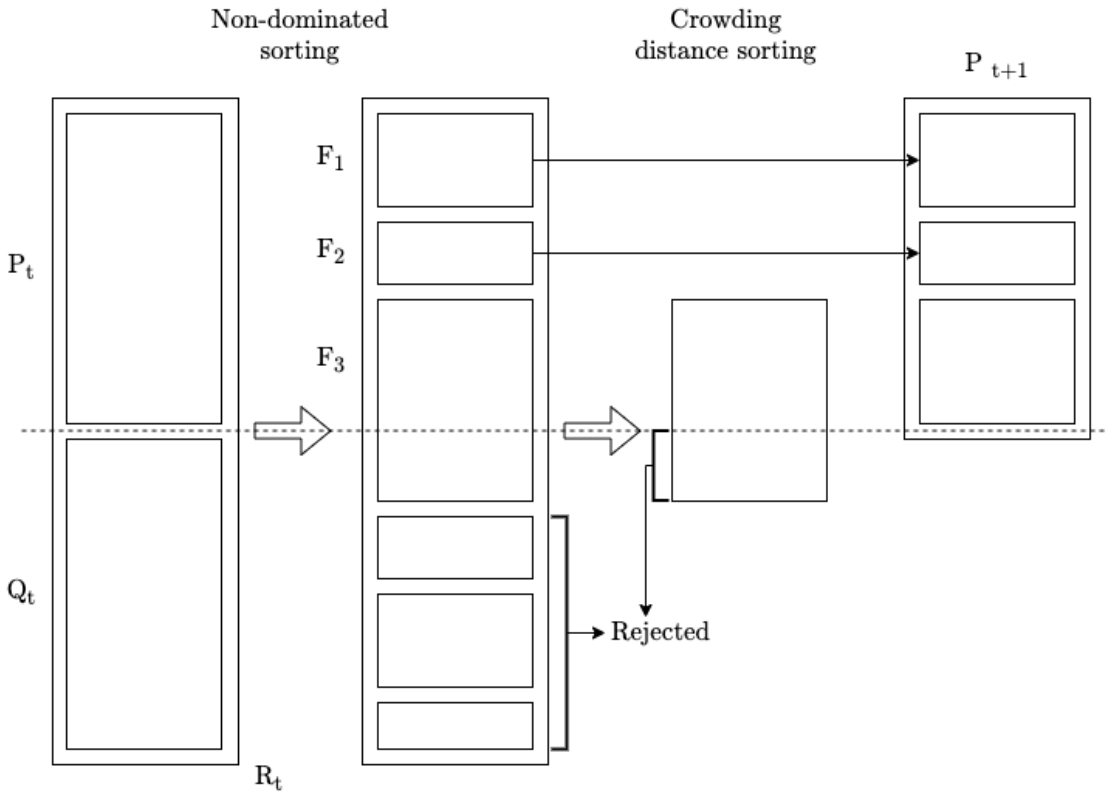


Figure 3.4: NSGA-II procedure (Deb *et al.*, 2002)

Algorithm 3.3.1: Non-dominated sorting genetic algorithm II pseudocode

Input: $Population_{size}$, $P_{crossover}$, $P_{mutation}$
Output: Population

```
1  Population ← InitialisePopulation(Population_size);
2  Children ← UniformCrossoverAndMutation(Population, P_crossover, P_mutation);
3  Population ← Merge(Population, Children);
4  EvaluatePopulation(Population);
5  while ¬StopCondition() do
6      Fronts ← FrontAssignment(Population);
7      CrowdingDistanceAssignment(Population);
8      Parents ← SelectParentsByFront(Population);
9      if Size(Parents)<Population_size then
10         Front_L ← SortByRankAndDistance(Front_L);
11         for P_1 to P_{Population_size − Size(Front_L)} do
12             Parents ← Pi;
13         end
14     end
15     Children ← UniformCrossoverAndMutation(Population, P_crossover,
           P_mutation);
16     EvaluatePopulation(Children);
17     Population ← Merge(Population, Children);
18 end
19 return Population;
```

3.4 Hybrid metaheuristics

Hybrid metaheuristics are a combination of population-based metaheuristics, single-based metaheuristics, mathematical programming, constraint programming and *machine learning* techniques (Talbi, 2009). When a simple genetic algorithm with tuning does not perform as required, it has been proven beneficial to combine the algorithm with others. Combining an evolutionary algorithm with an LS was coined as a memetic algorithm (Moscato, 1989). Noman *et al.* (2011) list other known references in literature for hybridisations as hybrid GAs, genetic local searches, Lamarckian GAs and Baldwinian GAs. The GA algorithm lends itself to hybridisation in three ways. First, to use a memetic algorithm as a high-level relay hybrid once a GA has found the best feasible solution, to exploit the neighbourhood to ensure each variable is at its best position by using LS, TS or GS. Secondly, it is applied as a low-level teamwork hybrid, by using local search as a mutation operator. Lastly, as a high-level relay hybrid where an approximation multilayer perceptron (MLP) model is built from previously evaluated solutions. The following sections briefly discuss the classification and taxonomy of hybrid metaheuristics.

3.4.1 Classification of hybrid metaheuristics

Talbi (2002) proposed a taxonomy for hybrid metaheuristics consisting of a classification and grammar. The classification consists of a hierarchical and flat classification with the grammar based on the classification.

The hierarchical and flat classification of the taxonomy are shown in Figure 3.5, followed by a brief discussion.

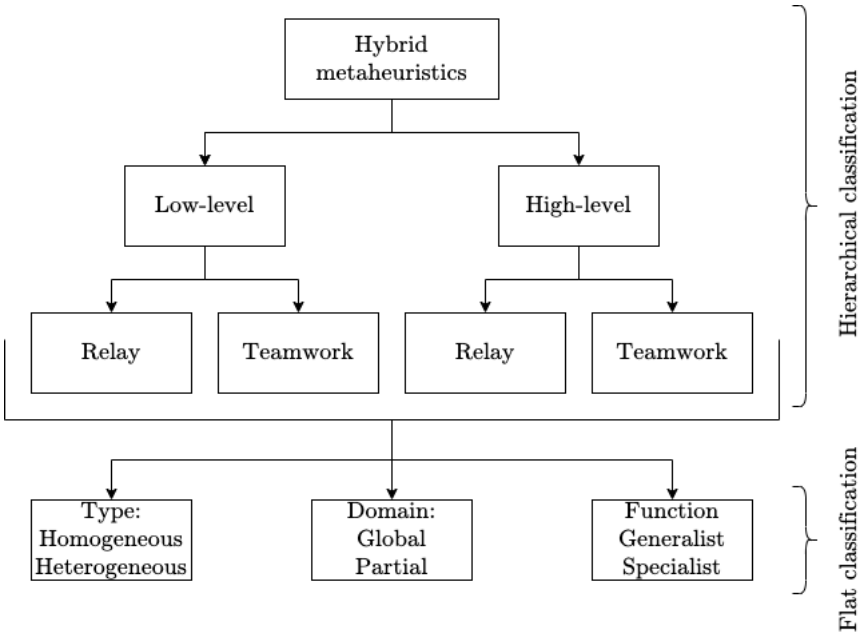


Figure 3.5: Taxonomy classification hybrid metaheuristics (Talbi, 2002)

The two categories in the hierarchical classification proposed by Talbi (2009) are *level* and *mode*. Blum *et al.* (2008) classify the combinations differently as *integrative* or *collaborative*.

The level or first classification is made between low-level and high-level hybrid metaheuristics. With low-level hybridisation, a function of a metaheuristic is replaced with another metaheuristic. With high-level hybridisation, the internal procedures of a metaheuristic have no direct interaction or replacement of functionality. Raidl and Puchinger (2008) describe integrative combinations as the combination of one technique embedded as a component of another technique. The latter lacks the differentiation of low or high integration.

The mode or second classification is made between relay and teamwork. With relay hybridisation, the output of one metaheuristic is the input of the next metaheuristic. This type is often referred to as sequential execution (Raidl and Puchinger, 2008). With teamwork hybridisation, the metaheuristics work together to search for

the optimum in parallel. This type is often referred to as *parallel* or *intertwined* execution (Raidl and Puchinger, 2008). Collaborative combinations focus on the execution, sequentially, intertwined, or in parallel. An illustration of the structural classification by Raidl and Puchinger (2008) can be simplified and extended to include other integrative combinations. An adapted structural classification is shown in Figure 3.6.

Raidl (2006) proposed a unified view of hybrid metaheuristics combining the work of Talbi (2002) including the views of Cotta-Porras (1998) and Blum *et al.* (2005a). The unified view excluded the grammar proposed by Talbi (2002). Hybrid metaheuristics literature uses either of the classifications or a combination of the two. The grammar classification makes it easier to discover literature on specific algorithms. This unified view, in reality, introduced another classification rather than unifying the views. Talbi (2013) extended his previous taxonomy to include mathematical programming, constraint programming and data mining techniques.

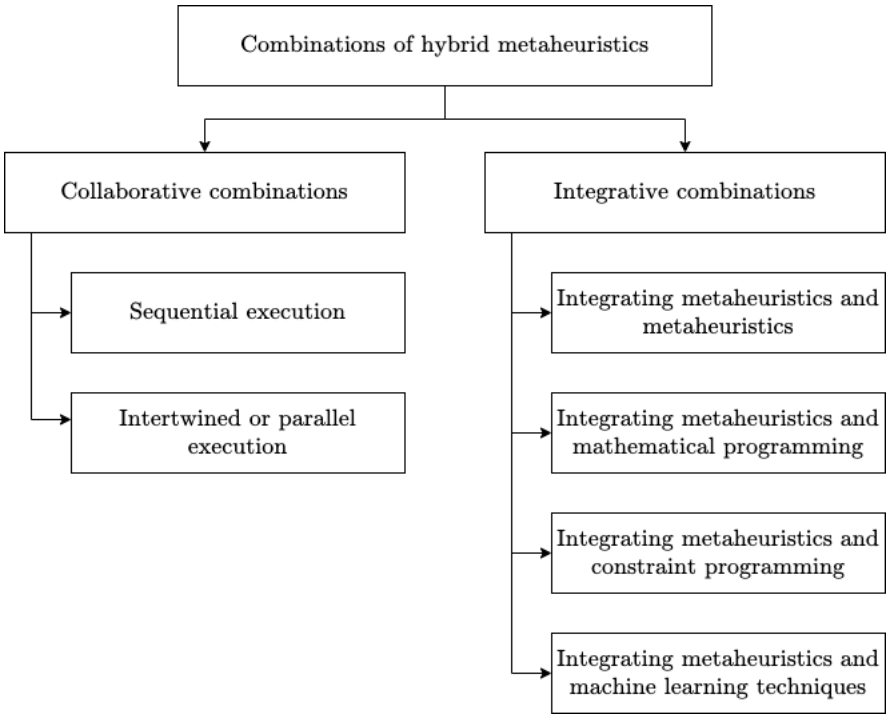


Figure 3.6: An extended hybrid metaheuristics structural classification (Raidl and Puchinger, 2008)

The second part of the taxonomy shown in Figure 3.5, is the flat classification that can be described as:

- Homogeneous or heterogeneous – With homogeneous hybrids, the island model belongs to the homogeneous hybrid where all the combined algorithms use the

same metaheuristic. In heterogeneous hybrids, the greedy randomised adaptive search procedure (GRASP) method may be seen as a heterogeneous hybrid where different metaheuristics are used;

- Partial or global – For partial hybrids, the problem is divided into smaller sub-problems, each having its own search space to build a viable global solution. For global hybrids the search space is shared and individual metaheuristics broadcast the best solution;
- Specialist or general – Specialist hybrids optimise different problems with a combination of metaheuristics. General hybrids solve the same optimisation problem with the same or different metaheuristics.

3.4.2 Grammar

Talbi (2002) proposed a grammar to generalise the basic hybridisation schemes with notation and classified more than 125 hybrid metaheuristics to show the usefulness of the proposed taxonomy. This makes literature on specific algorithms easier to find and summarises a paper in one notation. Researchers from different fields can understand each others' approach easier by having one taxonomy. Four classifications can be derived from a hierarchical taxonomy with the flat taxonomy options listed inline.

- Low-level relay hybrid – LRH ($A_1(A_2)$) (homogeneous, heterogeneous) (partial, global) (specialist, general): The metaheuristic A_2 is embedded into the single-solution metaheuristic A_1.
- High-level relay hybrid – HRH (A_1+A_2) (homogeneous, heterogeneous) (partial, global) (specialist, general): The self-contained metaheuristics A_1 and A_2 are executed in sequence.
- Low-level teamwork hybrid – LTH ($A_1(A_2)$) (homogeneous, heterogeneous) (partial, global) (specialist, general): The metaheuristic A_2 is embedded into the population-based metaheuristic A_1.
- High-level teamwork hybrid – HTH (A_1, A_2) (homogeneous, heterogeneous) (partial, global) (specialist, general): The self-contained metaheuristics A_1 and A_2 are executed in parallel and cooperate.

The three algorithms selected for hybridising the genetic algorithm are local search, greedy search and tabu search. Their implementation will be discussed in the next chapter.

3.4.3 Hybrid metaheuristics with machine learning

Machine learning can be broadly defined as the development and use of algorithms and statistical models to analyse and draw inferences from patterns in data. Machine learning has been researched for decades and is regarded as a disruptive innovation for businesses today in a range of industries from healthcare to manufacturing (Lee and Shin, 2020). Traditionally, machine learning can be divided into three main categories: unsupervised learning, supervised learning, and reinforcement learning (Lee *et al.*, 2018).

Unsupervised learning is when data that contains many features but is not labelled, is used to learn properties of the dataset. Unsupervised learning is useful in many scenarios, like detecting subtle relationships that a machine would otherwise miss or clustering data into similar groups (Goodfellow *et al.*, 2016).

Supervised learning is when each example in the dataset is also associated with a target or label. It is useful to identify patterns and relationships between the data and the target. The disadvantage of supervised learning is that the machine learning algorithm is usually trained on the labelled examples and only given more labelled examples as it grows in its understanding (Goodfellow *et al.*, 2016).

Reinforcement learning is when one creates an algorithm that tries to find the best strategies to maximise its outcomes. Often it is hard to tell what a good strategy is and a machine learning algorithm might end up making random choices. In some cases, methods such as dynamic programming or Monte Carlo simulation can help a machine discover near-optimal strategies (Sutton and Barto, 2017).

Artificial neural networks, a subset of machine learning, are inspired by biological neural networks. In a biological neural network, input is a voltage in the membrane of a neuron, the output is a signal, and it is the interaction between inputs and outputs that allows for change. Analogous to a neural network, a biological neuron connects to nearby neurons through synapses (connections) which are made by specialised proteins that have spines. As represented graphically in Figure 3.7, the synapses in a biological neural network are made in the dendrites which will connect with the axon (fibre) which will then connect to a postsynaptic cell.

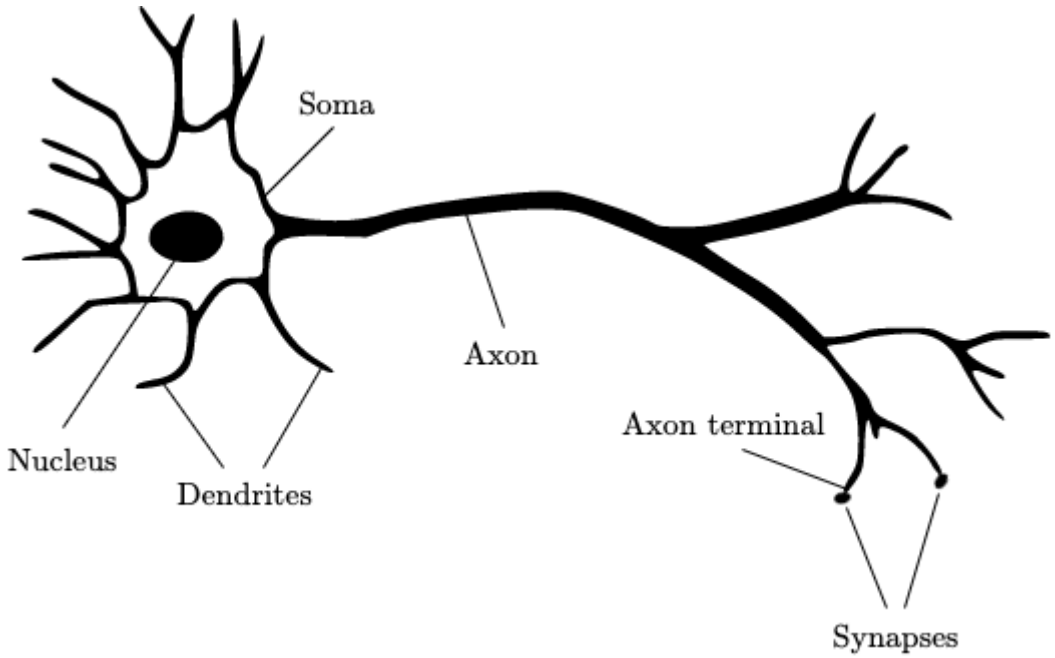


Figure 3.7: Graphical representation of a biological neuron adapted from Looxix (2003)

The first mathematical model of a neural network was proposed by McCulloch and Pitts (1943). This model consisted of a neuron with weights that needed to be set by a human operator. This was followed with the first model of a perceptron by Rosenblatt (1958) that demonstrated the learning of a single neuron. Rumelhart *et al.* (1986) introduced back-propagation, which continues to be a major concept of artificial neural network literature. A neural network consists of an input layer, one or more hidden layers and an output layer. Many concepts in machine learning evolve and are applied with other concepts. One such example is deep neural networks or deep learning that refers to the use of multiple layers between the input and output layers in the artificial neural network, that has been applied with reinforcement learning under the name deep reinforcement learning (Goodfellow *et al.*, 2016; Sutton and Barto, 2017). Over the last 10 years, machine learning has become very popular in many fields, like natural language processing, computer vision, image recognition and robotics. Some of the more interesting results from deep reinforcement learning has been in-game AI (Dargan *et al.*, 2020; Jaderberg *et al.*, 2019; Mnih *et al.*, 2013)

Shukla and Iriondo (2020) list 27 of the most used topologies in neural networks. For simplicity, two topologies are listed with a brief description.

- Multilayer perceptron (MLP) – A class of feedforward (FF) neural networks with at least three layers, one input layer, one or more hidden layers, and one output layer. It is typically used on tabular datasets.

- Long short-term memory (LSTM) – A class of recurrent neural network (RNN) that has feedback connections and typically consists of a cell state, an input gate layer, an output gate layer and a forget gate layer (Hochreiter and Schmidhuber, 1997). It is typically used on sequence datasets.

A typical MLP neural network is shown in Figure 3.8. During training, each input is pushed through the network by taking the scalar product also known as a weighted sum of the input and the weight between the input layer and hidden layer. The scalar product is passed to the activation function that transforms the weighted sum into a desired output. This forward pass is repeated until the output layer is reached with an output. An error rate is calculated by comparing the output with the expected output. The error rate is then used during backpropagation (propagating backwards through the network) which updates the weights in the network one layer at a time, according to the amount the weight contributed to the error rate. This process is repeated with the goal of minimising the error rate. Once the training has completed, the model weights and architecture can be saved for later use. Predictions are made by providing input data to the network and performing a forward-pass in order for the trained model to provide an output (Akyol and Bayhan, 2007).

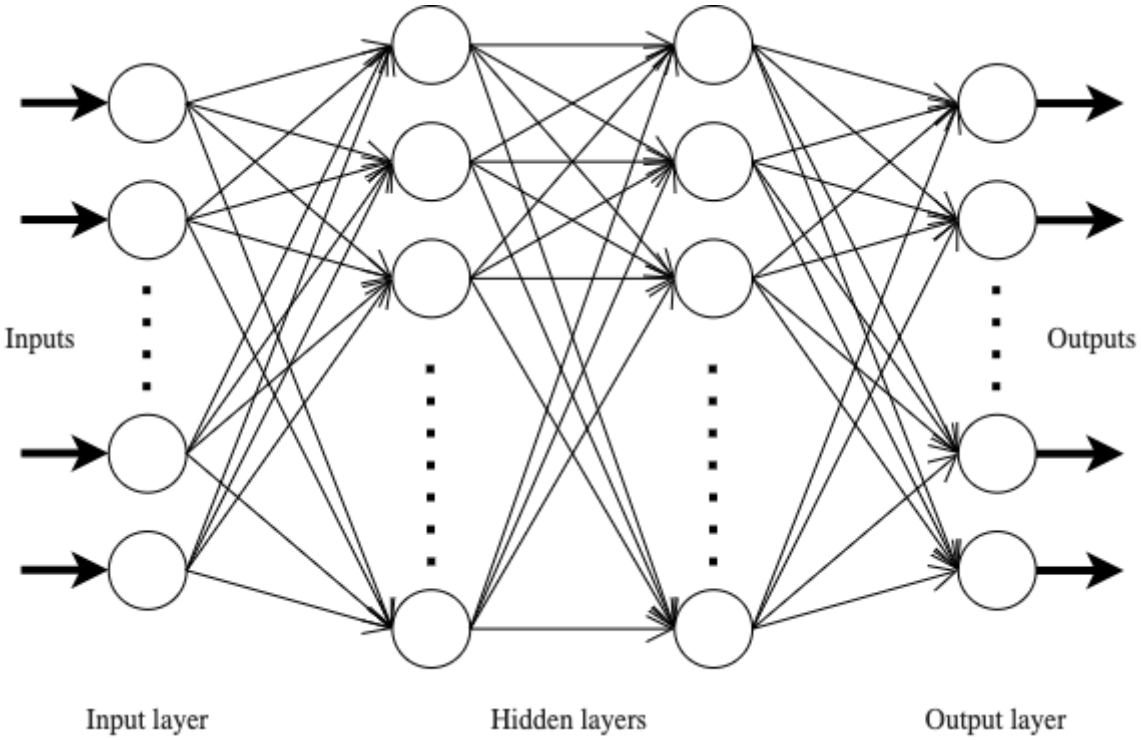


Figure 3.8: A multilayer perceptron (MLP)

One of the advantages of MLP neural networks is that most of the time is spent during data preparation and training so that the prediction can occur in a short period

of time or even in real-time. One of the disadvantages of MLP neural networks is that gradient-based training techniques and weight initialisation have the risk of getting stuck in a local maximum (Akyol and Bayhan, 2007).

Akyol and Bayhan (2007) do not recommend using MLP neural networks for combinatorial optimisation but rather hybridising the neural network with heuristics or metaheuristics to overcome the shortcomings of the individual methods. This can be done by generating training data or searching for the most appropriate design and/or parameter selection using metaheuristics, with the former being the focus of this study. Some hybridisations to generate training data for a neural network have been done by using simulation and have yielded good results (Dunke and Nickel, 2020; Mouelhi-Chibani and Pierreval, 2010; Sabuncuoglu and Touhami, 2002). Talbi (2009) classified this type of hybridisation as a high-level relay hybrid where an approximate model is built using an MLP or a radial-basis function of the objective function from previously evaluated solutions. MLP neural networks have shown to be effective for function approximation although this research area has not attracted much attention (Jin, 2005; Jin *et al.*, 2019).

3.4.4 Parallel metaheuristics

Hybrid metaheuristics are often applied to complex real-life problems. These types of problems can require intensive resources such as computing power and memory. The design of these methods was discussed in the previous section. A brief overview of the implementation of the methods will follow.

Most hybrid metaheuristics run on a single-core processor unless it has been programmed to divide the processes on multiple cores or over a distributed architecture. The high-level teamwork hybrid lends itself to such an implementation. Running hybrid metaheuristics in parallel or distributed computing can speed up the search, improve the quality of the solutions, improve robustness and/or solve large-scale problems (Talbi, 2009). They also suggest that clusters of workstations (COWs), networks of workstations (NOWs) or high-performance computing (HPC) can provide substantial benefits to parallel metaheuristics. Running multiple sequential and/or parallel metaheuristics on distributed computers at the same time with allocated cores and memory can shorten the time required to execute them.

Talbi (2009) named the three major parallel models as algorithmic, iteration and solution level:

- Algorithmic-level – Executing metaheuristics independently or cooperating that is problem-independent with the goal of being more effective.

- Iteration-level – Executing each iteration of the metaheuristics in parallel with the goal of being more efficient.
- Solution-level – Executing a resource-intensive operation in parallel with the goal of being more efficient.

Iteration-level parallel models are relevant to this study. The most resource-consuming part of metaheuristics is the generation of neighbour solutions, evaluating the constraints and calculating the fitness. For the LS, TS, SA and the GA, these are done synchronously. The model waits until all neighbours or chromosomes have been generated, evaluated against the constraints and the fitness calculated. The GS does this asynchronously by generating neighbours, evaluating against the constraints and calculating the fitness until an improved solution has been found.

3.5 Synthesis of literature review on optimisation algorithms

This section reflects on the literature discussed in this chapter. Metaheuristics can be used for finding a near-optimal solution to solve non-tractable problems and provide the flexibility needed to formulate and solve complex real-world industrial-sized problems.

The problem in the study is unique in the sense that metaheuristics do not have a known track record for being applied to this problem in the process industry. The algorithms must be implemented from pseudo-code with the problem-specific requirements. First, the original algorithms selected for the literature review were tabu search (TS), simulated annealing (SA), genetic algorithm (GA), cross-entropy method (CEM) and population-based incremental learning (PBIL). From the literature, the sampling for the CEM and probability vector for the PBIL were compelling features but these two algorithms were later removed as they converged to suboptimal results. For the CEM, as the variance decreased, the sampling excluded valid selections from the sample which resulted in early convergence. For the PBIL, moving the relevant characteristics from the population to the probability vector reduced the exploration to the point where the basic GA had produced significantly better results. To improve the results of the remaining algorithms, additional literature was reviewed on hybrid metaheuristics and local search (LS) was added. LS proved to be too slow to generate and evaluate each neighbour. After reviewing additional literature, greedy search (GS) was added for accepting an improvement move before evaluating all the neighbours. Both LS and GS were included as standalone algorithms and also in

the hybrid algorithms. After profiling the performance of the algorithms, additional literature was again reviewed to improve the performance of the algorithms. The algorithms were converted to iteration-level parallel models. Developing and testing 20 algorithms on one server and one notebook with multiple cores would be too time-consuming and from the literature it was learned that it is possible to run them at the same time or in parallel on a high-performance computing (HPC) cluster. The algorithms were adapted and moved to a HPC cluster for further testing.

In order for the problem to reflect the energy demand, an additional objective was added after reviewing additional literature. To further improve the execution speed, additional literature was review on hybridising the algorithms with machine learning methods.

It is not possible to understand certain aspects and complexities from a single literature review, hence the iterative literature review process followed.

3.6 Chapter summary

This chapter started with an introduction to important aspects of single-objective optimisation, followed by a review of a selection of optimisation algorithms that have been developed over the last few decades. A brief overview of multi-objective optimisation was presented, and finally, a review was carried out of hybrid metaheuristics literature with a specific focus on classification, grammar, parallel metaheuristics and machine learning. The succeeding chapter will describe and discuss the construction and implementation of the algorithms on the problem.

Chapter 4

Construction and implementation of algorithms

This chapter discusses important aspects related to construction of the algorithms using the knowledge gained from the literature in the previous chapters to achieve Objective 6. The chapter presents the requirements for the construction and implementation of the algorithms. It starts with the list of selected algorithms, followed by a detailed description of the three types of variables that are used in this study; namely, decision variables, balanced variables and coupling variables. The two representations for the daily interval and the hourly interval are presented with the proposed encoding scheme. This is followed by the daily interval and hourly interval evaluation process needed to evaluate a solution and a combined evaluation process is presented for the proposed discrete-time single non-uniform grid. Finally, a detailed description is given of the parallel model implementations followed by the hybrid implementations used in this study.

4.1 Selected algorithms

Five algorithms, four hybrid algorithms and one multi-objective algorithm have been selected using two different greatest common factor (GCF) intervals; namely, daily intervals and hourly intervals. The algorithms have been selected on the basis that they are well documented and have been researched extensively (Kallrath, 2002). The two intervals require different formulations that will be discussed in the next section. The complete list of algorithms for this study has been tabled in Table 4.1.

Table 4.1: List of algorithms used in this study

	Algorithm name	Interval
1	Local search (LS)	Daily
2	Tabu search (TS)	Daily
3	Simulated annealing (SA)	Daily
4	Genetic algorithm (GA)	Daily
5	Greedy search (GS)	Daily
6	Genetic algorithm (GA) with local search (LS)	Daily
7	Genetic algorithm (GA) with tabu search (TS)	Daily
8	Genetic algorithm (GA) with greedy search (GS)	Daily
9	Genetic algorithm (GA) with multilayer perceptron (MLP)	Daily
10	Non-dominated sorting genetic algorithm II (NSGA-II)	Daily
11	Local search (LS)	Hourly
12	Tabu search (TS)	Hourly
13	Simulated annealing (SA)	Hourly
14	Genetic algorithm (GA)	Hourly
15	Greedy search (GS)	Hourly
16	Genetic algorithm (GA) with local search (LS)	Hourly
17	Genetic algorithm (GA) with tabu search (TS)	Hourly
18	Genetic algorithm (GA) with greedy search (GS)	Hourly
19	Genetic algorithm (GA) with multilayer perceptron (MLP)	Hourly
20	Non-dominated sorting genetic algorithm II (NSGA-II)	Hourly

4.2 Algorithm variables

Different variable types that serve different purposes are used in the algorithms. Three types of variables will be discussed in this section, namely decision variables, coupling variables and balancing variables. The product margins that are used have been derived from public data and are outlined in Appendix A.

4.2.1 Decision variables

Decision variables are manipulated during the search procedure to see if the change in decision variables improved the fitness of the solution. Each plant or tank in Figure 4.1 will be represented by one or more decision variables, which are listed in Table 4.2. All the decision variables are represented in a vector and the vector for one interval is expressed as,
$[P_{3,2}, P_{4,2}, ..., P_{2,2}A, P_{2,2}B, P_{2,2}C, ..., T_{2,2}In, T_{2,2}Out]$. As discussed in **Chapter 2**, a discrete-time model is selected for this study. For every interval in the time horizon, the vector is extended by the same decision variables.

The decision variables in Table 4.2 have a minimum and a maximum value and the operating range can either *range* between the minimum and maximum or have *fixed* values which will be specified in the maximum column. This is a preserving constraint handling strategy to only include the operating range, thus reducing the number of constraint violations. As a range example, the operating range for plant $P_{4,2}$ starts at a minimum of 24 t/h to a maximum of 28 t/h with an increment of 0,5 t/h and can be represented as $P_{4,2} \in \{24; 24{,}5; 25; 25{,}5; 26; 26{,}5; 27; 27{,}5; 28\}$. Using a 0,5 t/h increment provides adequate accuracy on the operating range. As a fixed range example, the fixed operating range for $T_{2,2}In$ can be represented as $T_{2,2}In \in \{0, 8, 15\}$. A complete set of decision variables is generated for each interval to allow for manipulation of individual intervals. An example would be if planned maintenance is required for plant $P_{4,2}$ in interval 10. The representation for plant $P_{4,2}$ would then change to $P_{4,2} \in \{0\}$ for interval 10.

As specified in **Chapter 1**, two time horizons are required for this study: for the daily interval the time horizon is 90 days, while for the hourly interval it is 14 days. The daily time horizon decision variables will be discussed in the next section, followed by the hourly time horizon decision variables.

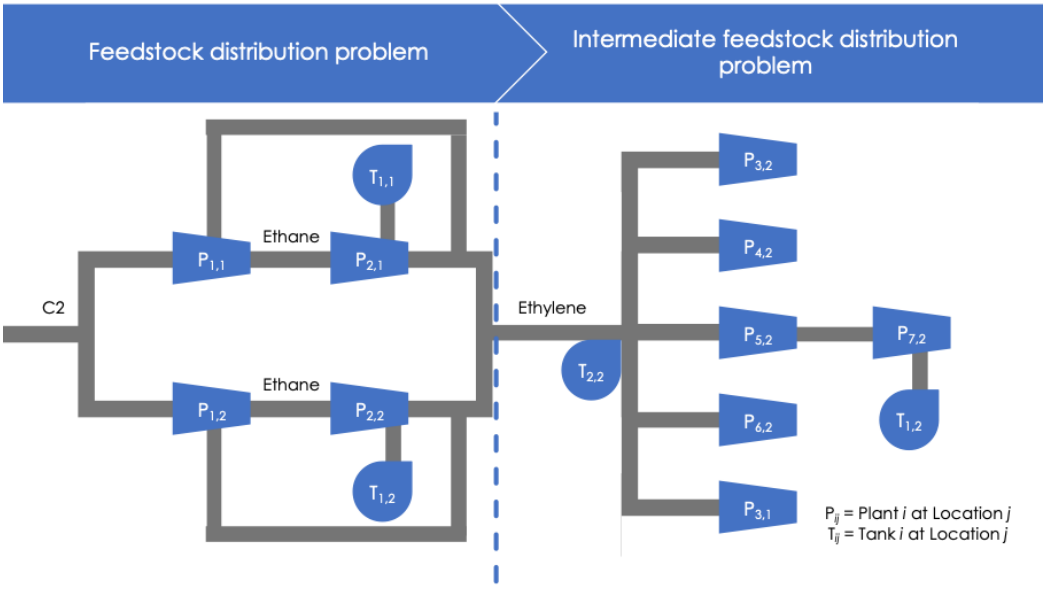


Figure 4.1: C_2 value chain

Table 4.2: Excerpt of the decision and related variables based on Appendix A

	Name	Margin	Start up cost	Min	Max	Yield	Type	UOM
0	$P_{3,2}$	R6 956	R1 000 000	14	19,5	0,95	Range	t/h
1	$P_{4,2}$	R7 337	R1 000 000	24	28	0,95	Range	t/h
7	$P_{2,2}A$	R3 049	R1 000 000	0	19	0,95	Range	kNm^3/h
8	$P_{2,2}B$	R3 049	R1 000 000	0	19	0,95	Range	kNm^3/h
9	$P_{2,2}C$	R3 049	R1 000 000	0	19	0,95	Range	kNm^3/h
20	$T_{2,2}In$	(-)R1 000	R0	0	[8,15]	1	Fixed	t/h
21	$T_{2,2}Out$	(-)R100	R0	0	50	1	Range	t/h

4.2.1.1 Daily decision variables

The representation of a daily interval decision variable could be the total tons produced in a day or the average flow rate for the day. To represent the total tons produced for a day, the plant $P_{4,2}$ can produce at the minimum flow rate for one hour to the maximum flow rate for 24 hours. The total tons for $P_{4,2}$ for a day can be represented as $P_{4,2} \in \{24, ..., 672\}$. This is a large set and provides high accuracy at a computational cost. To reduce the search space, the average flow rate for the day is used and the values $P_{4,2}$ can be represented as $P_{4,2} \in \{24; 24{,}5; 25; 25{,}5; 26; 26{,}5; 27; 27{,}5; 28\}$. The researcher compared the different representations and found the accuracy of the average flow rate sufficient for the daily interval.

Using a 90-day time horizon with 22 decision variables will result in a vector that contains 1 980 decision variables. Selecting a value in the range of each decision

variable for the 90 days will result in one candidate solution. Figure 4.2 illustrates a complete first interval or portion of the completed vector with the selected candidate. The decision variables are expanded vertically to illustrate the scope and selection. Note that the unit of measurement (UOM) "kNm^3" is read as "kilo-normal cube metres".

The two types of decision variables are defined as

$P_{i,j}$ = Plant i at Location j

$T_{i,j}$ = Tank i at Location j

A plant or tank can be represented by multiple decision variables, as follows:

$P_{2,2}$ A = Plant 2 at Location 2 and Furnace A

$P_{2,2}$ B = Plant 2 at Location 2 and Furnace B

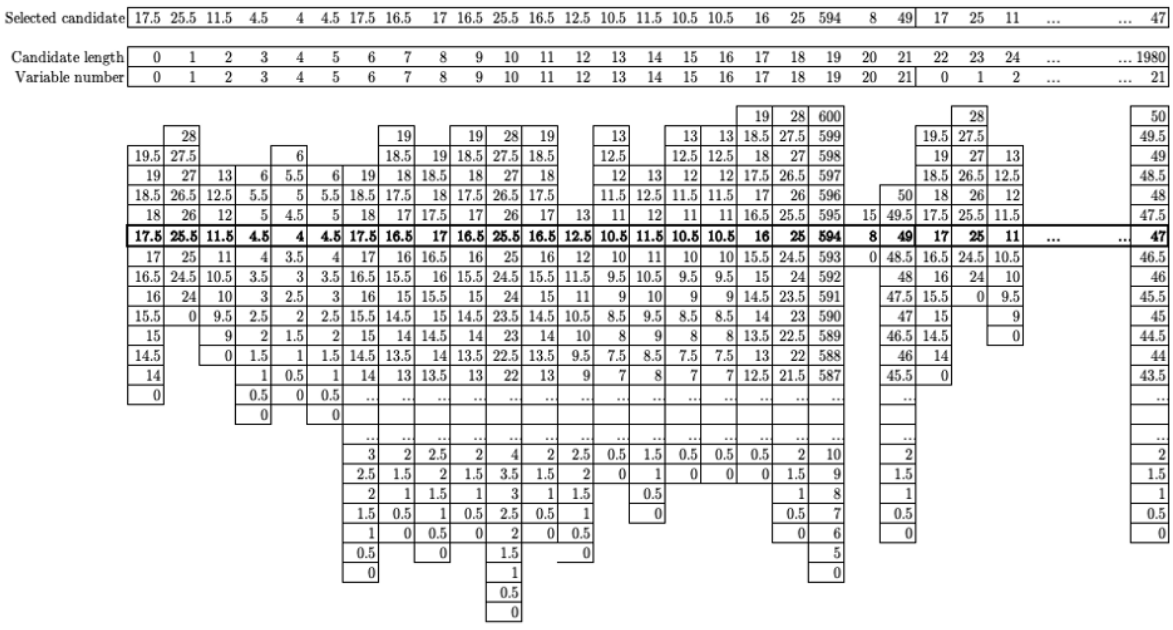


Figure 4.2: Values for daily interval decision variables

4.2.1.2 Hourly decision variables

The plant $P_{4,2}$ example from the previous section has an operating range for an hour starting at a minimum of 24 t/h to a maximum of 28 t/h with an increment of 0,5 t/h which can be represented as

$P_{4,2} \in \{24; 24{,}5; 25; 25{,}5; 26; 26{,}5; 27; 27{,}5; 28\}$. Using the flow rate representation for the hourly interval is sufficient. One limitation exists with this representation in that most plants cannot increase from the minimum rate to the maximum rate or decrease from the maximum rate to the minimum rate in one hour. Increasing or decreasing

the rate must happen gradually and the allowed rate increase is unique to each plant. Table 4.3 indicates the delta minimum and delta maximum value for each decision variable. Continuing with $P_{4,2}$ as an example, the maximum increase of the rate is three. It implies that if the plant is currently operating at 26 t/h, the maximum increase allowed is three positions higher. The plant can only increase from 26 t/h by three levels higher to 27,5 t/h as illustrated in Figure 4.4. Should the next interval decision variable recommend another three-step increase, the decoding process will not be able to move outside the operating range of the plant $P_{4,2}$. It will move to the maximum possible position available which in this case is one step as illustrated in Figure 4.4. The proposed step candidate is then repaired with the feasible step. Using a decoding repair constraint handling strategy reduces the number of constraint violations. The encoding scheme is done at a computational cost; however, the alternative required rebuilding the decision variables at each iteration and interval. Figure 4.3 illustrates a complete first interval or portion of the completed vector with the selected encoded candidate. The decision variables are expanded vertically to illustrate the scope and selection. Note that the UOM "kNm^3" is read as "kilo-normal cube metres".

Table 4.3: Excerpt of decision variables with hourly limits

	Name	Min	Max	Type	Delta Min	Delta Max	UOM
0	$P_{3,2}$	14	19.5	Range	-1	3	t/h
1	$P_{4,2}$	24	28	Range	-1	3	t/h
7	$P_{2,2}A$	0	19	Range	-1	4	kNm^3/h
8	$P_{2,2}B$	0	19	Range	-1	4	kNm^3/h
9	$P_{2,2}C$	0	19	Range	-2	5	kNm^3/h
20	$T_{2,2}In$	0	[8,15]	Fixed	-1	1	t/h
21	$T_{2,2}Out$	0	50	Range	-2	6	t/h

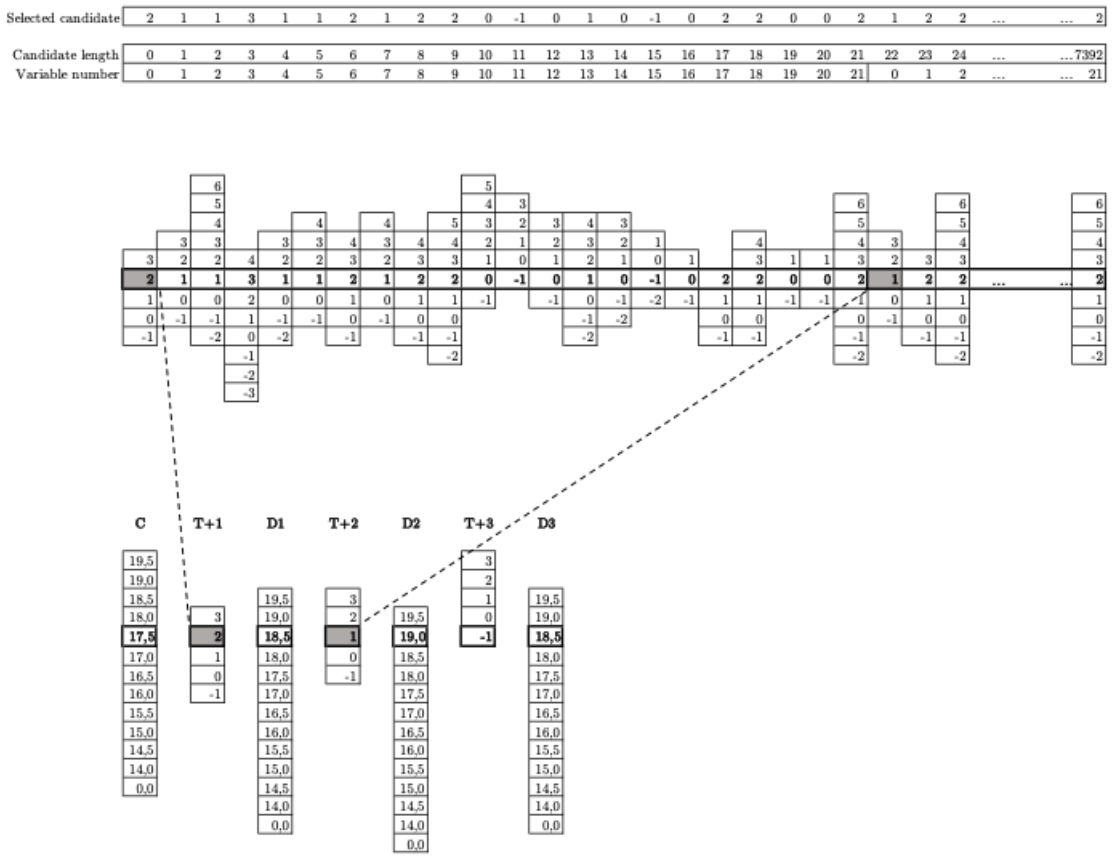


Figure 4.3: Step encoding decision variables

P4,2

$dv1_{t=0}$		$t+1$	$dv1_{t=1}$		$dv1_{t=1}$		$t+2$	$dv1_{t=2}$
						▷	+3	
				⟩			+2	
28,0			28,0		28,0	▷	+1	**28,0**
27,5	▷	+3	**27,5**	⟩	**27,5**		+0	27,5
27,0		+2	27,0		27,0		-1	27,0
26,5		+1	26,5	⟩	26,5			26,5
26,0		+0	26,0		26,0			26,0
25,5		-1	25,5	⟩	25,5			25,5
25,0			25,0		25,0			25,0
24,5			24,5	⟩	24,5			24,5
24,0			24,0		24,0			24,0
0,0			0,0	⟩	0,0			0,0

Figure 4.4: Example of step encoding proposed

4.2.2 Balancing variables

In **Chapter 3**, constraint-handling strategies were discussed and introduced *the balancing period*. All the decision variables are reconciled to the balancing variables. Both negative and positive imbalances need to be actioned to resolve an imbalanced network. The following steps show an example to resolve an imbalanced network:

1. Increasing the rate of the consuming plants within the feasible region;
2. Storing the excess product in the pipeline;
3. Liquefying the feed to a liquid and storing it in a tank;
4. Flaring the excess supply.

Step 1 and step 3 represent decisions that can be made and step 2 and step 4 represent the consequence of imbalance. The difference required to balance the network is stored in balance variables. Flaring the product in step 4 is the most costly option and is only done after all other steps have been considered.

The balancing variables used in this study are listed in Table 4.4.

Table 4.4: Balance variables with cost based on Appendix A

	Name	Cost	UOM
0	Ethylene pipeline delta	0	kPa
1	Sasolburg ethylene flaring	R3 049	t/h
2	Secunda ethylene flaring	R3 049	t/h
3	C2 pipeline delta	0	kPa
4	Sasolburg ethane flaring	R991	t/h
5	Secunda ethane flaring	R991	t/h
6	Secunda C2 flaring	R200	kNm^3/h

4.2.3 Coupling variables

Coupling between time intervals can be done by using inventory variables (Jackson *et al.*, 2003). Coupling variables used in this study are listed in Table 4.5 and include storage tanks levels and pipeline pressures. These variables are used to close each interval and are used as the opening balance for the next interval. The balancing variables use the inventory in the opening coupling variables to balance the network. The remainder of the inventory in the coupling variables is carried over to the next interval.

Table 4.5: Coupling variables with limits

	Name	Min	Max	UOM
0	Sasolburg ethane tank	50	500	t
1	Secunda ethane tank	50	300	t
2	C2 pipeline pressure	100	4 000	kPa
3	Ethylene pipeline pressure	100	5 000	kPa
4	Sasolburg ethylene tank	50	6 000	t

4.3 Objective function

The objective function is used to determine the quality of the solution. It is important to take care when defining the objective function properly to avoid unacceptable solutions (Talbi, 2009). Objective functions in the process industry are frequently profit-related which maximises the flow of products or the return on investment (Zyngier and Kelly, 2012).

The objective function of the problem in this study can be stated as

$$\max \text{ Profit} = \text{Product profit} \tag{4.1}$$
$$- \text{ Start-up cost} \tag{4.2}$$
$$- \text{ Cost of liquefaction} \tag{4.3}$$
$$- \text{ Cost of vaporisation} \tag{4.4}$$
$$- \text{ Cost of flaring} \tag{4.5}$$

The individual terms are calculated as follows:

$$\begin{aligned}
\text{Product profit (4.1)} &= \text{Feed} \times \text{yield or conversion} \times \text{margin of the product} \\
\text{Start-up cost (4.2)} &= \text{Cost for starting a plant} \times \text{the number of times started} \\
\text{Cost of liquefaction (4.3)} &= \text{Cost of liquefying gas} \times \text{amount of product liquefied} \\
\text{Cost of vaporisation (4.4)} &= \text{Cost of vaporising liquid} \times \text{amount of product vaporised} \\
\text{Cost of flaring (4.5)} &= \text{Margin of product} \times \text{amount of product flared.}
\end{aligned}$$

The product profit (4.1) is calculated for each plant by taking the feed it receives and multiplying it by the yield or conversion. This will give a production output for the plant. The profit is calculated by multiplying the output by the margin for the product produced.

Each time a plant is started, gas is vented or flared and steam and electricity are used to heat the plant. Similar costs are associated when shutting a plant down. This can be calculated per plant and combined into one start-up cost for the plant. Kim and Edgar (2014) included different start-up types such as hot, warm and cold that will have different costs with ramp up or ramp down rates and times. This will provide additional accuracy and can be included in future work. This cost of starting a plant (4.2) can be calculated by multiplying the cost for starting and shutting down a plant by the number of times a plant was started.

A gas such as ethylene can be stored in high-pressure storage tanks. The gas is liquefied into the storage tank and vaporised out of the tank. Both processes have a cost associated, which are the cost of liquefaction multiplied by the amount of product liquefied into tanks (4.3) and the cost of vaporising stored liquid multiplied by the amount of product being vaporised (4.4).

To avoid shutting down a producing plant, excess feedstock gas can be burned or flared. The cost of flaring (4.5) is calculated by the amount of product flared multiplied by the margin of the product.

4.3.1 Second objective function

An additional objective function; namely, energy consumption, is used for the multi-objective non-dominated sorting genetic algorithm II (NSGA-II). The energy consumption for a plant consists of electricity, high-pressure steam, low-pressure steam and fuel gas. The energy consumption deduced from the specific energy consumption (SEC) value is different at a lower production rate than at a higher production rate as illustrated in Figure 4.5. The chemical plants in the value chain have different SEC values and when multiplied by the production rate expresses the energy consumed for that production rate (Saygin *et al.*, 2009). The total energy consumed can be calculated by adding up the energy consumed for each individual plant as shown in (4.6).

$$\min \; Energy_{Total} = \sum SEC^{P}_{P_{i,j}} \times P_{i,j} \tag{4.6}$$

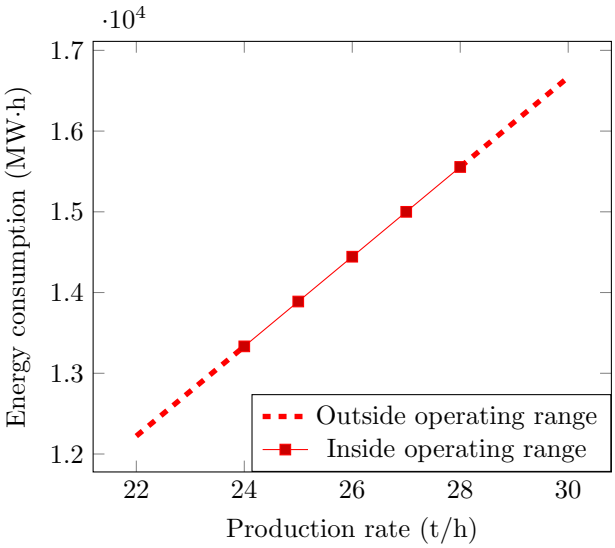


Figure 4.5: Energy consumed vs production rate for $P_{4,2}$

4.4 Solution evaluation

Every solution that has been generated initially or as part of a search procedure needs to be evaluated and this is done using an evaluation process. The *daily interval evaluation process* is illustrated in Figure 4.6. The opening variables contain the current state for both the decision variables and coupling variables. The opening variables ensure that the proposed candidate is anchored to the current state. The process starts with the complete candidate solution. The first interval from the completed candidate solution is passed with the opening variables to be evaluated against the constraints. Handling of a constraint violation was discussed in Section 3.1.5 and Section 4.2.2. Evaluating constraints and balancing the network has been labelled as the balancing period. The imbalances during the balancing period are stored in the balancing variables. The objective function discussed previously is used to calculate the profit. The profit is calculated using both the first interval of the candidate and the balancing variables containing the amount flared during the first interval. The coupling variables that are needed for the evaluation of the next interval are updated. This process is continued for each interval in the candidate solution. At the end of the evaluation, the profit from each interval is added together and returned as a total profit for the solution.

A variation of this process is needed for the hourly representation. The *hourly interval evaluation process* is illustrated in Figure 4.7. The main difference is the decoding that is needed before the process can start. Once a complete solution has been generated, the completed solution is first decoded as illustrated in Figure 4.4.

After decoding the completed solution, the solution follows the same process as the daily representation.

Figure 2.4 illustrated a proposed discrete-time single non-uniform grid. This will be evaluated by joining the hourly interval evaluation process with the daily interval evaluation process. Figure 4.8 illustrates the combined process needed for the proposed non-uniform grid. The first 72 hours is the critical time in the 14-day horizon. The complete solution candidate is split between the first 72 hourly intervals and the 11 daily intervals. The hourly interval part of the solution candidate is decoded and evaluated using the hourly interval evaluation process. Once completed, the decision variables from the last interval in the hourly interval part of the solution candidate are passed with the ending coupling variables from the hourly interval evaluation process to the daily interval evaluation process as opening variables. The daily part of the solution candidate is evaluated with the opening variables following the daily interval evaluation process.

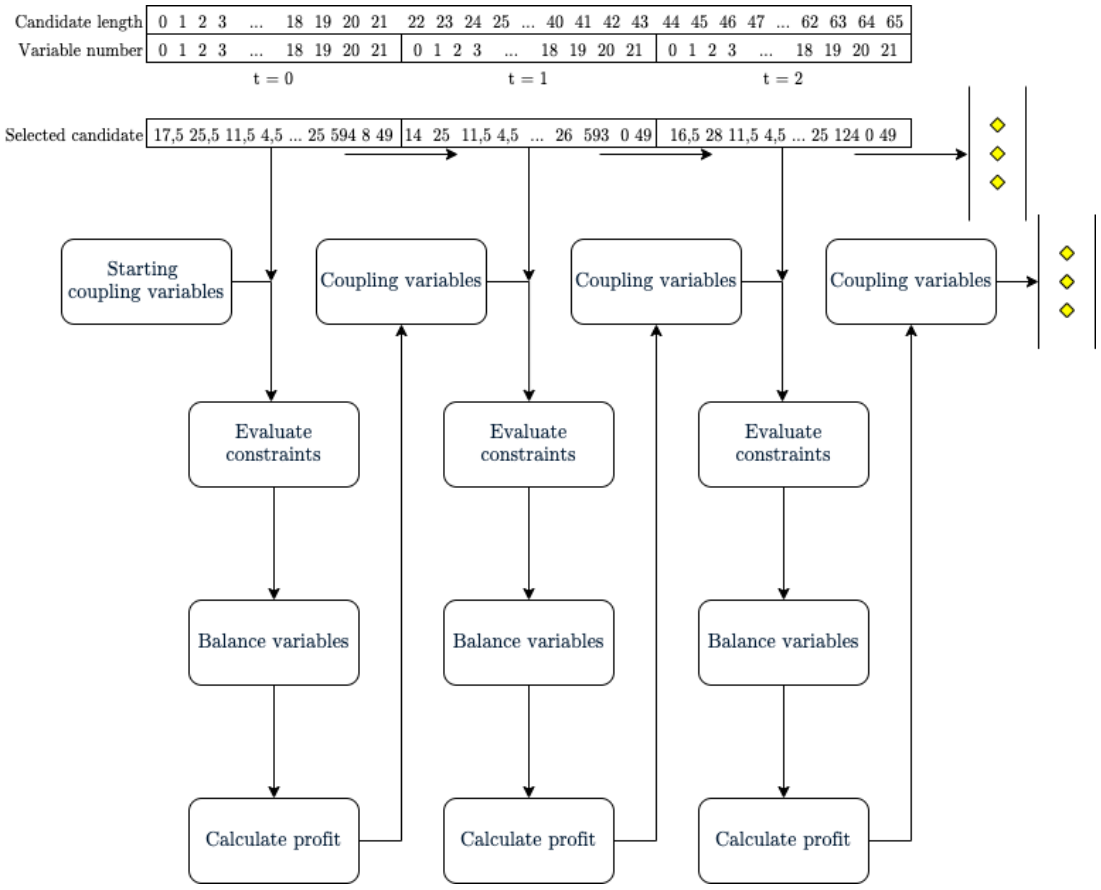


Figure 4.6: Evaluation process for the daily interval

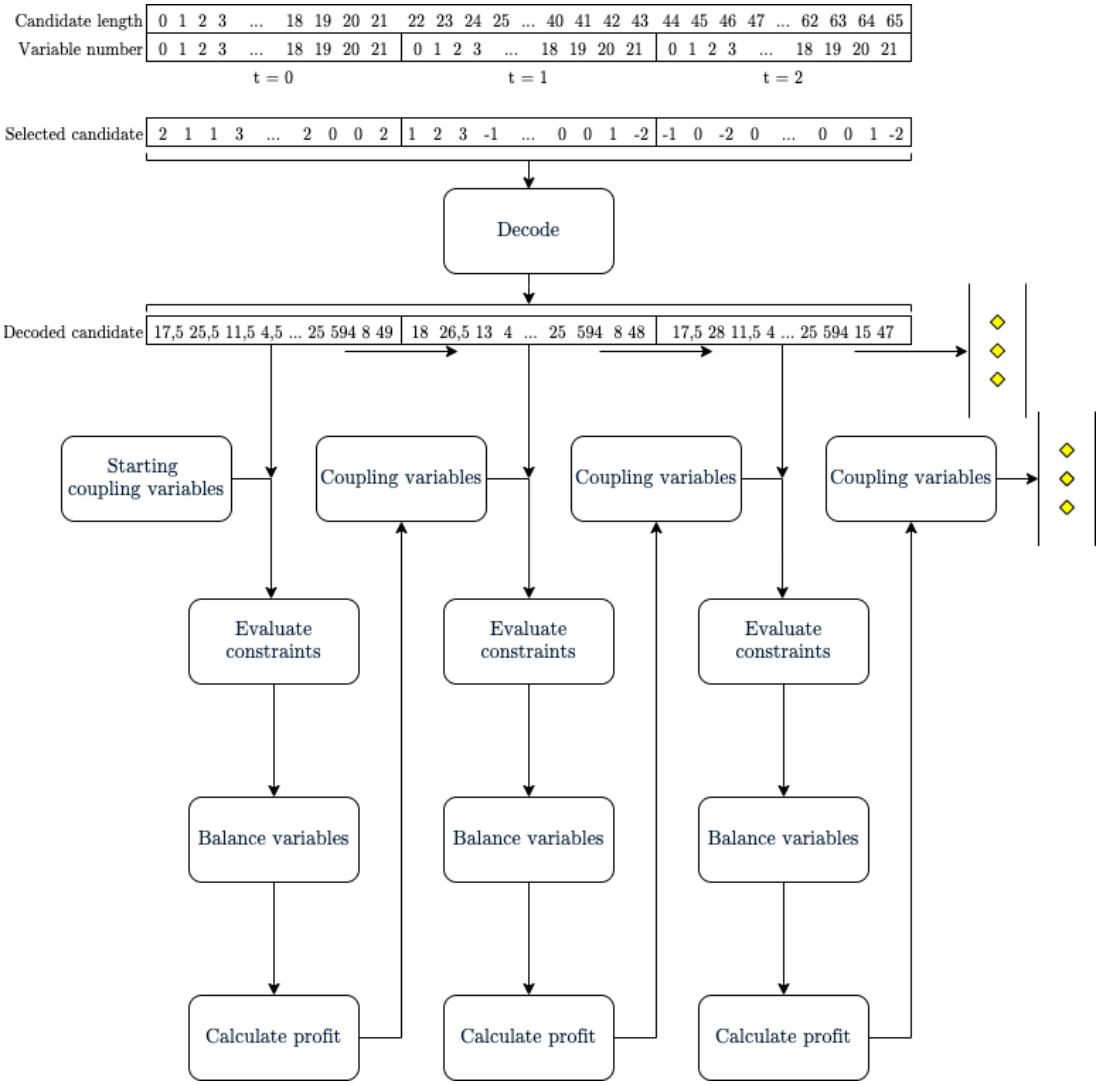


Figure 4.7: Evaluation process for the hourly interval

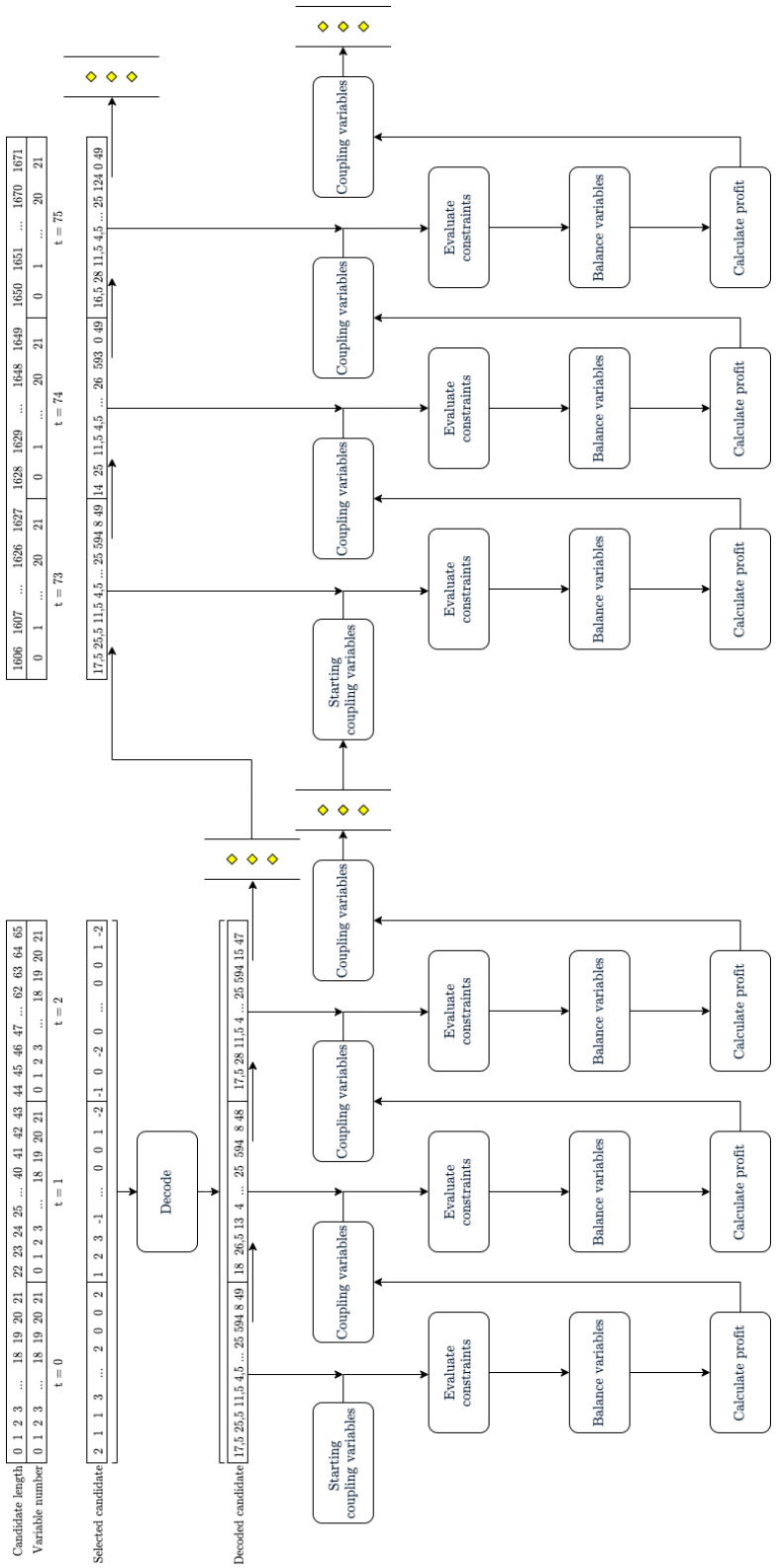


Figure 4.8: Evaluation process for the combined intervals

4.5 Parallel execution

Parallel models allow for computationally expensive steps to be executed in parallel, thus speeding up the optimisation process. Two iteration-level models are used in this study, one that is synchronous and the other asynchronous. Both are used to generate candidate solutions and evaluate the solutions. Figure 4.9 illustrates the synchronous iteration-level process for tabu search (TS). The same synchronous process is used for local search (LS), TS, simulated annealing (SA) and the genetic algorithm (GA) and the implementation can be described together. A pool of items is created based on the number of neighbouring solutions needed or the size of the population needed. The number of cores used by the algorithm can be specified in the code, limited by the host machine or allowed to use all available cores. Each core is considered to be a worker and will fetch an item from the pool. The worker will generate the candidate solution and follow an evaluation process to establish the profit. All the solutions are returned once all the items in the pool have been completed. The rejected solutions are ignored and the remaining solutions are sorted from most profitable to least profitable. The algorithms then continue with execution and the succeeding steps are then uniquely implemented to each algorithm.

The asynchronous iteration-level model is used for the greedy search (GS) algorithm. The number of items is generated based on the number of neighbouring solutions needed and placed in a process queue. The worker fetches an item from the queue, generates a candidate solution and follows the evaluation process to establish the profit. After completing the evaluation, the profit is returned and the next item in the queue is selected. As the profit for each candidate solution is returned, the profit is compared with the current solution profit. If the current solution profit is greater than the candidate solution, then the candidate solution is ignored and the algorithm waits for the next candidate solution profit to be returned. If the next candidate solution profit is larger the current solution profit, the candidate solution is accepted and the search process is abandoned.

The main difference between the two model implementations is waiting for all the items to be completed versus waiting until an improved solution is found.

Figure 4.9: Tabu search (TS) with iteration-level parallel execution

4.6 Hybrid metaheuristic implementations

In the literature review in Section 3.4 it was mentioned that the combination of different metaheuristics can improve the solution quality. The GA is used as the base for the hybridisations and is implemented in three ways.

The *first type* of hybridisation is intended to improve the final solution of the GA. The high-level relay hybrid implementation uses the final solution from the GA as the starting point for the relay algorithms. The exploration of the GA can end in a good neighbourhood but small exploitations of the neighbourhood might improve the solution. The relay algorithms local search (LS), greedy search (GS) and tabu search (TS) can complement the final GA in different ways. Being in a good neighbourhood, the local search can evaluate all the neighbours and move to the one with the biggest improvement and repeat the process if necessary. Due to the size of the candidate solution, that can be too time-consuming and using a parallel model can improve the speed. A greedy search could perform better by moving to an improved solution without the evaluation of all the neighbouring solutions. Lastly, a TS algorithm might find an improved solution in a nearby neighbourhood by escaping the current neighbourhood.

The *second type* of hybridisation is by using an LS algorithm for a mutation operator. The value of the mutation operator is fixed at 0,07 for this study. A random value is generated and if it is smaller than the value of the mutation operator, the selected chromosome is passed to the LS algorithm and set as the starting point for the search. This will introduce an exploited chromosome that can contain elite features, back into the population.

The grammar for the hybrid algorithms selected for this study is listed below.

- HRH(LTH(GA(LS)) + LS (hom,glo,gen)) (hom,glo,gen) – Genetic algorithm 3.2.4 using local search algorithm 3.2.1 for mutation with a relay to local search algorithm 3.2.1.
- HRH(LTH(GA(LS)) + TS (hom,glo,gen)) (hom,glo,gen) – Genetic algorithm 3.2.4 using local search algorithm 3.2.1 for mutation with a relay to tabu search algorithm 3.2.2.
- HRH(LTH(GA(LS)) + GS (hom,glo,gen)) (hom,glo,gen) – Genetic algorithm 3.2.4 using local search algorithm 3.2.1 for mutation with a relay to greedy search algorithm 3.2.5.

The *third type* of hybridisation is using the GA to train a multilayer perceptron (MLP) neural network for the hourly interval. The neural network (NN) then predicts

values for the feedstock distribution problem and intermediate feedstock distribution problem illustrated in Figure 1.4. The hybridisation is divided into two parts as illustrated in Figure 4.10; namely, training and prediction.

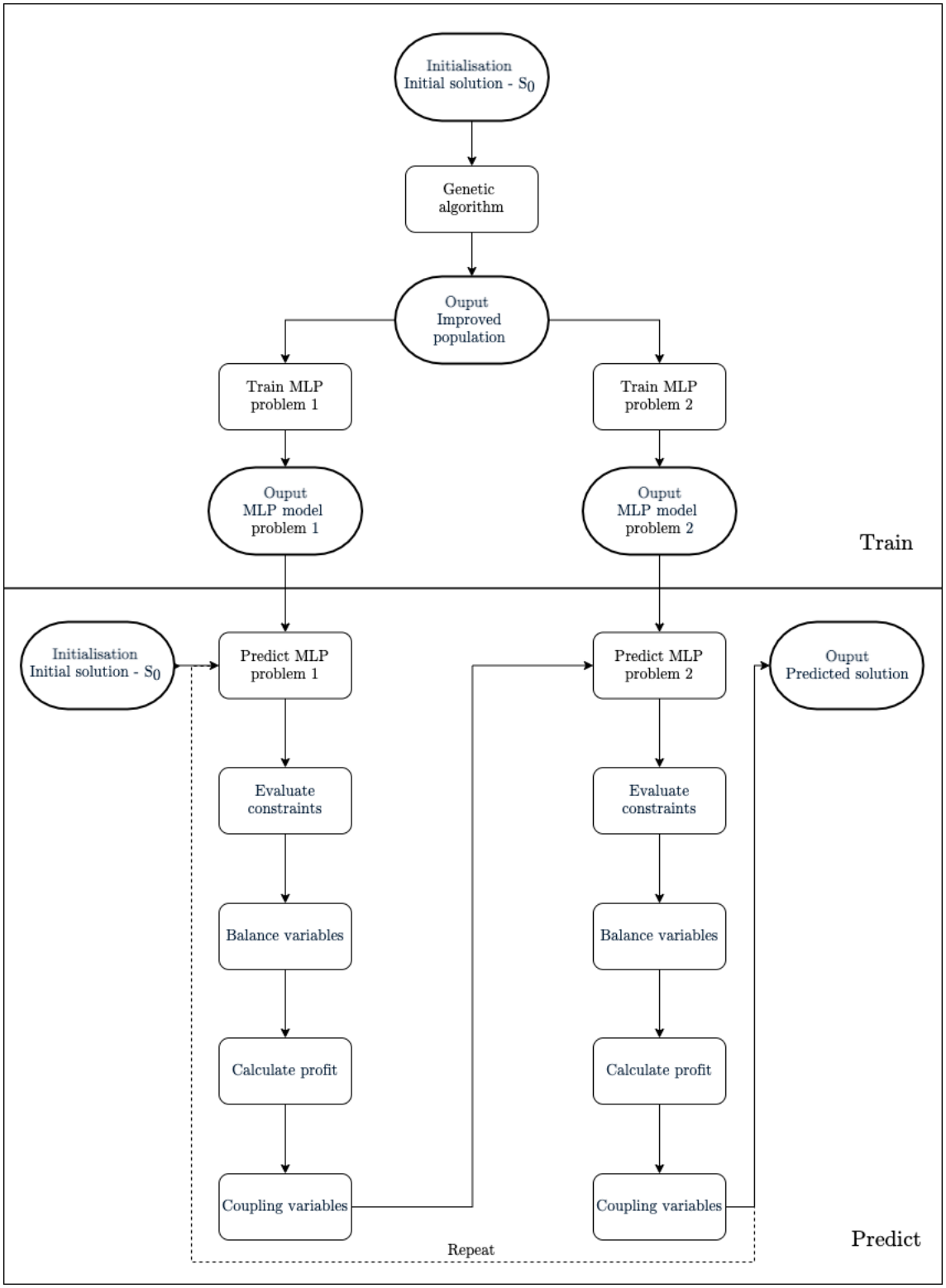


Figure 4.10: Genetic algorithm (GA) integrated with an multilayer perceptron (MLP) neural network

The training dataset is generated using the final population from GA. Running the GA once will only provide one population and to have a representative population,

different scenarios are used in the GA. The changes that can be made to the scenario include:

- Varying the feedstock rates;
- Varying the plant availability by setting the production rate variable to a value of 0 for a specified time interval;
- Using different initial plant rates;
- Using different initial coupling variable values.

The parameters for the GA were set to 1000 for the population size and 500 generations. The top 20% from each final population with the coupling and balancing variables are included in the training dataset.

The training dataset is then prepared for the two problems. First, the multi-period data is flattened to input values and targets. This is done by taking two consecutive periods in the data and converting the first period values as the input values and the second period values as the target or predicted values. Thereafter, features are selected from the dataset that are relevant to each problem. Custom features are then constructed from existing data. The features include combinations of values that are constrained, individual and combined profit contribution, and cost incurred. Finally, the model is fitted on the training dataset and evaluated on the test dataset.

The prediction part starts with the opening variables that contain the current state for both the decision variables and coupling variables. This is passed to the model for problem one in Figure 1.4; the model predicts values for the first period and passes the values to be evaluated. The evaluation follows the same process as described in Section 4.4 omitting the evaluation for problem two in Figure 1.4. After the evaluation for the first problem, the values for the first model with the balancing and coupling variables are passed to the second model for problem two. The second model predicts values for the first period and passes the values for evaluation. The evaluation is then performed in respect of problem two. Once completed, the process is repeated for the remaining time periods to provide a completed chromosome.

The crucial component of this hybridisation is the neural network models. The two models are both MLP neural networks which allow for various configurations. The list of configurations consists of the number and shape of hidden layers, connectivity between layers, the activation function, the initialiser, the optimiser and the loss function. Different configurations were experimented within this study and the final version is summarised next.

The two models use five dense hidden layers with the last layer having 13 and eight output nodes respectively. A mean squared error loss function is used together with the Adam algorithm as the optimiser. A rectified linear unit (ReLU) activation function is used, with layer weights being initialised uniformly. The MLP model used for problem two in Figure 1.4 is illustrated in Figure 4.11.

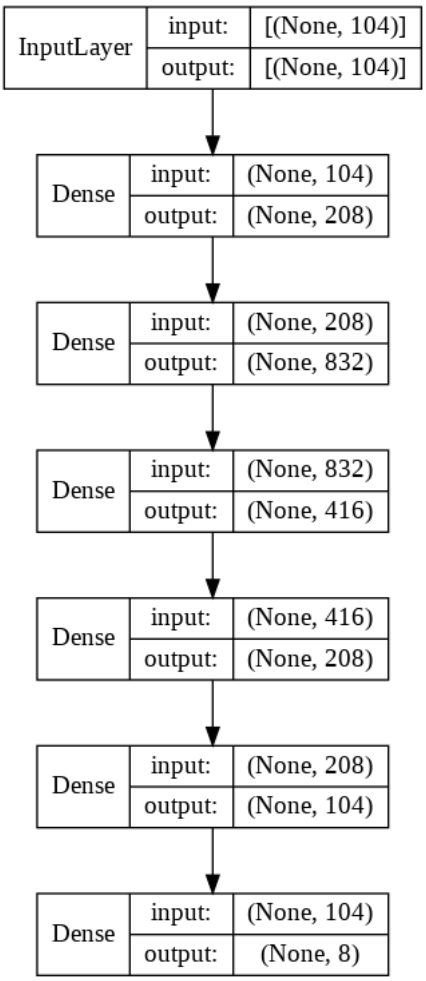


Figure 4.11: The multilayer perceptron (MLP) model used for problem 2

This concludes the description of the hybrid implementations.

4.7 Chapter summary

The chapter started with detailed descriptions of the three types of variables, decision variables, balanced variables and the coupling variables that are used in this study. The two representations for daily and hourly intervals are presented with the proposed encoding scheme, preceded by a discussion of the objective functions. This was followed by the daily and hourly interval evaluation process needed to evaluate

the quality of a solution and a combined evaluation process was presented for the proposed discrete-time single non-uniform grid. Finally, a detailed description was given of the parallel model implementations followed by the hybrid metaheuristics implementations used in this study. The algorithms form the building blocks of the decision support system (DSS) and in the succeeding chapter, a discussion of the design and implementation of the DSS.

Chapter 5

System design and implementation

Chapter 4 presented the construction and implementation of the algorithms for this study.

This chapter focuses on the design and implementation of the decision support system (DSS). Nižetić *et al.* (2007) describe a DSS as an information system aimed to support human decision-making. Relevant to this study, a model-driven DSS that models the decision problem and uses optimisation or analytical tools to suggest actions is developed. Four components are typically required for a model-driven DSS namely, a user, a user interface, a model base and a database. In the next section, a discussion is presented on the proposed model-driven DSS architecture, followed by an analysis of these components.

5.1 System architecture

When using a DSS, the decision process should at least affect one of the PAIRS (productivity, agility, innovation, reputation, satisfaction) in a positive manner (Burstein and Holsapple, 2008). The PAIRS should be considered when deciding on an architecture. An uncomplicated architecture should be easier to maintain, scalable to accommodate other algorithms and permit segregated development by multiple developers.

The four components needed for a model-driven DSS can be described as:

1. User – A user or subject matter expert (SME) who uses the system;
2. User interface -– The interaction and communication point of the system between the user and system;
3. Model base -– Optimisation or analytical tools suggesting decisions in the decision-making process;

4. Database – Database management system (DBMS) for data collection, storage and presentation.

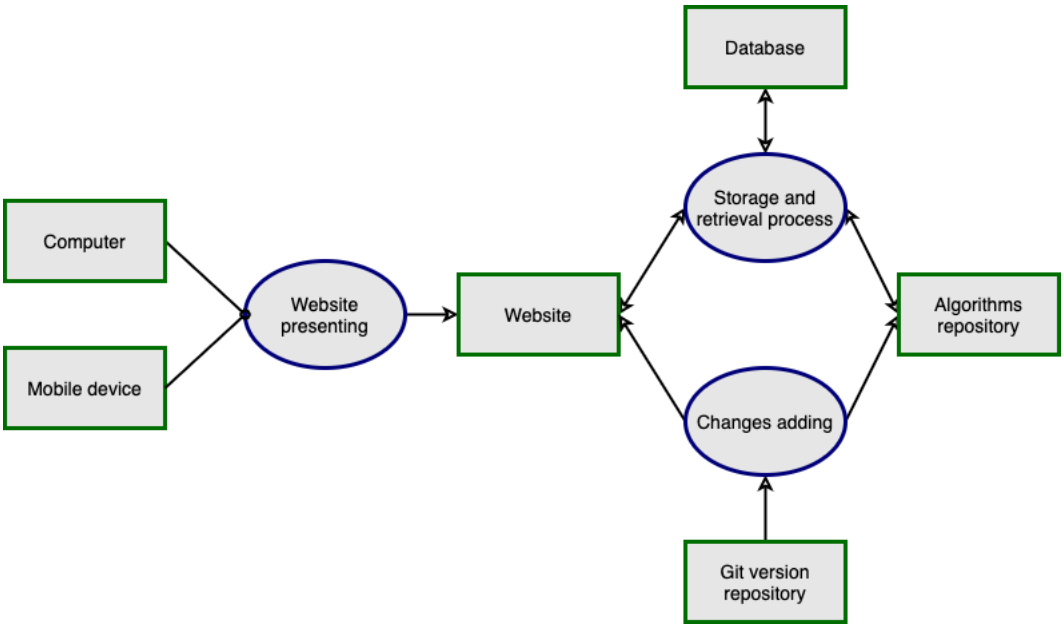


Figure 5.1: Proposed decision support system architecture

A proposed decision support system architecture is presented in Figure 5.1. The architecture diagram is presented using the object-process methodology (OPM) modelling paradigm. The modelling paradigm is described by the international organization for standardization (ISO) in ISO/PAS 19450. This standard enables practitioners and vendors to utilise the concepts, semantics and syntax for automation systems and integrations (ISO, 2015). Two modalities can be produced simultaneously, the object-process diagram (OPD) in Figure 5.1 and the describing text generated using object-process language (OPL) from the OPD. The description of the system architecture is as follows:

Website presenting requires Computer or Mobile device.
Website presenting yields Website.
Storage and retrieval process affects Algorithms repository, Website, and Database.
Changes adding consumes Git version repository.
Changes adding yields Website and Algorithms repository.

The different components in a model-driven DSS and the proposed DSS architecture can be described in more detail. The user is an SME that will interact with a computer or a mobile device via the user interface. The interactions that the user can have are to change the variables, start the algorithms with specific values and

read the suggestions provided by the DSS after completing the execution. The user interactions will be done on a web-based user interface. The website will present data from the database to allow for the interactions by the user. The database acts as the intermediary between the model base and the user interface. The data in the database can be grouped into two sections, namely the user interface data and the model base data. The model base contains all the algorithms in the system. The model base continuously checks the database to see if *jobs* or optimisation tasks have been scheduled for execution. The model base communicates with the database using an application programming interface (API). The model base fetches the job data needed for execution from the database, and on completing the job the model base returns the results to the database. The code used for the user interface and the model base is updated from a git repository. The git repository keeps track of all the changes made and can be integrated to allow for automatic deployment. A git repository has been included in the architecture to allow for scaling the system. The git repository can be used by one developer and one production environment as has been done in this study but in many scenarios, more than one developer can work on the code and they can work over multiple environments. The environments can include development, testing and production or live environment.

Separating the user interface and the model base provides additional options. For smaller deployments or a less processing-intensive model base, the model base can be deployed on the same server as the user interface. For a more computationally expensive model base, the model base could be hosted on a different high-performance server or deployed over a high-performance computing (HPC) cluster.

A second advantage of separating the user interface and the model base is to allow for different types of developers. A common requirement in large organisations is that a specific DBMS must be used and the user interface must be developed in a certain language or included in a system currently in use. The user interface can be developed by a developer without any background knowledge of the problems being addressed while the model base can be developed by engineers, operations researchers or statisticians. The model base developers can focus on the algorithms and deploy them to the model base, leveraging the existing functionality in the user interface. Using an API to communicate with the database enables the model base developers to use any language or solver that is needed for the problem.

For this study, all the components were developed according to the proposed architecture. The model base was deployed and used on the same server, a notebook and an HPC cluster. In the following sections, the user interface, database and model base will be discussed.

5.2 System user interface

The user interface allows the SME to interact with the system. The interaction is focused around the jobs or optimisation tasks that can be performed on the DSS. The requirements for the user interface are:

- Web-based – Accessible with any modern internet browser.
- Responsive design – Adjust the content for a computer screen size or mobile screen size.
- Database connectivity – Connect to the database on the same server or different server.
- Graph visualisation capability – Render results data from the database in a graph.

The architecture allows for any user interface that meets the previously stated requirements. Table 5.1 lists the technology stack that was used in this study.

Table 5.1: Technology stack used for the user interface

Client-side framework	Bootstrap – HTML, CSS and JavaScript
Client-side charting library	Plotly – JavaScript
Server-side	PHP
Webserver	NGINX
Operating system	Ubuntu 18.04 LTS

Figure 5.2 indicates the navigation options available to the SME. The main activities the SME will perform on the system include maintaining variables, managing jobs and reading the results for a job. The SME will update the variables to reflect the current scenario, create a job for the model base and wait for the results. Once the model base completes the job, the results will be available on the user interface for the SME to read. Each of the activities will be discussed next.

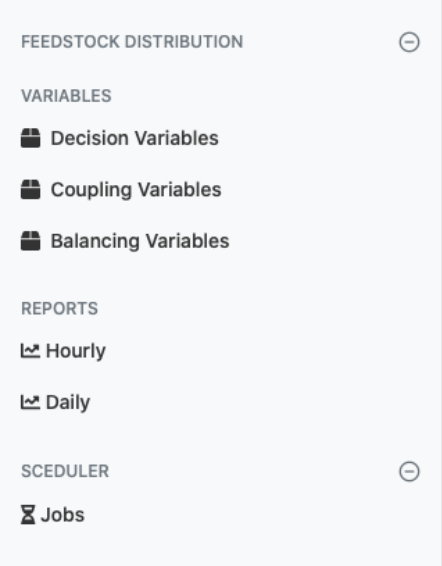


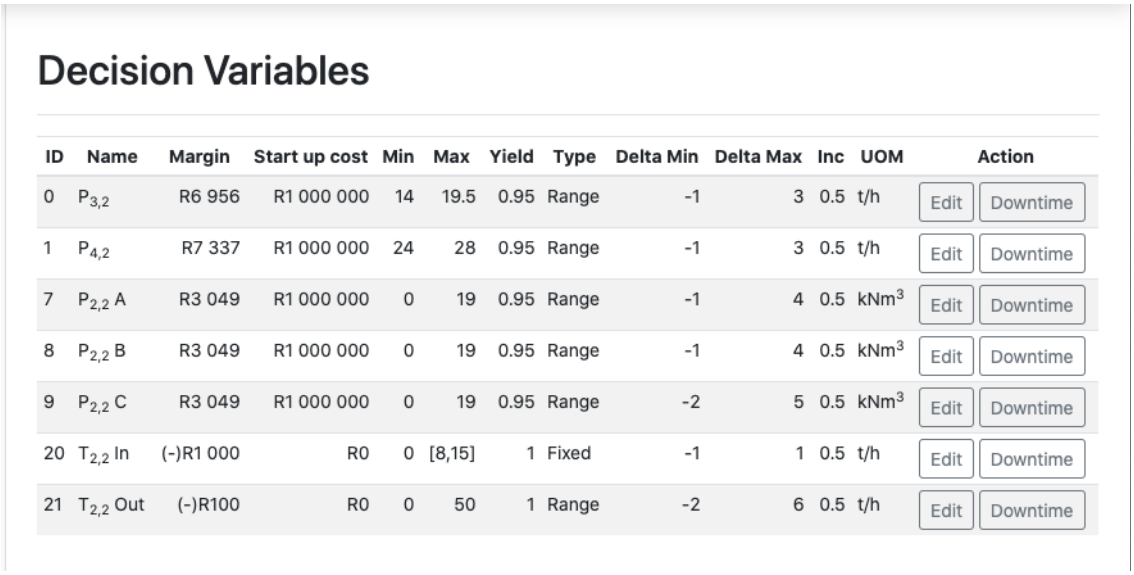
Figure 5.2: Decision support system navigation options

5.2.1 Maintaining variables

Three types of variables are used in the decision support system namely, decision variables, coupling variables and balancing variables. Figure 5.3 illustrates an extracted view that an SME will see in the system for decision variables. For the study, the names have been changed to match the variable names used in previous chapters and values have been adjusted according to the method described in Chapter 4.

Decision variables can be changed to different values for different scenarios. The downtime button allows downtime to be added to the decision variable. The downtime system view is displayed in Figure 5.4 and both planned maintenance or unexpected downtime for the decision variable are added on this display.

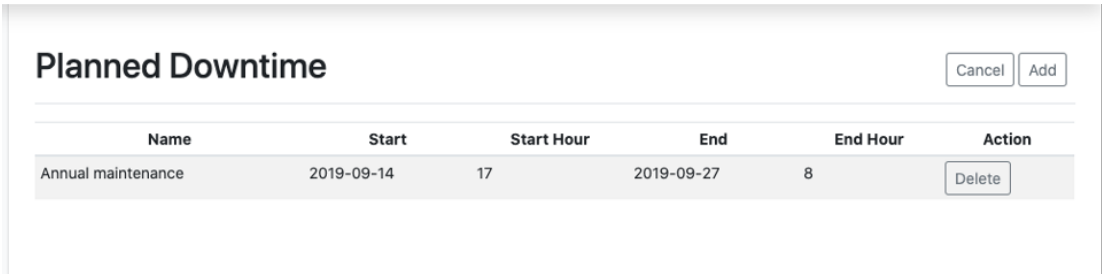
The coupling and balancing variables can be updated in the same way as illustrated in Figure 5.3 for decision variables.



Decision Variables

ID	Name	Margin	Start up cost	Min	Max	Yield	Type	Delta Min	Delta Max	Inc	UOM	Action
0	$P_{3,2}$	R6 956	R1 000 000	14	19.5	0.95	Range	-1	3	0.5	t/h	Edit Downtime
1	$P_{4,2}$	R7 337	R1 000 000	24	28	0.95	Range	-1	3	0.5	t/h	Edit Downtime
7	$P_{2,2}$ A	R3 049	R1 000 000	0	19	0.95	Range	-1	4	0.5	kNm^3	Edit Downtime
8	$P_{2,2}$ B	R3 049	R1 000 000	0	19	0.95	Range	-1	4	0.5	kNm^3	Edit Downtime
9	$P_{2,2}$ C	R3 049	R1 000 000	0	19	0.95	Range	-2	5	0.5	kNm^3	Edit Downtime
20	$T_{2,2}$ In	(-)R1 000	R0	0	[8,15]	1	Fixed	-1	1	0.5	t/h	Edit Downtime
21	$T_{2,2}$ Out	(-)R100	R0	0	50	1	Range	-2	6	0.5	t/h	Edit Downtime

Figure 5.3: Decision variable extract from the DSS



Planned Downtime

Cancel Add

Name	Start	Start Hour	End	End Hour	Action
Annual maintenance	2019-09-14	17	2019-09-27	8	Delete

Figure 5.4: Downtime view for one decision variable

5.2.2 Managing jobs

Creating a job for the model base is illustrated in Figure 5.5. The first option the SME can select from is the problem. Currently, the only option is the Feedstock distribution. The second option that the SME can select is the planning horizon. Two horizons are available for the feedstock distribution problem, namely 14 days with an hourly interval and 90 days with a daily interval. The third option shows the algorithms available in the model base for the selected problem and the selected horizon. The last two options are the starting date and the starting hour for the model. It is possible to add a historical starting date. This is beneficial when trying to find alternative operating philosophies for events that occurred in the past. The start date and time are important for the downtime that has been added. Specific dates are used for downtime and the model requires a relative starting point for the downtime in the system. After submitting the job, it is added to the system.

A list of jobs in the system can be seen in Figure 5.6. The list includes the 20 algorithms that have been developed for the study; 10 for the hourly interval and the other 10 for the daily interval. The list has been edited to include the examples in one view which will now be discussed. The SME can find more information on a job by using the Detail button.

Job IDs 897, 898 and 899 have been added to the model base and are currently in disabled status. These jobs can be queued for execution by pressing the Queue button. The Edit button allows the starting variables to be changed to reflect the current scenario. The starting variables include feedstock entering the C_2 value chain, the current values for each of the decision variables and the current coupling variables. Queuing a job without editing the starting variables will queue the job with previously used starting variables. Once the queue button is pressed, the scenario data is generated for the job.

Job 894, 895 and 896 have been queued for execution and will be executed at the next available opportunity. Any job in the queue can be cancelled by pressing the Cancel button. The job will then return to Disabled status and can be edited and queued again. The configuration parameters for each algorithm are seen in the comment column.

Job 892 and 893 are currently being executed and indicate the status as In Progress. The comment field includes the percentage completed and the JOBID on the HPC. The algorithms check in with the website after each iteration to update the progress. The JOBID is useful for fault-finding when looking for the logs on the HPC.

Job 884 has been found to be infeasible.

Job 885 to 891 have been completed. The Report button will take the SME to the report for the job that will be discussed next.

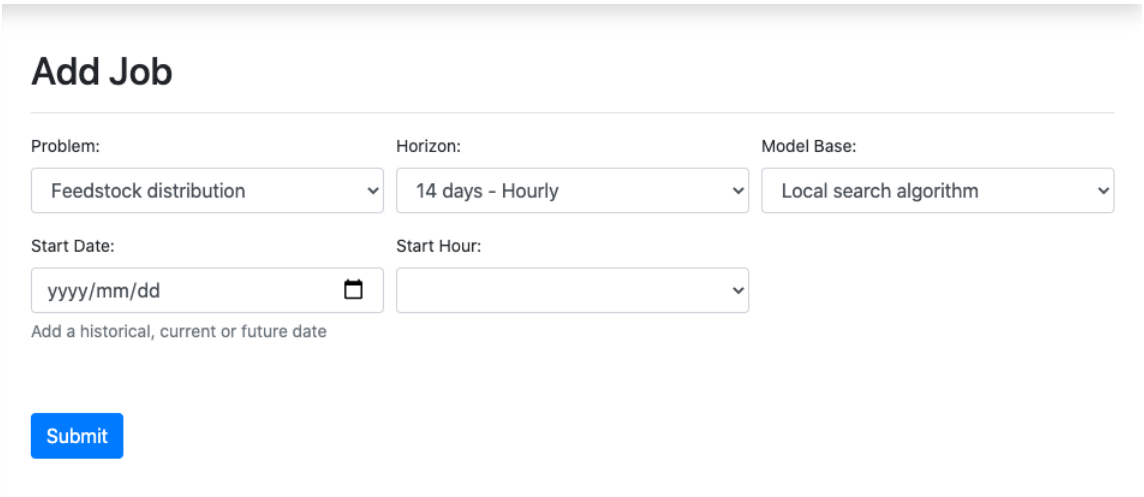


Figure 5.5: Add a job for the model base to execute

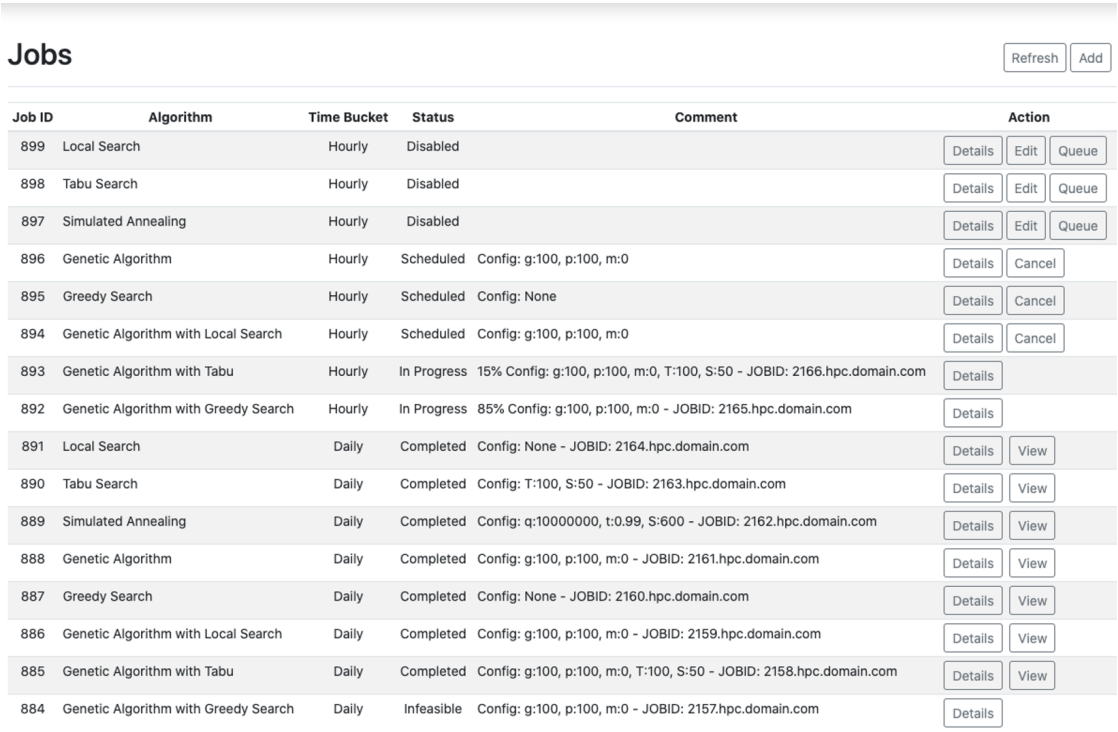


Figure 5.6: List of jobs in the system

5.2.3 Reading results

The SME can read the results from the model base on a report similar to that shown Figure 5.7. For this study, two reports were developed to indicate how feedstock should be distributed every hour for the next 14 days and every day for the next 90 days. The report includes a title, navigation buttons to previous reports in the same category, and a job detail summary and graphs. A Show/Hide Details button on

the report makes additional information available such as the variable configurations, downtime and starting variables. The job detail summary shows the Algorithm name, Profit achieved, Run time and Comment field.

Two important fields are the profit and run time. The profit is the fitness of the solution that the algorithm achieved, while the run time indicates the execution time of the job by the algorithm. These two values will be discussed further in the next chapter.

The Graphs section includes multiple graphs to provide insight to the solution generated by the algorithm. The graphs include consumer production rates, flaring on the C_2 value chain, pipeline pressures, Secunda ethane producers production rates and Sasolburg ethane producers production rates. The consumer production rate graph seen in Figure 5.7 will be discussed. The graph shows an hourly operating rate for each consumer plant. The x-axis indicates the time horizon of 336 hours, and the y-axis indicates the production rate of the consumer plants in t/h. On the graphs it can be seen at hour 200, plant $P_{3,2}$ must produce at 17 t/h and $P_{5,2}$ at 13 t/h.

The architecture allows for different graphs to be developed depending on the data available in the database. In the next section, the database will be discussed.

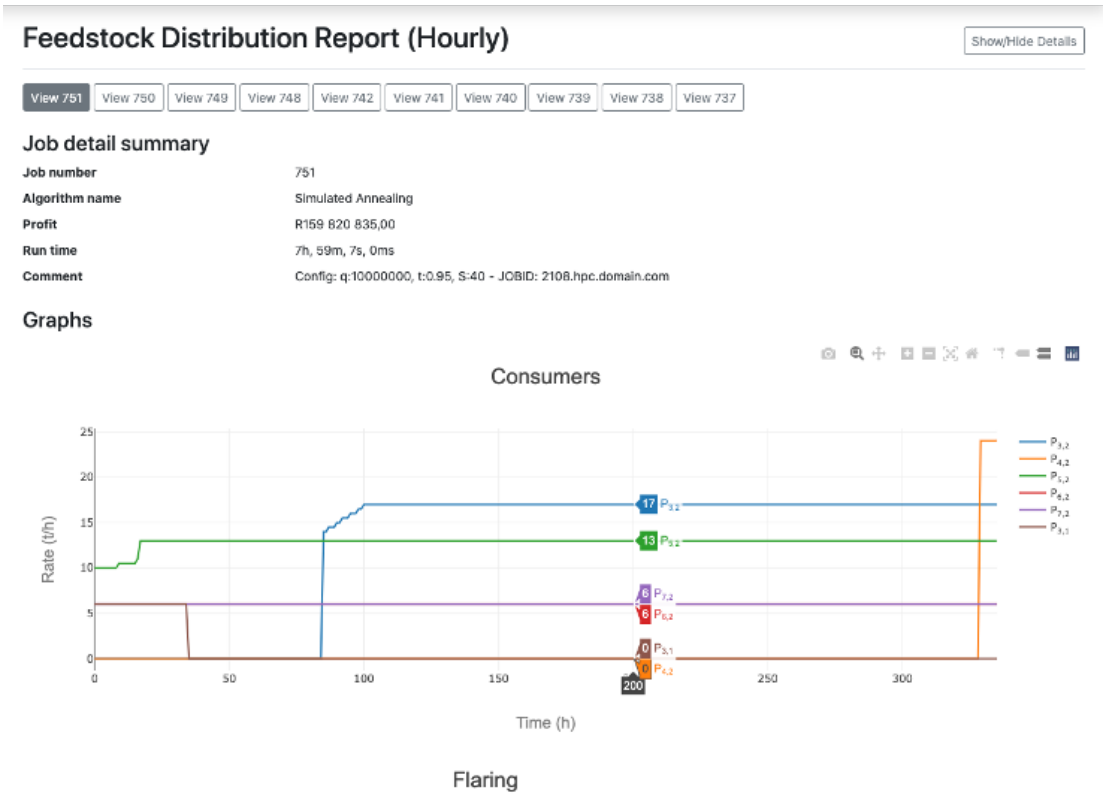


Figure 5.7: A job report in the system

5.3 System database

The database stores the algorithm configurations, variables, scenarios, results and data needed for the user interface. The database acts as a storage interface between the model base and the user interface. The technology stack used for the database is listed in Table 5.2. The database can be hosted on the same server as the website or on a separate server. The architecture can accommodate the use of existing database environments for a database. An extended entity-relationship diagram (EERD) illustrates the database tables and relationships between the tables. The EERD for the database is shown in Figure 5.8 and a description for each data table is listed in Table 5.3.

Table 5.2: Technology stack used for the database

Database	PostgreSQL
Operating system	Ubuntu 18.04 LTS

Table 5.3: Descriptions for the tables in the EERD

system	Status of the system and controllers registered on the system
scenario	Starting variables generated for each job
balancing_variables	List of the balancing variables used in the system
coupling_variables	List of coupling variables used in the system
decision_variables	List of decision variables used in the system
downtime	Downtime associated with each decision variable
jobs	Jobs listed for the model base to execute
algorithms	List of algorithms available in the system with filenames
results	The results for each job

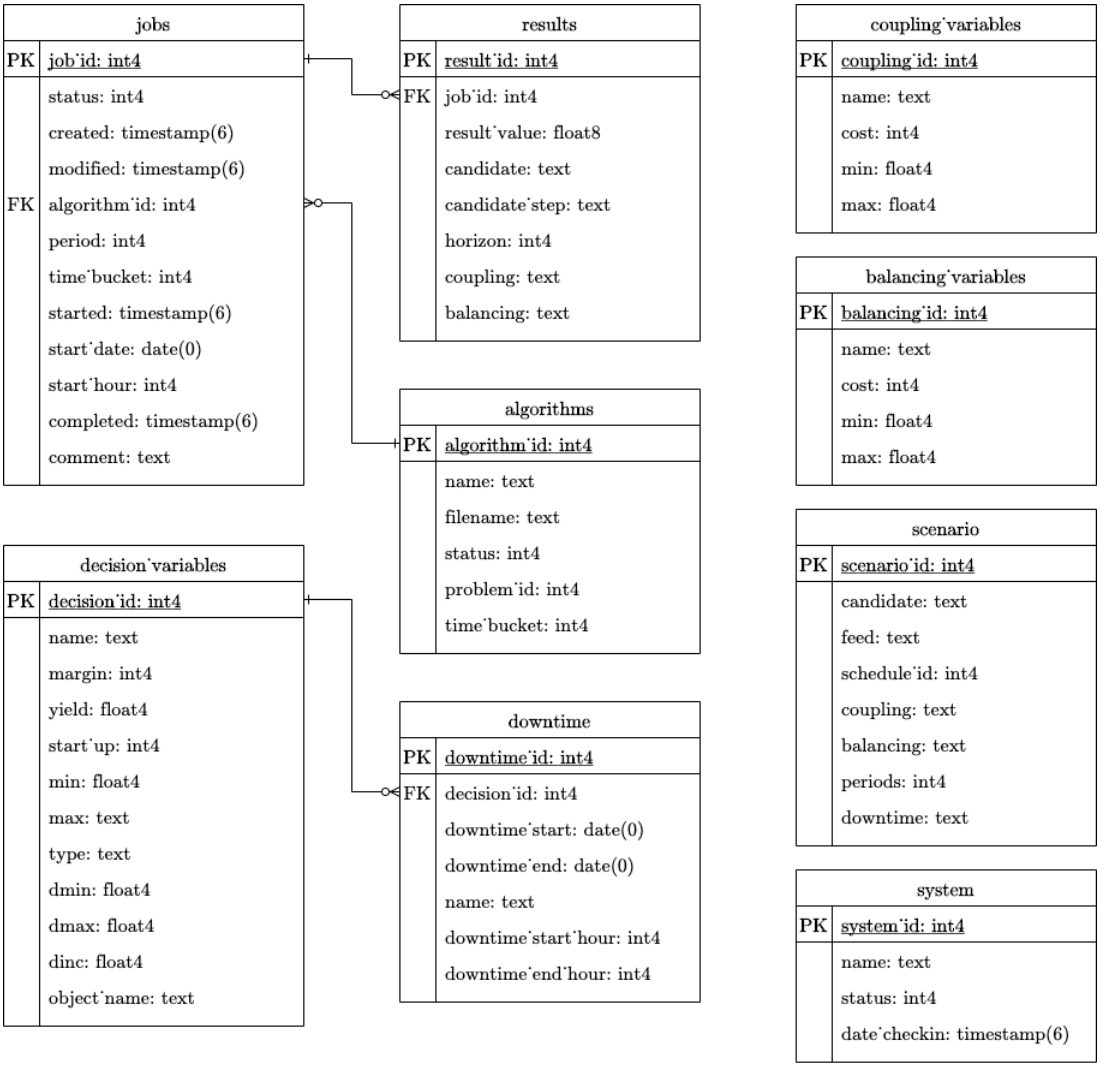


Figure 5.8: Entity relationship diagram for the DSS

5.4 System model base

The model base includes all the algorithms available in the system. The technology stack used for the model base is listed in Table 5.4. The model base is discussed by referring to the following components: input data, models, output data and controller.

5.4.1 Model base input data

The models or algorithms require a set of input data to execute a job in the queue. The following input data is used in the model base:

- Algorithm configuration – Each algorithm requires different parameters such as population size, number of generations, tabu list size, etc;

- Downtime – Downtime of decision variables such as a plant or tank will be from the date selected for the length of the horizon;
- Scenario – Starting values for decision, coupling, balancing variables and the feedstock that is available to distribute over the length of the horizon;
- Configuration files for variables – Files containing the minimum, maximum, margin, cost and yield for the decision, coupling and balancing variables that are used in the algorithms, the constraints and objective function.

5.4.2 Model base models

The model base for this study includes the 20 algorithms listed in Table 4.1. The architecture allows the model base to include other algorithms, optimisers or solvers that can fit into the method of execution. The algorithms with pseudocode were discussed in Chapter 3 and the model construction with implementation in Chapter 4. The algorithms will start with the configuration received as input data from the controller. The algorithms will read all the input data and converts it into the datasets needed. The algorithms will execute, and on completion, pass the output data to the controller.

5.4.3 Model base output data

The output data received from the algorithm is passed to the controller and contain the following data:

- Job identifier – Unique job number in the system;
- Profit and energy consumption (when applicable) – Result from the objective functions;
- Suggested solution – Solution for each interval in the horizon and the step encoded solution for the hourly time interval;
  - The solution length for the 90-day daily time interval is 1 980 variables long;
  - The solution length for the 14-day hourly time interval is 7 392 variables long;
  - The step-encoded solution length for the 14-day hourly time interval is 7 392 variables long;

Table 5.4: Technology stack used for the model base

Machine learning library	TensorFlow and Keras
Programming language	Python 3
Operating system development environment	MacOS
Operating system server	Ubuntu 18.04 LTS
Operating system HPC	Red Hat 4

- Testing identifier – Testing number assigned to job during specific tests execution for this study;
- Coupling variables – Coupling variables for each interval in the horizon;
- Balancing variables – Balancing variables for each interval in the horizon;
- System identifier – Server name or node name.

5.4.4 Model base controller

The controller continuously monitors the database for jobs in the queue using the API. Once a job is added to the queue, the controller fetches the input data and stores it in JavaScript Object Notation (JSON) format files. The model controller starts the algorithm associated with the job. Once the algorithm has completed the execution, the controller submits the output data from the algorithms to the database using the API. Each model base deployment requires a controller. The controller can be limited to a specific algorithm or problem. This allows for one controller or server to be dedicated to an algorithm or problem that should always be ready for execution. A controller that is open for all jobs will execute jobs in a sequential order as processing capacity frees up.

5.5 Chapter summary

A decision support system architecture was presented in this chapter. The different components were then discussed to provide additional detail on the design and implementation. The presented architecture provides flexibility for selecting different technologies for different requirements, scalability for different deployment sizes and can accommodate additional algorithms, optimisers and solvers. In the next chapter, the verification and evaluation of the decision support system will be discussed.

Chapter 6

Verification and evaluation

Chapter 5 presented a decision support system (DSS) design and discussed the implementation of the DSS. This chapter presents the verification and evaluation of the DSS. The chapter starts by specifying the tests that are needed to verify the code used in the model base. The performance of the algorithms is then compared with each other, followed by an analysis of the parameters used with the best performing algorithms. Finally, an evaluation of the DSS is presented leading up to an experiment based on the recommendation of the subject matter experts.

6.1 Code verification

Testing code identifies failures and errors within it. This is done by doing unit and integration testing. Unit testing checks that a specific response is received from a piece of code to a set of inputs. Most modern programming languages have built-in unit-testing or support a unit testing library.

Integration testing checks that different components work with each other and can include integration with external systems.

For this study, unit testing was done on the functions and integration testing on the application programming interface (API). Although all known errors have been addressed, it is possible that latent errors may still exist. A test runner executes a test group or test case and provides a unit test report. Test fixtures contain test data that is needed in a test case. Figure 6.1 illustrates the unit test report with the major functions that are used in the model base and Figure 6.2 shows the project structure with test scripts for the model base. The report indicates that all tests have passed. It is possible for a test to fail or to give an error. A failure indicates that the expected result was not received from the test and an error indicates that an error in the code occurred before doing the test. The details of

two test groups can be seen on the unit test report. The project structure is useful to understand the naming convention used in the unit test report. The test group project.tests.unit.test_functions.generate_random_candidate is described in Table 6.1 and project.tests.integration.test_api_get.api_get_job in Table 6.2.

Table 6.1: Generate random candidate unit test

Test group	project.tests.unit.test_functions.generate_random_candidate
Inputs passed	A multi-dimensional list containing all the options for each decision variable for the complete horizon.
Function	A random item is selected between the minimum and maximum for each interval in the multidimensional list and added to a new list named candidate. The candidate is the output of the function.
Test case 1	Is the candidate length equal to 1 980 for daily, the interval 7 392 for the hourly interval and 1 826 for the combined interval?
Test case 2	Are all the items in the candidate found in the multidimensional list for the correct interval?

Table 6.2: Fetch job integration test

Test group	project.tests.integration.test_api_get.api_get_job
Inputs passed	None.
Function	Request a job from the API that has been queued in the database.
Test case 1	Has a job identifier been returned? The expected response is an integer for job identifier or 0 for no jobs.

Unit and integration testing checks for errors in code but does not report the performance of the algorithms. In the next section, a discussion on testing the performance of the algorithms is presented.

Unit Test Report

Status: Pass 24 Failure 0 Error 0

Show Summary Failed All

Test Group/Test case	Description	Count	Pass	Fail	Error	View
project.tests.unit.test_functions.generate_random_candidate	Generate a random candidate	2	2	0	0	Detail
test_candidate_length	Candidate length			pass		
test_valid_candidate	Each value of candidate in multi-dimensional list			pass		
project.tests.unit.test_functions.generate_recursive_candidate	Continuously generating a candidate until stopping criteria is met	1	1	0	0	Detail
project.tests.unit.test_functions.calculate_profit	Calculate the profit	1	1	0	0	Detail
project.tests.unit.test_functions.check_fitness_with_constraints	Check fitness from a single period	3	3	0	0	Detail
project.tests.unit.test_functions.check_fitness_with_constraints_multi_period	Check fitness for all periods using the single period function	1	1	0	0	Detail
project.tests.unit.test_functions.decode_candidate	Decode the candidate	2	2	0	0	Detail
project.tests.unit.test_functions.downtime_list_0_1	Convert downtime dates into binary variables for a single decision variable	3	3	0	0	Detail
project.tests.unit.test_functions.generate_scope_from_config_downtime	Generate multi-dimensional list using the downtime_list_0_1 and merge multiple downtimes event into one timeline	4	4	0	0	Detail
project.tests.unit.test_functions.generate_config_files	Store JSON array to JSON file for decision, coupling and balancing variables	6	6	0	0	Detail
project.tests.unit.test_functions.generate_scenario_files	Store JSON array to JSON file for scenario and shutdown	4	4	0	0	Detail
project.tests.unit.test_functions.generate_scope_from_config	Generate a multi-dimensional list containing all the options for each decision variable for the complete horizon	2	2	0	0	Detail
project.tests.unit.test_functions.generate_step_scope_from_config	Generate a multi-dimensional list containing all the step encoded options for each decision variable for the complete horizon	2	2	0	0	Detail
project.tests.unit.test_functions.convert_json_array_to_list	Convert JSON array to python list	1	1	0	0	Detail
project.tests.unit.test_functions.generate_neighbourhood	Generate all neighbours for a candidate	3	3	0	0	Detail
project.tests.unit.test_functions.neighbourhood_result	Check fitness for all neighbourhood over multiple processors	2	2	0	0	Detail
project.tests.integration.test_api_get.api_get_job	Get a job in the queue	1	1	0	0	Detail
test_get_job	Get a job from the queue			pass		
project.tests.integration.test_api_get.api_get_scenario	Get the job scenario	1	1	0	0	Detail
project.tests.integration.test_api_get.api_get_detail	Get the job detail	1	1	0	0	Detail
project.tests.integration.test_api_post.api_submit	Submit data to the API	1	1	0	0	Detail
project.tests.integration.test_api_post.api_update	Update the job progress in the database	1	1	0	0	Detail
project.tests.integration.test_api_post.api_add	Add a job to the database	1	1	0	0	Detail
project.tests.integration.test_api_post.api_status	Update job status in the database	1	1	0	0	Detail
project.tests.integration.test_api_post.api_complete	Update job status to complete in the database	1	1	0	0	Detail
project.tests.integration.test_api_post.api_comment	Add a comment to the job	1	1	0	0	Detail
Total		**24**	**24**	**0**	**0**	

Figure 6.1: Unit test report

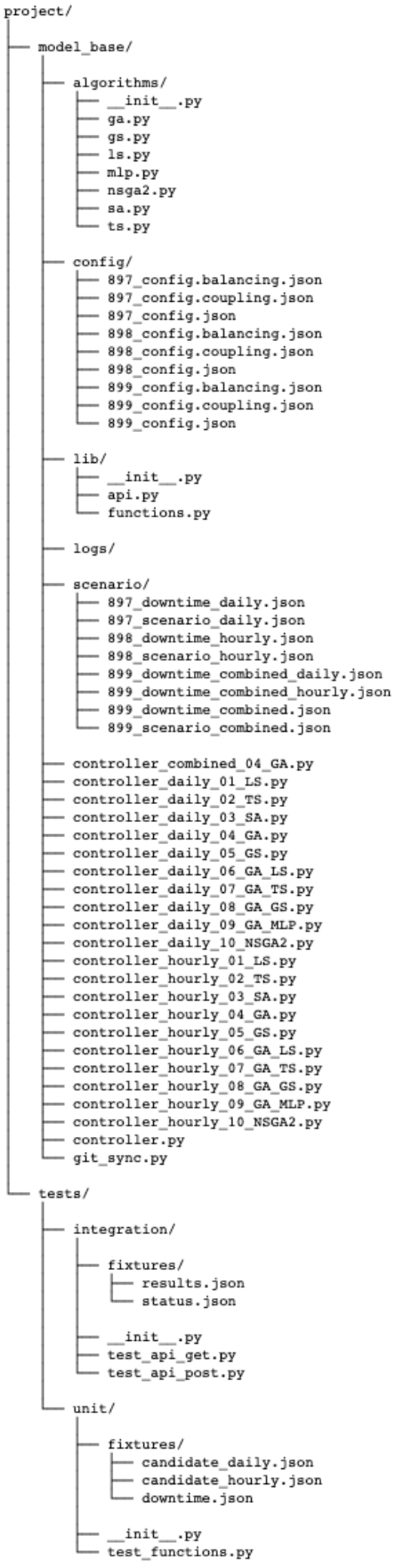


Figure 6.2: Project file structure

6.2 Algorithm testing

In Chapter 1, the following five questions were listed on the operating philosophy of the C_2 value chain:

1. What is the most profitable schedule for the next 90 days?
2. What is the most profitable schedule for the next 14 days?
3. How should the feedstock be distributed in the next 90 days when a unit has unplanned downtime?
4. How should the feedstock be distributed in the next 14 days when a unit has unplanned downtime?
5. What is the trade-off between the profitability of the schedule and energy consumption?

To test the algorithms, they are executed to find answers to these questions. The tests are grouped into the three categories; namely, the most profitable schedule, the unplanned downtime schedule and the trade-off between profitability and energy consumption. In the following two sections, these categories will be discussed, followed by a conclusion on the testing of the algorithms.

6.2.1 Testing for the most profitable schedule

To test the algorithms in this category, two fixed scenarios were used that will allow for the comparison of algorithms.

The differences between the two scenarios were the time interval, horizon length and different starting values for the decision variables. The first scenario included a 90-day horizon with a daily interval and the second included a 14-day horizon with an hourly interval. Both scenarios used the same starting date, the same feedstock that is available for distribution over the length of the horizon, the same downtime and the same starting values for the coupling and balancing variables.

Two dimensions can be used to compare the performance of the algorithms. The first dimension is the algorithm that achieved the highest profit and the second is the run time of the algorithm.

6.2.1.1 90-day horizon with daily interval results

Table 6.3 lists the top 30 results for the 90-day horizon with daily intervals. The table includes the results of algorithms that were executed with different parameters. The performance using different parameters will be discussed later in the section.

For the first performance dimension on profit, the hybrid algorithms outperformed the other algorithms consistently on profit but had the longest run times. The performance of the genetic algorithm (GA) varied widely depending on the parameters used. The remaining algorithms all performed fairly. Interestingly, the results numbered 19 and 20 from the greedy search (GS) and local search (LS) generated the same solution. Further investigation into the execution path revealed that the first improvement in the first neighbourhood was the better solution which directed the search for both algorithms into the same neighbourhood.

For the second dimension on run time, the GA consistently completed the execution in the shortest time. Nevertheless, the run time is not critical for the 90-day horizon with a daily interval.

Table 6.3: Top 30 daily interval solutions

	Profit	Run time	Algorithm name
1	R1 415 541 940	12h, 51m, 11s	Genetic Algorithm with Local Search
2	R1 410 779 580	4h, 45m, 31s	Genetic Algorithm with Tabu
3	R1 394 279 960	10h, 23m, 33s	Genetic Algorithm with Greedy Search
4	R1 389 168 880	15h, 34m, 8s	Genetic Algorithm with Tabu
5	R1 384 988 520	4h, 14m, 20s	Genetic Algorithm
6	R1 378 777 340	13h, 46m, 13s	Genetic Algorithm with Local Search
7	R1 348 584 420	15h, 20m, 27s	Genetic Algorithm
8	R1 335 219 340	0h, 36m, 54s	Genetic Algorithm
9	R1 332 734 000	0h, 13m, 43s	Genetic Algorithm
10	R1 329 543 823	0h, 10m, 26s	Genetic Algorithm
11	R1 303 820 565	1h, 47m, 35s	Genetic Algorithm
12	R1 298 072 880	1h, 57m, 13s	Genetic Algorithm
13	R1 278 366 305	1h, 14m, 15s	Genetic Algorithm
14	R1 232 988 050	0h, 25m, 20s	Genetic Algorithm
15	R1 229 056 565	0h, 31m, 53s	Genetic Algorithm
16	R1 224 520 853	0h, 47m, 57s	Simulated Annealing
17	R1 213 248 360	3h, 31m, 20s	Genetic Algorithm
18	R1 211 576 590	0h, 22m, 45s	Simulated Annealing
19	R1 204 511 760	0h, 31m, 41s	Greedy Search
20	R1 204 511 760	3h, 29m, 15s	Local Search
21	R1 187 377 800	0h, 10m, 30s	Genetic Algorithm
22	R1 160 720 853	0h, 22m, 1s	Genetic Algorithm
23	R1 134 057 525	0h, 24m, 5s	Genetic Algorithm
24	R1 118 841 707	0h, 8m, 50s	Genetic Algorithm
25	R1 116 317 500	2h, 26m, 59s	Tabu Search
26	R1 115 859 343	0h, 17m, 19s	Genetic Algorithm
27	R1 084 935 935	0h, 14m, 56s	Genetic Algorithm
28	R1 078 870 075	0h, 7m, 5s	Genetic Algorithm
29	R1 034 769 583	0h, 16m, 40s	Genetic Algorithm
30	R1 020 576 590	0h, 36m, 24s	Genetic Algorithm

A selection of decision variable values for the most profitable schedule for the daily interval scenario is illustrated by Figures 6.3, 6.4, 6.5 and 6.6. Figure 6.3 illustrates the production rate for the consumer units, three of which are kept at a constant rate for 90 days after adjusting from the starting variables on day zero. $P_{5,2}$ is running on day zero and is shut down on day one. The unit is restarted on day 79 with a

fluctuating production rate until day 87. The low ethylene pipeline pressure shown in Figure 6.4 prevents additional units from starting during this interval and $P_{5,2}$ is started for a short period of time when sufficient line pressure is available. Figures 6.5 and 6.6 show the ethylene recovery units' ($P_{1,1}$ and $P_{1,2}$) rates. The recovered ethylene is passed to the ethylene pipeline and the ethane to the ethane cracking units, each of which consists of a separation unit and furnaces.

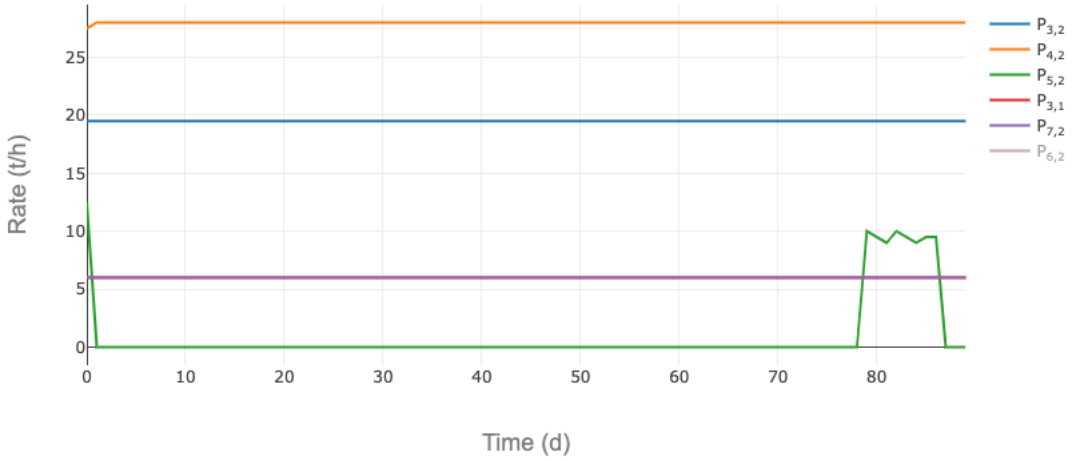


Figure 6.3: Consumer units schedule for the most profitable daily interval scenario

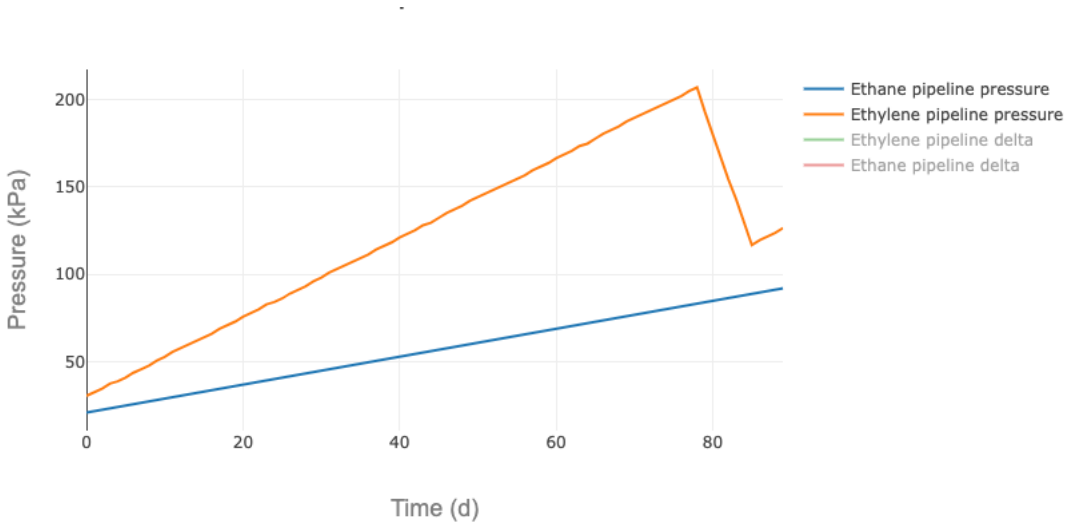


Figure 6.4: Pipeline pressure for the most profitable daily interval scenario

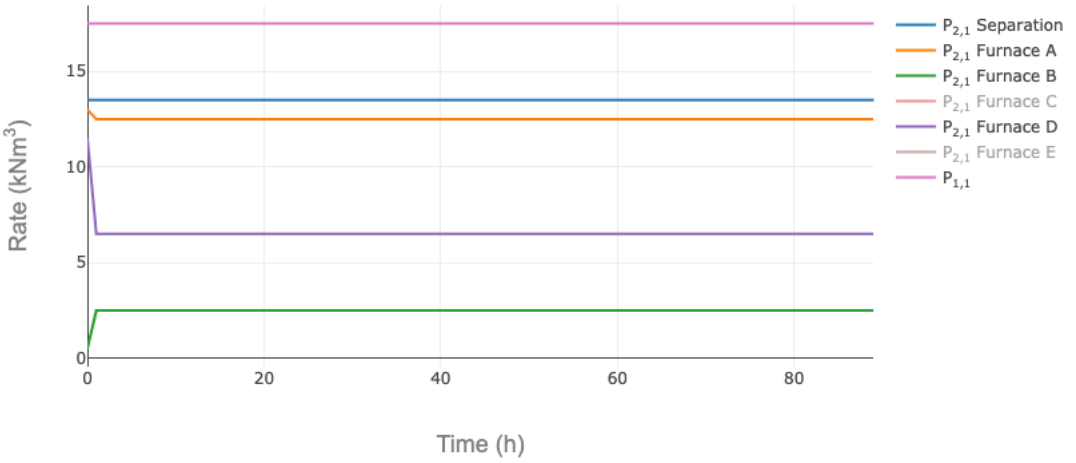


Figure 6.5: Secunda ethane units schedule for the most profitable daily interval scenario

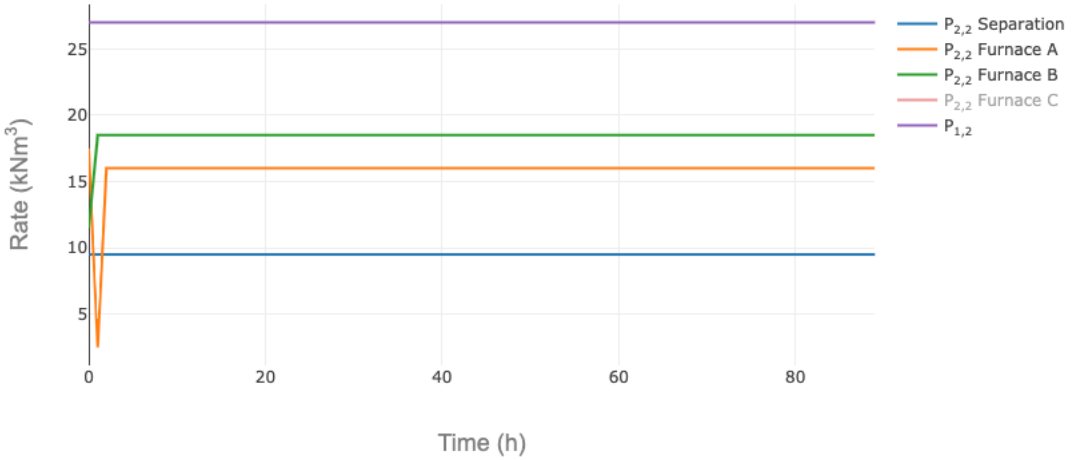


Figure 6.6: Sasolburg ethane units schedule for the most profitable daily interval scenario

6.2.1.2 14-day horizon with hourly interval results

Table 6.4 lists the top 30 results for the 14-day horizon with hourly intervals. The table includes the results of algorithms that were executed with different parameters. The performance using different parameters will be discussed later in the section.

For the first performance dimension on profit, the highest profit was achieved by the simulated annealing (SA) algorithm. The hybrid algorithms provided good results with reasonable run times. As highlighted in the problem description, an upset in the value chain requires time-sensitive decision-making and any decision support system

should propose a solution within two hours and preferably in less than an hour. Table 6.5 lists the 19 results from the top 30 that were completed within two hours. The hybrid algorithms and the GA managed to complete within the allocated time. Table 6.6 lists the algorithms that could not finish within two hours.

All the SA algorithm results exceeded the two hours allowed for execution. The LS and GS algorithms exceeded the allocated two hours but ranked in the top results when paired as a hybrid with the GA. The tabu search (TS) exceeded the two hours but as a hybrid with the GA managed to complete some of the jobs with specific parameters. One GA with a GS algorithm did not complete within two hours due to the parameter values used with the GA.

The 14-day horizon with hourly intervals should provide decision support during the time-sensitive decision interval.

Table 6.4: Top 30 hourly interval solutions

	Profit	Run time	Algorithm name
1	R179 620 070	75h, 35m, 51s	Simulated Annealing
2	R177 274 788	53h, 46m, 28s	Genetic Algorithm with Greedy Search
3	R176 842 443	1h, 21m, 59s	Genetic Algorithm with Local Search
4	R176 600 338	2h, 17m, 21s	Genetic Algorithm with Tabu
5	R176 492 128	2h, 13m, 52s	Genetic Algorithm with Tabu
6	R175 542 540	2h, 16m, 2s	Genetic Algorithm with Tabu
7	R173 436 175	0h, 49m, 25s	Genetic Algorithm with Greedy Search
8	R172 622 533	0h, 36m, 44s	Genetic Algorithm with Greedy Search
9	R168 391 905	2h, 15m, 47s	Genetic Algorithm with Tabu
10	R167 192 623	0h, 5m, 29s	Genetic Algorithm
11	R166 871 828	0h, 50m, 35s	Genetic Algorithm with Greedy Search
12	R165 333 305	10h, 3m, 24s	Tabu Search
13	R164 118 015	0h, 30m, 37s	Genetic Algorithm with Tabu
14	R163 598 853	1h, 11m, 35s	Genetic Algorithm
15	R162 879 413	0h, 31m, 56s	Genetic Algorithm with Tabu
16	R162 076 055	0h, 2m, 11s	Genetic Algorithm
17	R161 754 330	1h, 9m, 58s	Genetic Algorithm
18	R161 496 950	1h, 16m, 50s	Genetic Algorithm
19	R161 492 853	0h, 31m, 24s	Genetic Algorithm with Tabu
20	R159 820 835	7h, 59m, 7s	Simulated Annealing
21	R159 510 625	1h, 10m, 3s	Genetic Algorithm
22	R159 429 568	1h, 12m, 26s	Genetic Algorithm
23	R159 091 690	1h, 15m, 8s	Genetic Algorithm
24	R158 356 993	1h, 14m, 42s	Genetic Algorithm
25	R156 426 428	0h, 5m, 28s	Genetic Algorithm
26	R151 928 318	1h, 14m, 54s	Genetic Algorithm
27	R147 795 165	10h, 3m, 20s	Local Search
28	R144 261 008	3h, 58m, 4s	Greedy Search
29	R142 854 915	0h, 30m, 12s	Genetic Algorithm with Tabu
30	R142 748 318	2h, 56m, 44s	Simulated Annealing

Table 6.5: The hourly interval solutions obtained within two hours of execution

	Profit	Run time	Algorithm name
1	R176 842 443	1h, 21m, 59s	Genetic Algorithm with Local Search
2	R173 436 175	0h, 49m, 25s	Genetic Algorithm with Greedy Search
3	R172 622 533	0h, 36m, 44s	Genetic Algorithm with Greedy Search
4	R167 192 623	0h, 5m, 29s	Genetic Algorithm
5	R166 871 828	0h, 50m, 35s	Genetic Algorithm with Greedy Search
6	R164 118 015	0h, 30m, 37s	Genetic Algorithm with Tabu
7	R163 598 853	1h, 11m, 35s	Genetic Algorithm
8	R162 879 413	0h, 31m, 56s	Genetic Algorithm with Tabu
9	R162 076 055	0h, 2m, 11s	Genetic Algorithm
10	R161 754 330	1h, 9m, 58s	Genetic Algorithm
11	R161 496 950	1h, 16m, 50s	Genetic Algorithm
12	R161 492 853	0h, 31m, 24s	Genetic Algorithm with Tabu
13	R159 510 625	1h, 10m, 3s	Genetic Algorithm
14	R159 429 568	1h, 12m, 26s	Genetic Algorithm
15	R159 091 690	1h, 15m, 8s	Genetic Algorithm
16	R158 356 993	1h, 14m, 42s	Genetic Algorithm
17	R156 426 428	0h, 5m, 28s	Genetic Algorithm
18	R151 928 318	1h, 14m, 54s	Genetic Algorithm
19	R142 854 915	0h, 30m, 12s	Genetic Algorithm with Tabu

Table 6.6: Hourly interval solutions exceeding two hours of execution

	Profit	Run time	Algorithm name
1	R179 620 070	75h, 35m, 51s	Simulated Annealing
2	R177 274 788	53h, 46m, 28s	Genetic Algorithm with Greedy Search
3	R176 600 338	2h, 17m, 21s	Genetic Algorithm with Tabu
4	R176 492 128	2h, 13m, 52s	Genetic Algorithm with Tabu
5	R175 542 540	2h, 16m, 2s	Genetic Algorithm with Tabu
6	R168 391 905	2h, 15m, 47s	Genetic Algorithm with Tabu
7	R165 333 305	10h, 3m, 24s	Tabu Search
8	R159 820 835	7h, 59m, 7s	Simulated Annealing
9	R147 795 165	10h, 3m, 20s	Local Search
10	R144 261 008	3h, 58m, 4s	Greedy Search
11	R142 748 318	2h, 56m, 44s	Simulated Annealing

A selection of decision variables for the most profitable schedule for the hourly

interval scenario is illustrated by Figures 6.7, 6.8, 6.9 and 6.10. Figure 6.7 illustrates the production rate for the consumer units. Three of the units are kept at a constant rate most of the time for 14 days. It can be seen that on hour 34 $P_{3,2}$ is increased to the maximum operating rate. Figure 6.8 shows that with the current ethylene consumers, the ethylene pipeline pressure continues to increase minimally. Figure 6.9 and 6.10 show the rates for the ethylene recovery units, ethane separation units and furnaces.

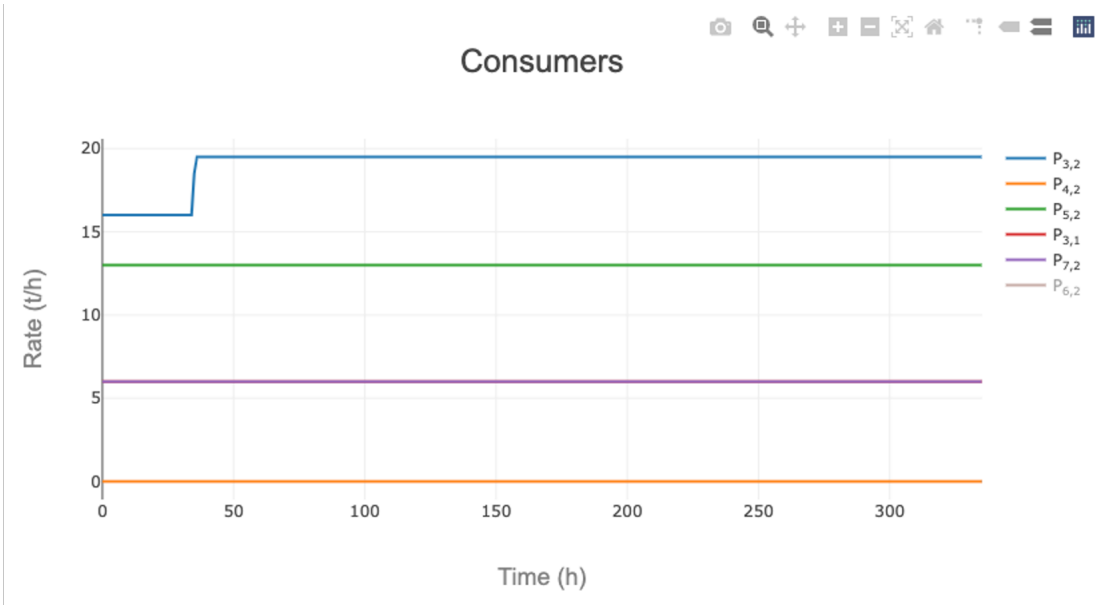


Figure 6.7: Consumer units schedule for the most profitable hourly interval scenario

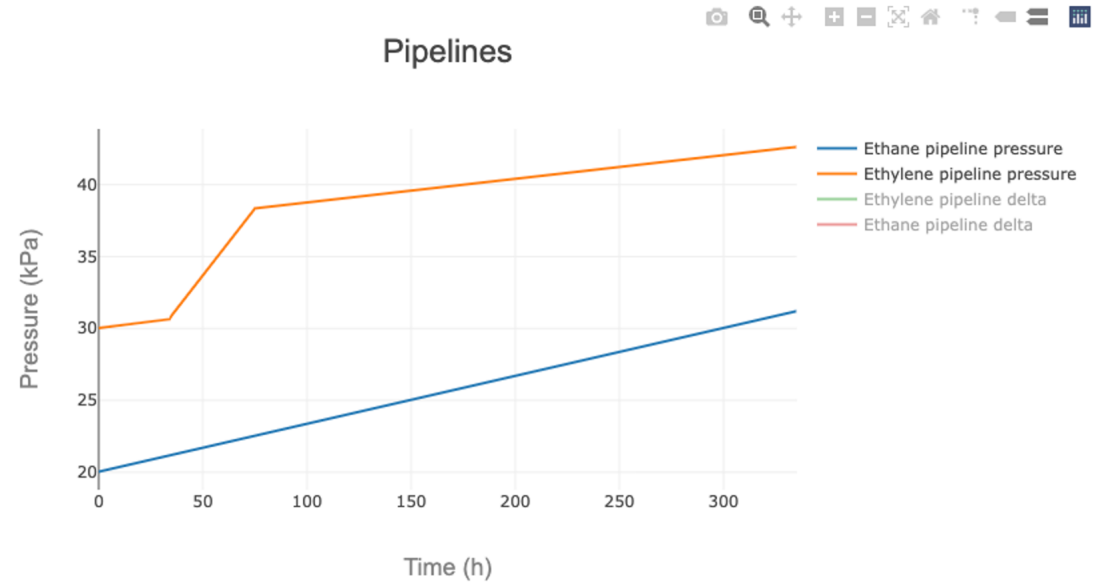


Figure 6.8: Pipeline pressure for the most profitable hourly interval scenario

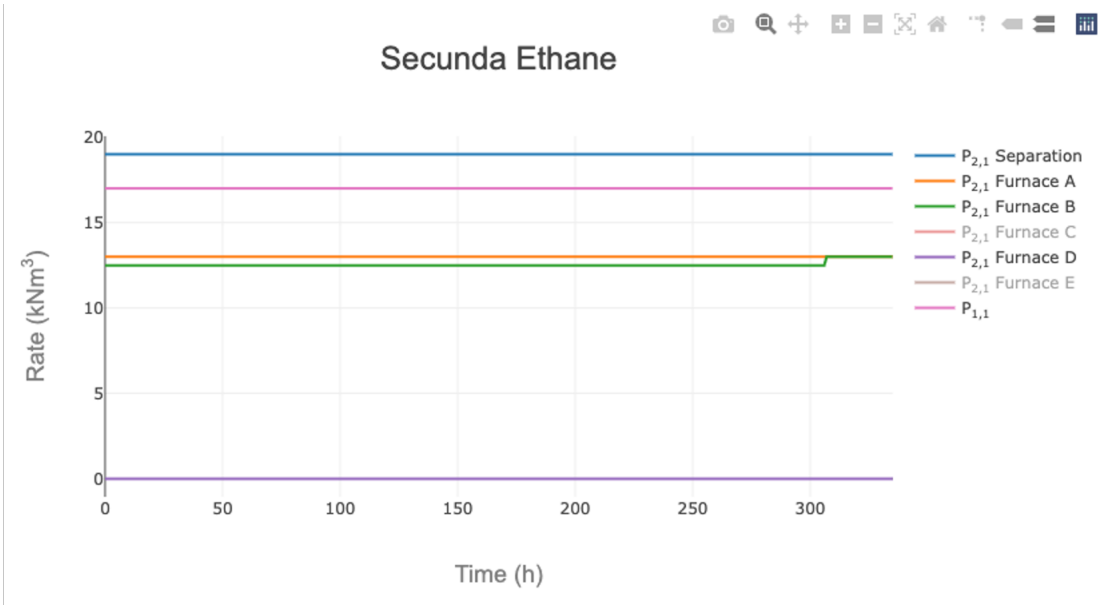


Figure 6.9: Secunda ethane units schedule for the most profitable hourly interval scenario

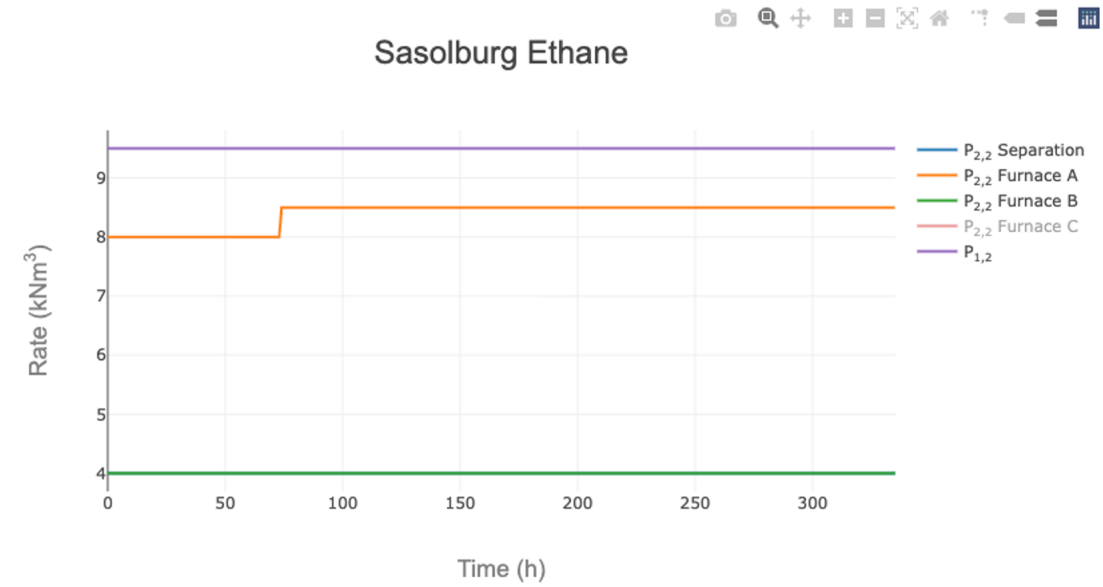


Figure 6.10: Sasolburg ethane units schedule for the most profitable hourly interval scenario

6.2.1.3 Hybrid GA with a multilayer perceptron (MLP) neural network implementation for the 14-day horizon with hourly intervals

Due to the time-sensitive decision-making required for hourly interval, the literature indicated that the neural network's run time is relatively low as most of the time

is spent during the training phase. Using a GA to train an MLP concurs with the time-sensitive decision-making requirement.

The hybridisation was approached with different views and configurations. Using one MLP neural network for all the decision variables for the complete time interval did not predict feasible solutions. This could be due to the sequential nature of the problem and a sliding window method could be applied to sequential supervised learning (Dietterich, 2002). This is similar to the moving-window time-based decomposition that can cause infeasible or suboptimal results due to the lack of forward-looking over all the intervals (Harjunkoski *et al.*, 2014).

A few variations were tested using this method:

1. One MLP neural network per decision variable for each time interval;
2. One MLP neural network for all the decision variables per time interval;
3. One MLP neural network per problem for each time interval.

The last-mentioned variation was the only method that predicted solutions, providing impractical suboptimal results in an unconstrained scenario within a short run time. Two problems remained with this method. First, the model does not consider the historical states and would start or stop plants at every interval. This behaviour is penalised during execution in the other algorithms in this study to prevent impractical schedules. The second problem is that when the environment becomes constrained, the model predicts infeasible solutions that violate constraints.

To possibly consider the historical states, lag features were introduced to include an additional time interval. This did not resolve the problem and a custom loss function was attempted to introduce a penalty similar to that in the other algorithms.

To address the second problem, downstream decision variables were included as features when predicting the upstream variables and vice versa, to reduce the constraint violations when the environment is constrained. This proved to be inconclusive.

The results are summarised in Table 6.7; albeit these are impractical and do not match the results achieved by the other algorithms. The short run times highlight the possible advantage of this type of hybridisation.

Despite the shortcoming of not producing adequate results, the use of MLP, as a possible method, should not be ruled out in its entirety. The problem in this study is specific and has not been addressed before. The immense field of machine learning is still currently being actively researched and a different formulation or configuration could resolve the problems faced in this study. Other classes of neural networks such

as long short-term memory (LSTM), graph neural network (GNN) or a different field of machine learning, reinforcement learning (RL) could also be explored.

Table 6.7: Hybrid GA with an MLP neural network results

	Profit	Run time
1	R11 552 380	27s
2	R7 013 945	27s
3	R2 062 925	27s

6.2.1.4 14-day horizon with combined interval results

Figure 2.4 illustrated the discrete-time single non-uniform grid proposed to reduce the problem size by combining a 72-hour grid and an 11-day grid. The GA was used and a combined solution evaluation illustrated in Figure 4.8 was used. 61 tests were conducted using different parameters that mainly resulted in infeasible solutions. Table 6.8 lists the scheduling jobs that were completed.

Figure 6.11 shows the consumer units combined time intervals schedule and when comparing the combined interval results with the results from the 14-day horizon with hourly intervals, it is clear that the results are inferior.

The researcher surmises that there is an incompatibility when combining the two grids due to the first grid using this specific encoding scheme and the other grid having no encoding scheme. The encoding scheme naturally repairs the violations in the solution over time and continues to consider infeasible solutions during the search process. Developing a similar encoding scheme for the daily interval could address the problem. This would require redeveloping most of the functions used in the 90-day horizon with a daily interval to work with the encoding scheme.

Table 6.8: Combined intervals results

	Profit	Run time	Algorithm name
1	R84 613 535	0h, 8m, 14s	Genetic Algorithm
2	R60 038 977	0h, 5m, 45s	Genetic Algorithm
3	R80 697 861	0h, 5m, 28s	Genetic Algorithm
4	R69 306 842	0h, 5m, 7s	Genetic Algorithm
5	R84 837 160	0h, 10m, 18s	Genetic Algorithm

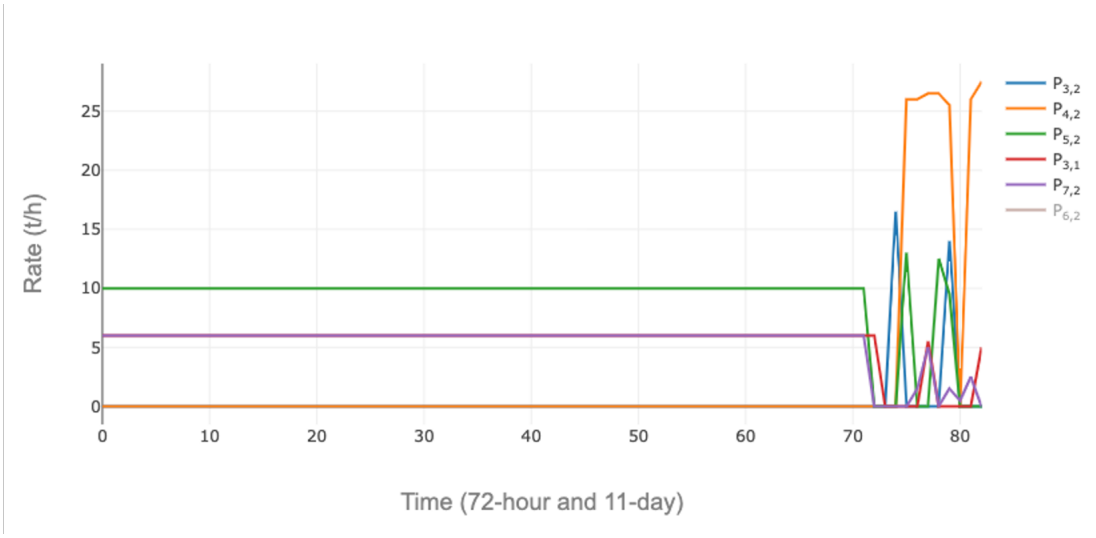


Figure 6.11: Consumer units schedule for the most profitable combined intervals scenario

6.2.2 Testing for the unplanned downtime schedule

To test the algorithms in this category, a consumer unit downtime event was manually added to the system and tested with the same scenario as in the first category to allow the algorithm to recommend how feedstock should be distributed.

6.2.2.1 Distribution of feedstock with unplanned downtime on the daily interval

In this scenario, $P_{4,2}$ had a downtime event for 16 days that prevented the unit from starting. Figure 6.12 indicates the different rates of the consumer units and Figure 6.13 the flaring. The algorithm did not find a more profitable schedule that did not require flaring. Ethane flaring is preferred over ethylene flaring due to the lower cost of ethane. Once $P_{4,2}$ started, the remainder of the results settled into a stable operating range.

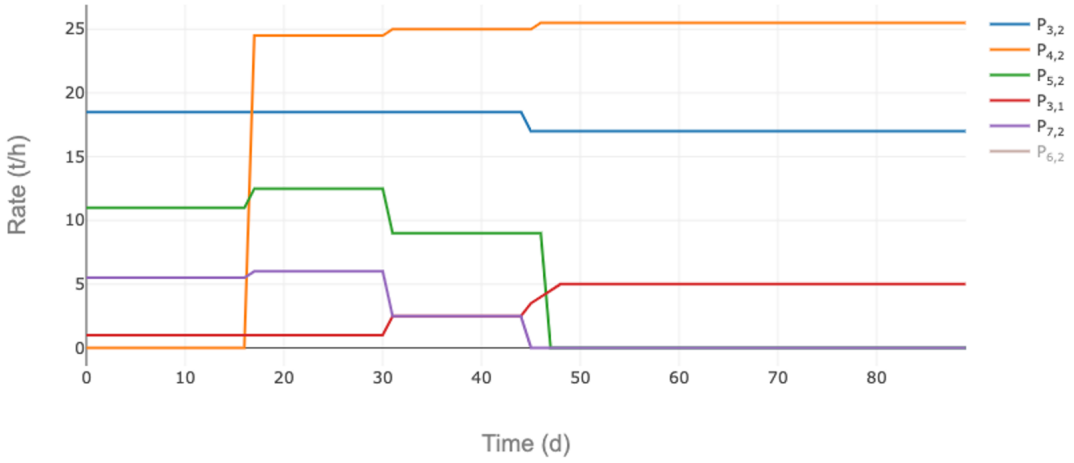


Figure 6.12: Feedstock distribution with consumer unit downtime event for the daily interval

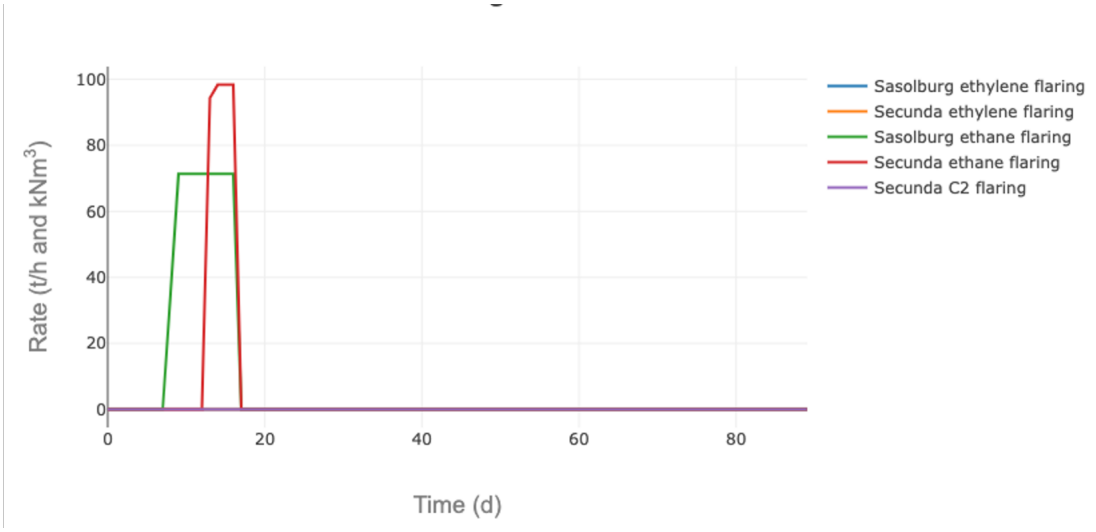


Figure 6.13: Flaring with consumer unit downtime event for the daily interval

6.2.2.2 Distribution of feedstock with unplanned downtime on the hourly interval

In this scenario, $P_{3,2}$ had a downtime event for 85 hours manually added to the system that prevented the plant from starting. Figure 6.14 indicates the different rates of the consumer units and Figure 6.15 indicates the ethylene pipeline pressure over the 336 hours. The ethylene line pressure increases at a fast rate until $P_{3,2}$ starts and is then stable until $P_{4,2}$ starts. This triggers a decrease in the ethylene pipeline until the end of the interval. A subject matter expert (SME) noted that it is not uncommon for models to end the horizon in an unsustainable way.

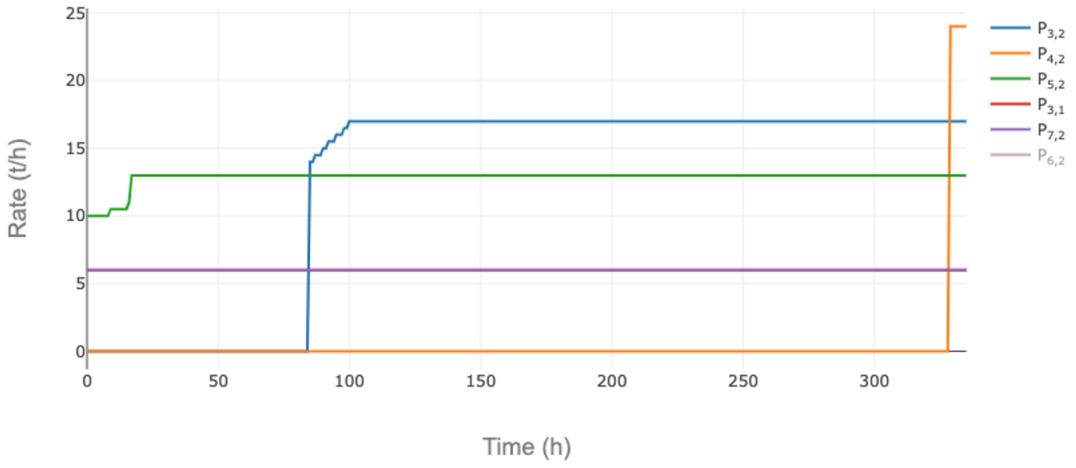


Figure 6.14: Feedstock distribution with consumer unit downtime event for the hourly interval

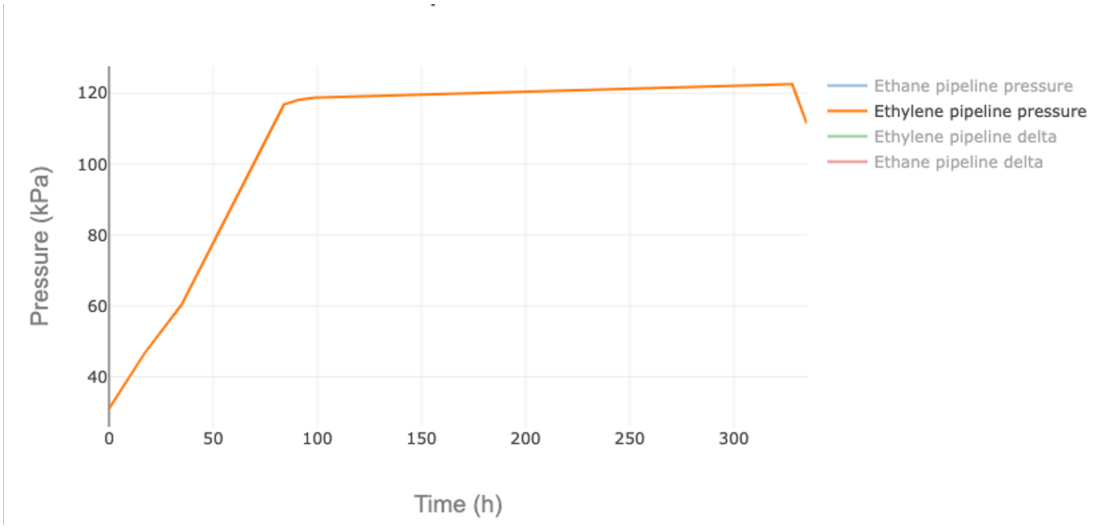
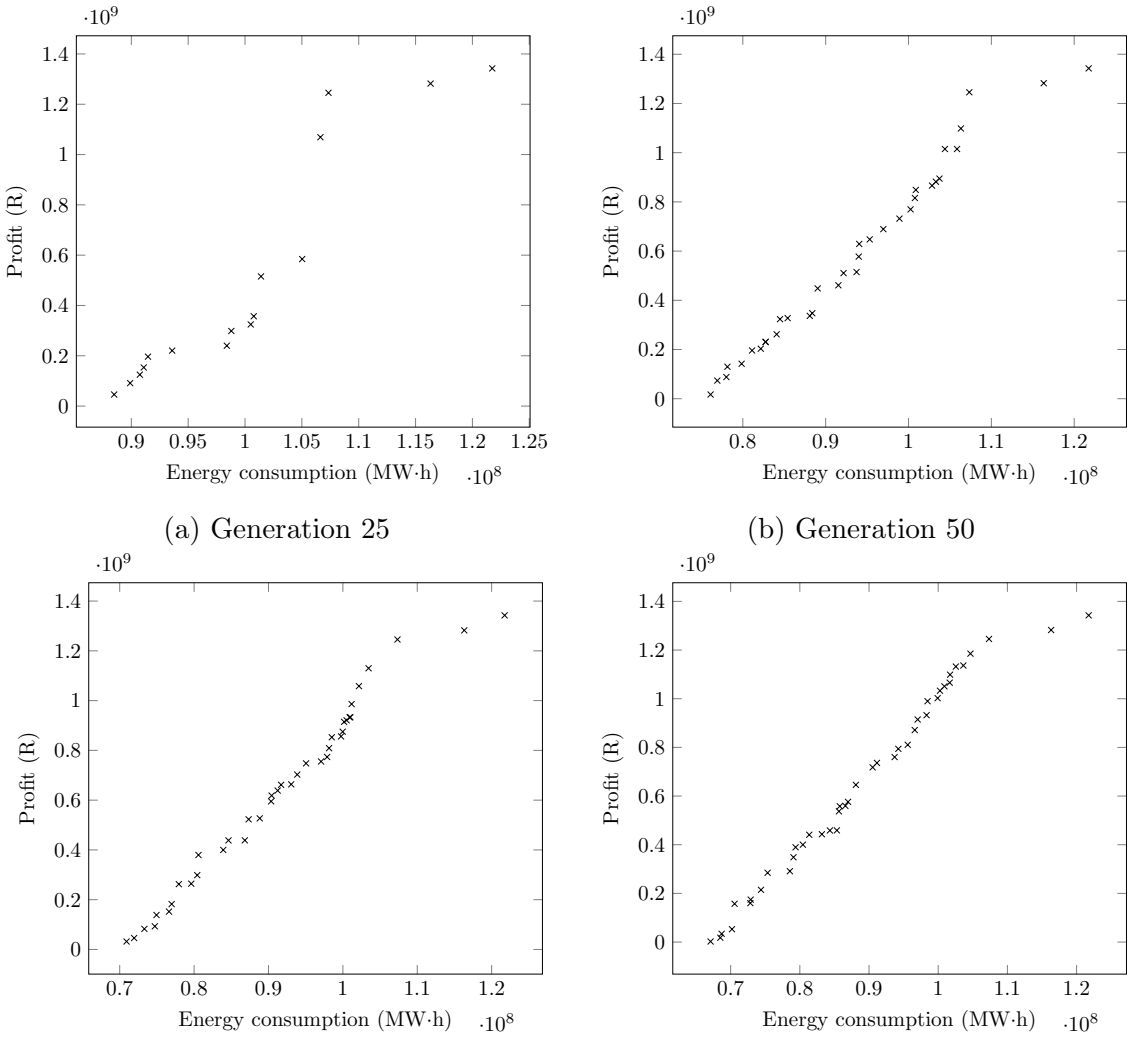


Figure 6.15: Ethylene pipeline pressure for the hourly interval downtime event

6.2.3 Testing the trade-off between profitability and energy consumption

To test the non-dominated sorting genetic algorithm II (NSGA-II) in this category, 100 generations were used in both the daily and hourly interval. Figure 6.16 illustrates the formulation of the Pareto set during the progress of the search for the daily interval and Figure 6.17 for the hourly interval. It can be seen that both intervals converge to form a Pareto set.



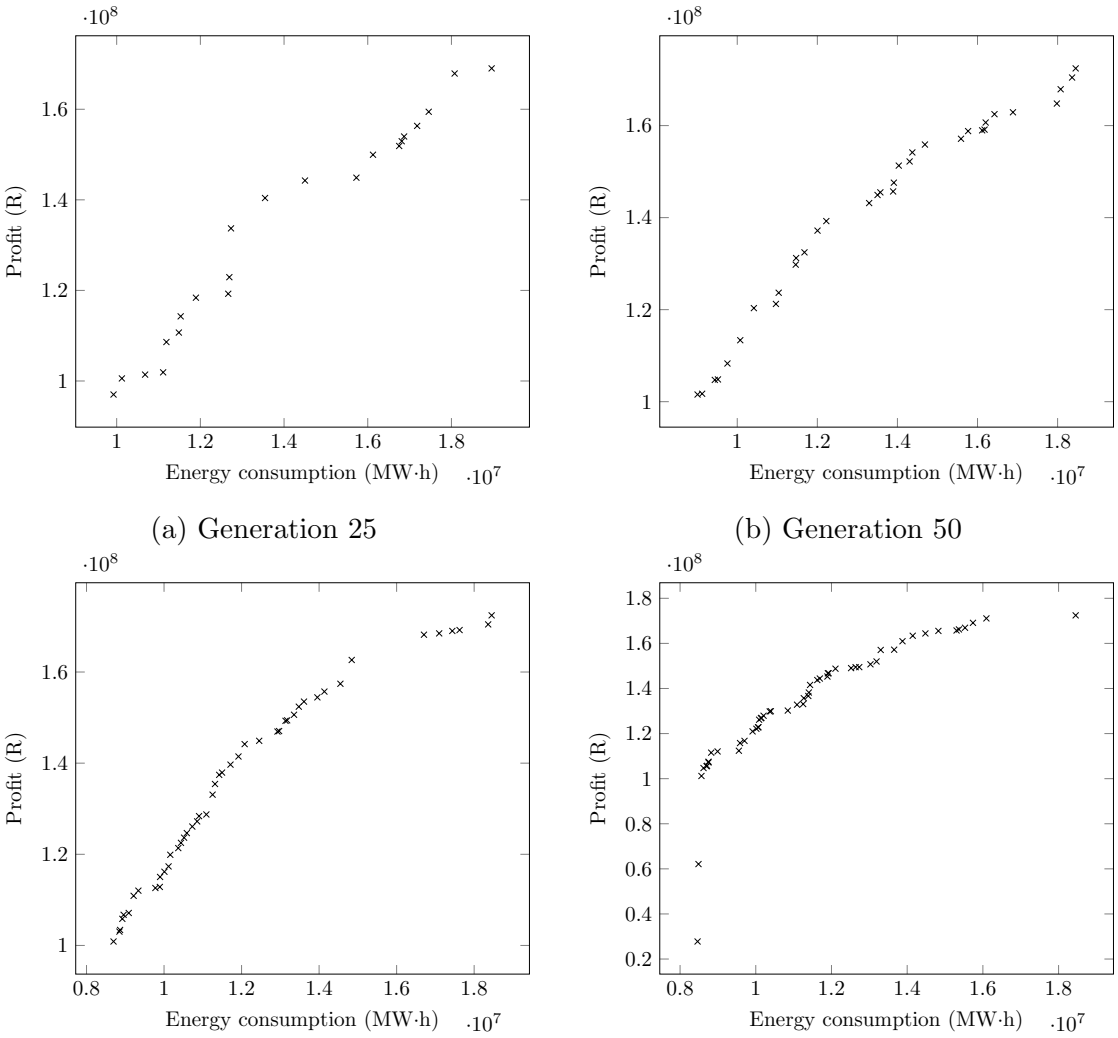
(a) Generation 25

(b) Generation 50

(c) Generation 75

(d) Generation 100

Figure 6.16: Daily interval Pareto set achieved with different generations



(a) Generation 25

(b) Generation 50

(c) Generation 75

(d) Generation 100

Figure 6.17: Hourly interval Pareto set achieved with different generations

Figure 6.18 and Figure 6.19 illustrate the final generation which contains both the Pareto set and the dominated solutions on multiple frontiers. As an exploratory comparison, the most profitable solution on the Pareto front achieved R1 342 541 860 for the daily interval and R172 431 806 for the hourly. These two solutions would rank eight and ninth respectively on Table 6.3 and Table 6.4 listing the top 30 results.

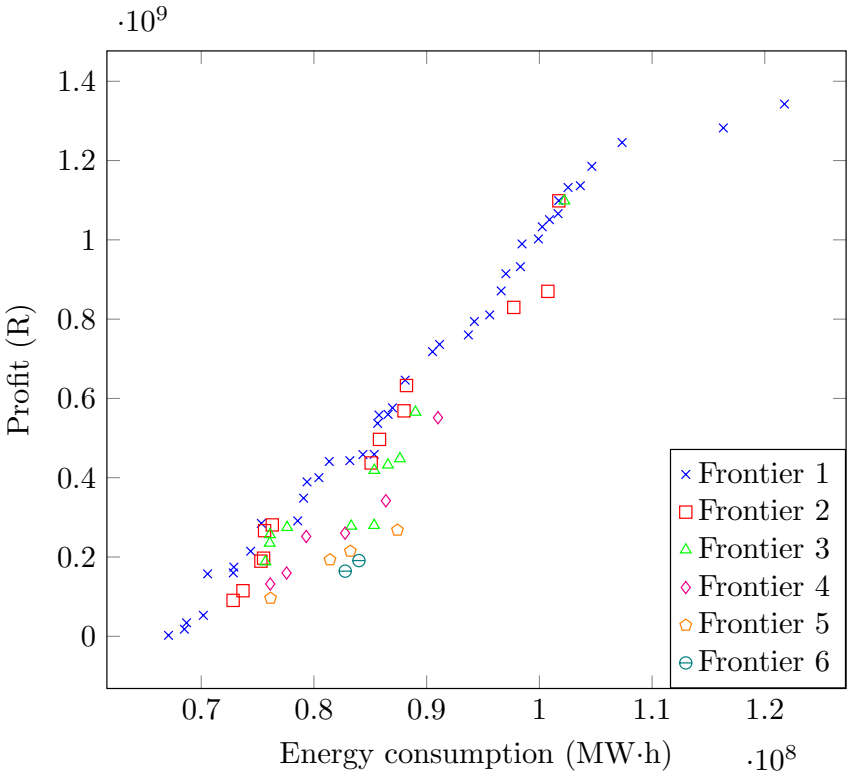


Figure 6.18: Daily interval final generation

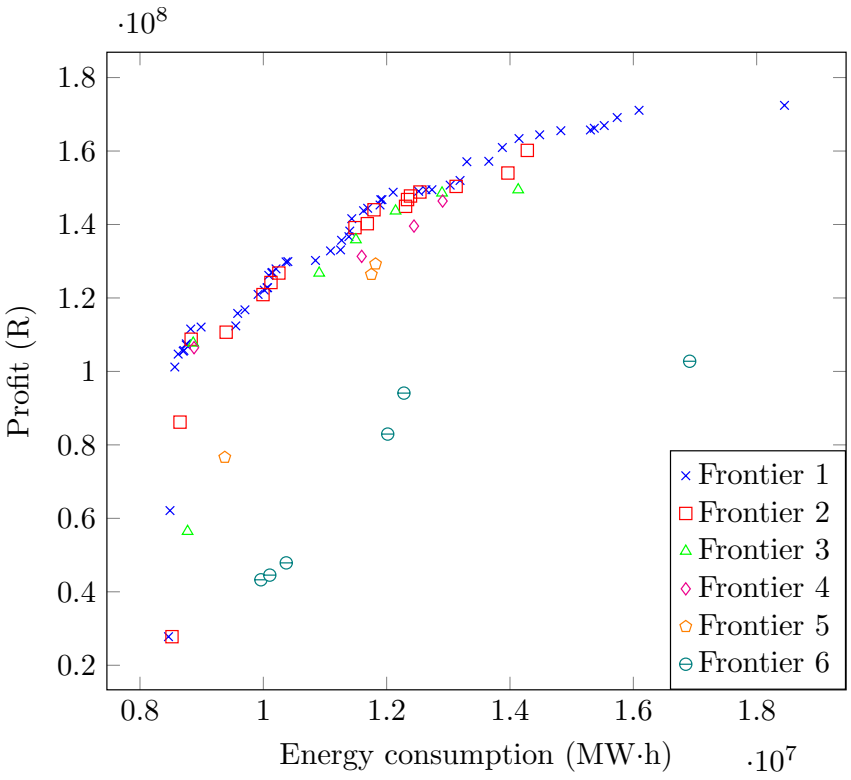


Figure 6.19: Hourly interval final generation

Several performance indicators have been developed to assess the solution of a multi-objective optimisation problem (MOOP). Okabe *et al.* (2003) did a critical survey of performance indicators and from the empirical evaluations found no single existing performance indicator that can describe all the aspects of the solution quality. Talbi (2009) classifies the indicators as convergence-based indicators, diversity-based indicators and hybrid indicators, while recommending that one indicator from each class should be used to evaluate an MOOP.

A review of 57 performance indicators gave prominence to the hyper volume (three objectives) and hyper area difference (two objectives) indicator for the good mathematical properties in dominance and distribution (Audet *et al.*, 2021). They state that the hyper volume and hyper area can be considered the most relevant performance indicator for an MOOP.

The binary counterpart of the hyper volume performance indicator, the hyper area difference, was used to evaluate the performance of the solution. The hyper area difference can be used when the theoretical Pareto set is not known and it is the approximated size of the dominated area, circumventing the objective spaces between two adjacent points and the reference point. The hyper area difference is illustrated in green in Figure 6.20.

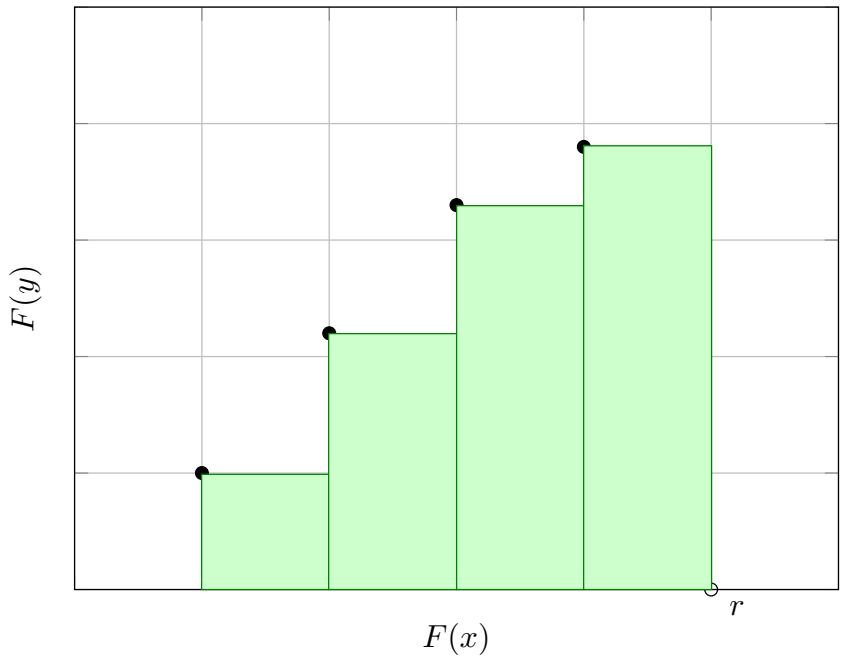


Figure 6.20: Example of the hyper area difference performance indicator

To examine the performance of the algorithm using the hyper area difference performance indicator, 100 replications were performed for each time interval. The mean ($\bar{x}$) and standard deviation (S) for the samples can be calculated as follows:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n} \tag{6.1}$$

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{6.2}$$

The margin of error or half-width at 95% confidence using the t-distribution with $n-1$ degrees of freedom can be stated for both time intervals as:

$$h = t_{n-1;1-\alpha/2}\frac{S}{\sqrt{x}} \tag{6.3}$$

$$\bar{x} \pm 1,9842 S_{\bar{x}} \tag{6.4}$$

The standard error for the sample mean can be expressed as:

$$S_{\bar{x}} = \frac{S}{\sqrt{x}} \tag{6.5}$$

The confidence interval has been calculated and the results are listed in Table 6.9. For the daily interval, the researcher is 95% confident that the hyper area will be between $4{,}620 \times 10^{16}$ and $4{,}891 \times 10^{16}$ and for the hourly interval, between $1{,}605 \times 10^{15}$ and $1{,}630 \times 10^{15}$.

Table 6.9: Confidence interval for the two time intervals

	Daily interval	Hourly interval
Sample size (n)	100	100
Degrees of freedom (df)	99	99
Confidence level	95%	95%
Mean ($\bar{x}$)	$4{,}756 \times 10^{16}$	$1{,}618 \times 10^{15}$
Standard deviation (S)	$6{,}817 \times 10^{15}$	$6{,}445 \times 10^{13}$
Margin of error	$\bar{x} \pm 1,9842 S_{\bar{x}}$	$\bar{x} \pm 1,9842 S_{\bar{x}}$
Standard error ($S_{\bar{x}}$)	$6{,}817 \times 10^{14}$	$6{,}445 \times 10^{12}$
Lower CI	$4{,}620 \times 10^{16}$	$1{,}605 \times 10^{15}$
Upper CI	$4{,}891 \times 10^{16}$	$1{,}630 \times 10^{15}$

6.2.4 Conclusion on algorithm testing

From the above tests, it is possible to find profitable schedules and the researcher accepts that the results are near-optimal. As seen in the unplanned downtime test category, a change in unit availabilities can recommend entirely different schedules and similarly with feedstock availabilities, pipeline pressures and tank levels. Continuous testing with different scenarios will provide further confidence in the recommended schedules and the fixed scenarios can be used as an internal benchmark for future work on the algorithms or future work on the problems.

6.3 Parameter analysis

The algorithms have been developed to accept different input parameters. These parameters have been discussed in different sections in Chapter 3 that range from the population size for the GA to the stopping limit for TS. The following sections list the parameters for the different algorithms and are limited to the top 30 results listed in Table 6.3 and Table 6.4.

6.3.1 Tabu search (TS) parameters

Two parameters can be changed for the TS algorithm in this study; namely, tabu list size and the maximum number of iterations. After preliminary testing, a tabu list size of 100 solutions with 50 iterations was used. Different parameters were selected with the hybrid algorithms and these will be discussed in a later section.

6.3.2 Simulated annealing (SA) parameters

Two parameters can be changed for the SA algorithm in this study; namely, initial temperature and cooling rate. The stopping temperature was set at 0,01 for all jobs.

The researcher started with 100 as the initial temperature and fixed the cooling rate at 0,95 and increased the temperate tenfold for the following job. The temperature of the top two results was then fixed and the cooling rate changed by increasing or decreasing it. Table 6.10 lists the daily interval result parameters and Table 6.11 the hourly interval result parameters.

Table 6.10: Simulated annealing daily interval result parameters

Profit	Run time	Temperature	Cooling
R1 224 520 853	0h, 21m, 57s	10 000 000	0,95
R1 211 576 590	0h, 22m, 45s	1 000 000	0,96

Table 6.11: Simulated annealing hourly interval result parameters

Profit	Run time	Temperature	Cooling
R179 620 070	75h, 35m, 51s	10 000 000	0,99
R159 820 835	7h, 59m, 7s	10 000 000	0,95
R142 748 318	3h, 56m, 44s	1 000 000	0,95

6.3.3 Genetic algorithm (GA) parameters

Three parameters can be changed for the GA in this study; namely, the number of generations, population size and probability of mutation.

The two mutation probabilities used in the study are 0 and 0,7 with an LS mutation on the selected chromosome. Due to the size of the problems, the mutation method caused multiple LS mutations that prevented the job from completing within 24 hours. The remainder of the tests were done without mutation.

Table 6.12 ranks the daily interval GA results by profit and Table 6.13 ranks the daily interval by run time for different combinations of the number of generations and population sizes used. The results varied widely on both the profit achieved and the run time. The parameters, a population size of 400 with 400 generations for the top performing solution, were selected for further use in the hybrid algorithms.

Table 6.14 ranks the hourly interval GA results by profit and Table 6.15 ranks the hourly interval by run time for the different parameter values. The larger combinations had consistently longer run times without higher profit. The parameters, a population size of 120 with 250 generations for the top performing solution, were selected for further use in the hybrid algorithms.

Table 6.12: Genetic algorithm daily interval results for different parameter values, sorted by profit

Profit	Run time	Generations	Population
R1 384 988 520	4h, 14m, 20s	400	400
R1 348 584 420	15h, 20m, 27s	100	4 000
R1 335 219 340	0h, 36m, 54s	400	50
R1 332 734 000	0h, 13m, 43s	100	20
R1 329 543 823	0h, 10m, 26s	100	50
R1 303 820 565	1h, 47m, 35s	200	400
R1 298 072 880	1h, 57m, 13s	400	400
R1 278 366 305	1h, 14m, 15s	100	400
R1 232 988 050	0h, 25m, 20s	200	100
R1 229 056 565	0h, 31m, 53s	100	100
R1 115 859 343	0h, 17m, 19s	100	100
R1 084 935 935	0h, 14m, 56s	100	100
R1 078 870 075	0h, 7m, 5s	200	20
R1 034 769 583	0h, 16m, 40s	100	50
R1 020 576 590	0h, 36m, 24s	100	100

Table 6.13: Genetic algorithm daily interval results for different values of parameters, sorted by runtime

Profit	Run time	Generations	Population
R1 348 584 420	15h, 20m, 27s	100	4 000
R1 384 988 520	4h, 14m, 20s	400	400
R1 298 072 880	1h, 57m, 13s	400	400
R1 303 820 565	1h, 47m, 35s	200	400
R1 278 366 305	1h, 14m, 15s	100	400
R1 335 219 340	0h, 36m, 54s	400	50
R1 020 576 590	0h, 36m, 24s	100	100
R1 229 056 565	0h, 31m, 53s	100	100
R1 232 988 050	0h, 25m, 20s	200	100
R1 115 859 343	0h, 17m, 19s	100	100
R1 034 769 583	0h, 16m, 40s	100	50
R1 084 935 935	0h, 14m, 56s	100	100
R1 332 734 000	0h, 13m, 43s	100	20
R1 329 543 823	0h, 10m, 26s	100	50
R1 078 870 075	0h, 7m, 5s	200	20

Table 6.14: Genetic algorithm hourly interval results for different values of parameters, sorted by profit

Profit	Run time	Generations	Population
R167 192 623	0h, 5m, 29s	250	120
R163 598 853	1h, 11m, 35s	250	4 000
R162 076 055	0h, 2m, 11s	60	120
R161 754 330	1h, 9m, 58s	250	400
R161 496 950	1h, 16m, 50s	500	4 000
R159 510 625	1h, 10m, 3s	250	4 000
R159 429 568	1h, 12m, 26s	250	4 000
R159 091 690	1h, 15m, 8s	500	4 000
R158 356 993	1h, 14m, 42s	500	400
R156 426 428	0h, 5m, 28s	500	120
R151 928 318	1h, 14m, 54s	500	4 000

Table 6.15: Genetic algorithm hourly interval results for different values for parameters, sorted by runtime

Profit	Run time	Generations	Population
R161 496 950	1h, 16m, 50s	500	4 000
R159 091 690	1h, 15m, 8s	500	4 000
R151 928 318	1h, 14m, 54s	500	4 000
R158 356 993	1h, 14m, 42s	500	400
R159 429 568	1h, 12m, 26s	250	4 000
R163 598 853	1h, 11m, 35s	250	4 000
R159 510 625	1h, 10m, 3s	250	4 000
R161 754 330	1h, 9m, 58s	250	400
R167 192 623	0h, 5m, 29s	250	120
R156 426 428	0h, 5m, 28s	500	120
R162 076 055	0h, 2m, 11s	60	120

6.3.4 Hybrid algorithm parameters

The three hybrid algorithms used in this study: namely, GA with LS, GA with GS and GA with TS, can take different parameters. The LS and GS algorithms do not have parameters to change but the GA they extend does. The parameter values from the most profitable GA solution for the daily interval and hourly interval will be used and are listed in Table 6.16.

The parameter values for the GA with TS algorithm is listed for the daily interval in Table 6.17 and for the hourly interval in Table 6.18. The tabu list size did not make a big difference in profit but having a too small tabu list size resulted in a lower profit. The higher number of iterations had an increased profit at the expense of time.

Table 6.16: Genetic algorithm parameter values for the two intervals

Interval	Generations	Population
Daily	400	400
Hourly	250	120

Table 6.17: Hybrid genetic algorithm with tabu search daily interval results for different parameter values

Profit	Run time	Generations	Population	Tabu list	Iterations
R1 410 779 580	4h, 45m, 31s	400	400	100	50
R1 389 168 880	15h, 34m, 8s	400	400	100	100

Table 6.18: Hybrid genetic algorithm with tabu search hourly interval results for different parameter values

Profit	Run time	Generations	Population	Tabu list	Iterations
R176 600 338	2h, 17m, 21s	250	120	50	10
R176 492 128	2h, 13m, 52s	250	120	100	10
R175 542 540	2h, 16m, 2s	250	120	200	10
R168 391 905	2h, 15m, 47s	250	120	20	10
R164 118 015	0h, 30m, 37s	250	120	100	2
R162 879 413	0h, 31m, 56s	250	120	50	2
R161 492 853	0h, 31m, 24s	250	120	200	2
R142 854 915	0h, 30m, 12s	250	120	10	2

6.3.5 Parameter analysis conclusion

The parameter setting was only changed in the two scenarios used in this study and different scenarios can potentially give different results.

The algorithms were developed to make use of caching to increase the performance of the repetitive functions. Caching stores the result of a frequent or computationally expensive function with a specific input for future matching requests to read the stored

results. The impact of caching should be considered when changing a parameter and expecting a change in the results.

The researcher is aware that the results from the different parameter settings are not conclusive but they indicate an opportunity that parameter tuning can provide improved results. This will be investigated in the next section.

6.4 Parameter tuning

The parameter analysis indicated an opportunity to further improve the results by tuning the parameters. The genetic algorithm was selected for both time intervals to investigate the significance of different parameters. A non-parametric test, the Friedman test, can be performed to test the difference between multiple parameter groups and if a significant difference is present, a post hoc test, the Nemenyi test, can determine the relative performance among the parameter groups (Derrac *et al.*, 2011).

Nel (2021) applied the Friedman test and Nemenyi test to three multi-objective algorithms to perform parameter evaluations and relative performance comparisons. Wang *et al.* (2021) compared 11 common nature-inspired optimisation algorithms with each other using the Friedman test and the Nemenyi test. Similarly, the Friedman test and Nemenyi test combination is selected for this study with the difference being on the parameters for the GA.

Therefore, the two hypotheses considered for this experiment are
H_0 : All parameter groups are the same,
H_a : At least one parameter group is different.

6.4.1 Parameter tuning experimental setup

The earlier parameter analysis provided a basis for selecting the parameters in the experiment. For the number of generations, 100, 300, and 500 were selected and for the population size, 100, 300, 500, 700, and 900 were selected. This results in 15 parameter groups as listed in Table 6.19 and they were tested for both intervals, daily and hourly, with ten replications per parameter group. The level of significance used in the experiment was set at $\alpha = 0{,}05$. The Friedman test statistic can be defined as

$$F_T = \left(\frac{12}{b(k)(k+1)} \sum_{j=1}^{k} T_j^2 \right) - 3b(k+1) \tag{6.6}$$

where k is the number of treatments, with b the number of blocks and T_j^2 the squared sum of the ranks for the sample treatment j. Derrac *et al.* (2011) states the rule of

thumb as $b > 10$ and $k > 5$. For this study, $b = 15$ and $k = 10$ where b is the number of groups and k the number of the replications.

Table 6.19: List of parameter group combinations

#	Number of generations	Population size	Group combination
1	100	100	G100:P100
2	300	100	G300:P100
3	500	100	G500:P100
4	100	300	G100:P300
5	300	300	G300:P300
6	500	300	G500:P300
7	100	500	G100:P500
8	300	500	G300:P500
9	500	500	G500:P500
10	100	700	G100:P700
11	300	700	G300:P700
12	500	700	G500:P700
13	100	900	G100:P900
14	300	900	G300:P900
15	500	900	G500:P900

6.4.2 Parameter tuning experiment results

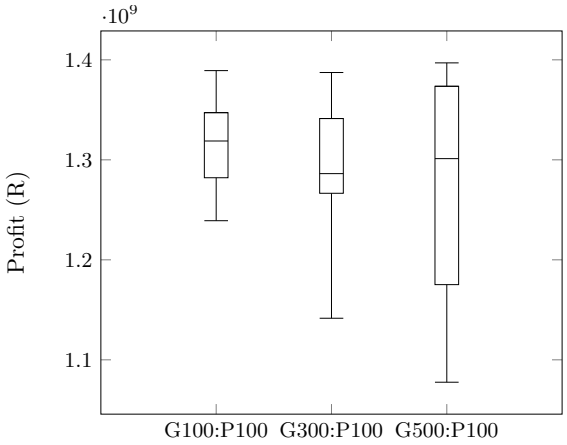
The profit achieved for the daily and hourly parameter groups are graphically summarised using box plot charts to show the distributional characteristics of a group. Notably, with the daily interval results, on all the box plots in Figure 6.21, the 100 generations performs poorly while the larger number of generations with the larger population size performs better. The hourly interval results in Figure 6.22 indicate visually that a larger generation size with a larger population size achieves higher profit with the exception of the population size of 100.

Table 6.20: Nemenyi test P-values for the daily interval

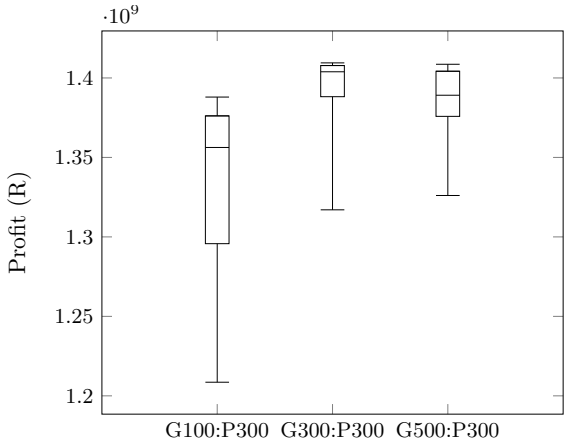
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	–	0,900	0,900	0,900	0,057	0,275	0,900	0,030	0,480	0,900	0,057	0,015	0,900	0,078	0,078
2		–	0,900	0,900	0,015	0,103	0,900	0,007	0,221	0,674	0,015	0,003	0,900	0,021	0,021
3			–	0,900	0,118	0,445	0,900	0,067	0,641	0,900	0,118	0,035	0,900	0,153	0,153
4				–	0,275	0,674	0,900	0,174	0,867	0,900	0,275	0,103	0,900	0,337	0,337
5					–	0,900	0,275	0,900	0,900	0,900	0,900	0,900	0,057	0,900	0,900
6						–	0,674	0,900	0,900	0,900	0,900	0,900	0,275	0,900	0,900
7							–	0,174	0,867	0,900	0,275	0,103	0,900	0,337	0,337
8								–	0,900	0,899	0,900	0,900	0,030	0,900	0,900
9									–	0,900	0,900	0,900	0,480	0,900	0,900
10										–	0,900	0,770	0,900	0,900	0,900
11											–	0,900	0,057	0,900	0,900
12												–	0,015	0,900	0,900
13													–	0,078	0,078
14														–	0,900
15															–

The Friedman test is used to assess whether there is a significant difference between the parameter groups. For the daily interval, the Friedman test (F_T) statistic is 67,31 with a P-value of $5{,}9 \times 10^{-9}$. The $\tilde{\chi}^2$ critical value is therefore 23,68 and the daily interval null hypothesis (H_0) is rejected in favour of the alternative hypothesis. For the hourly interval, the Friedman test (F_T) statistic is 69,26 with a P-value of $2{,}63 \times 10^{-9}$. The $\tilde{\chi}^2$ critical value is therefore 23,68 and the hourly interval null hypothesis (H_0) is rejected in favour of the alternative hypothesis.

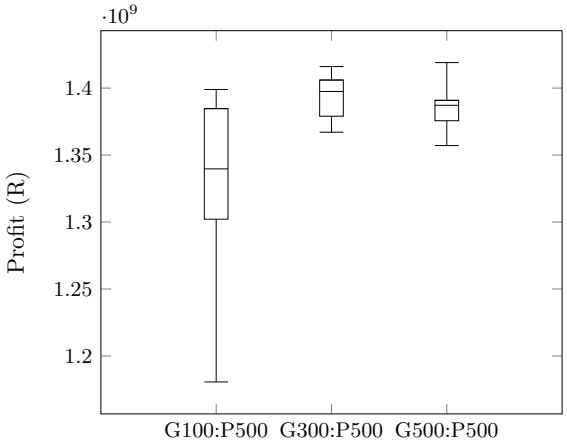
With both experiments being significantly different, the Nemenyi test is applied to determine the relative performance among the parameter groups. Consider Table 6.20 for the daily interval and Table 6.21 for the hourly interval with a P-values at a 5% level of significance indicated in red. As an alternative graphical representation, the P-values are represented in a heat map in Figure 6.23 and Figure 6.24.



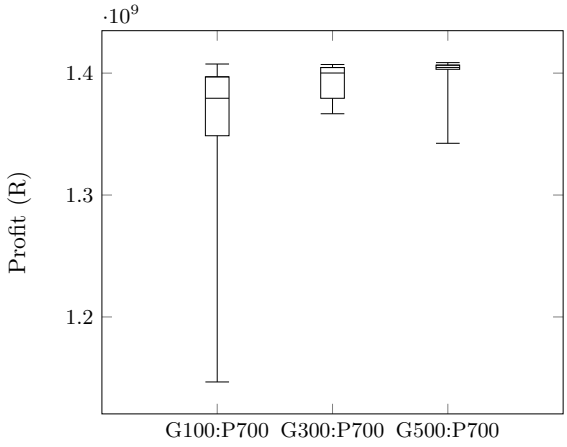
(a) Daily interval group 1



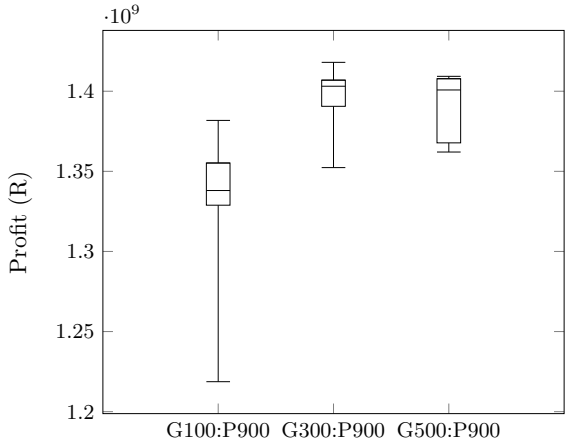
(b) Daily interval group 2



(c) Daily interval group 3

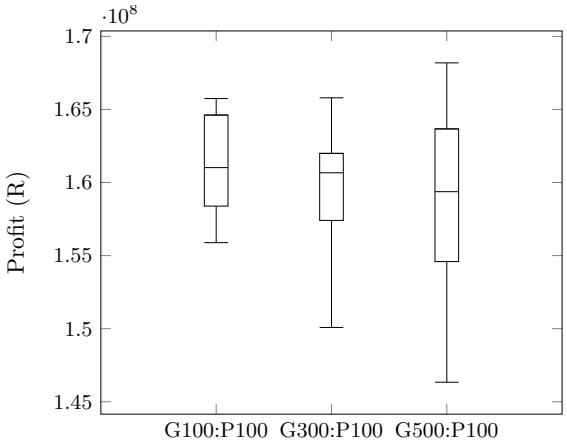


(d) Daily interval group 4

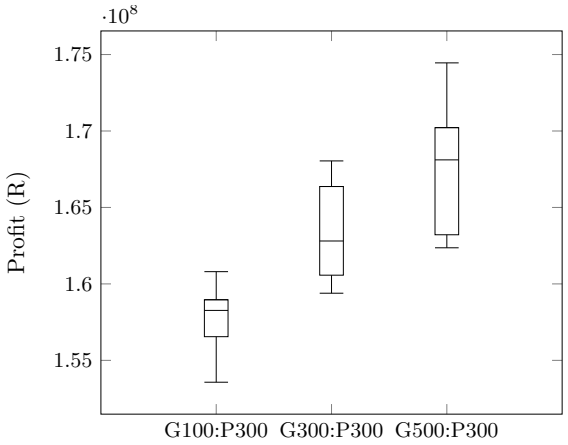


(e) Daily interval group 5

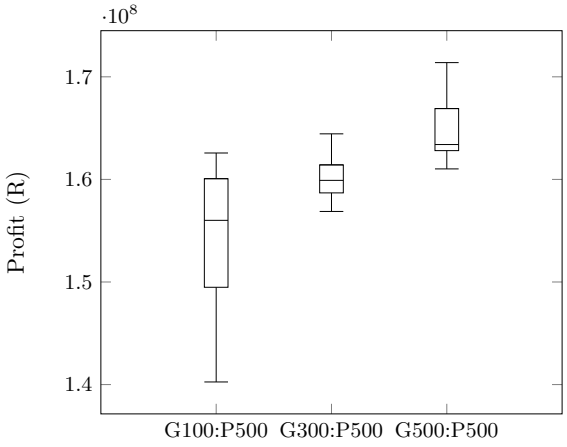
Figure 6.21: Daily interval box plots of the profit achieved grouped by population size



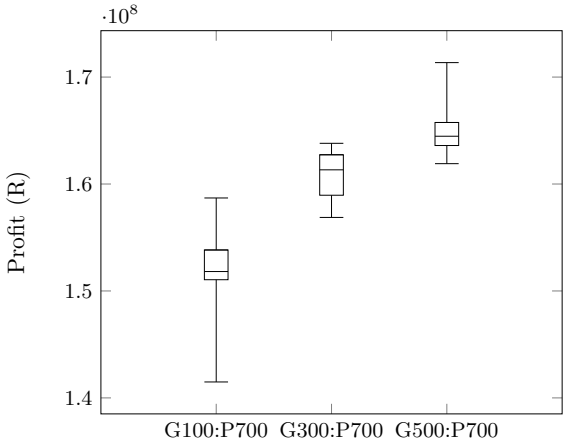
(a) Hourly interval group 1



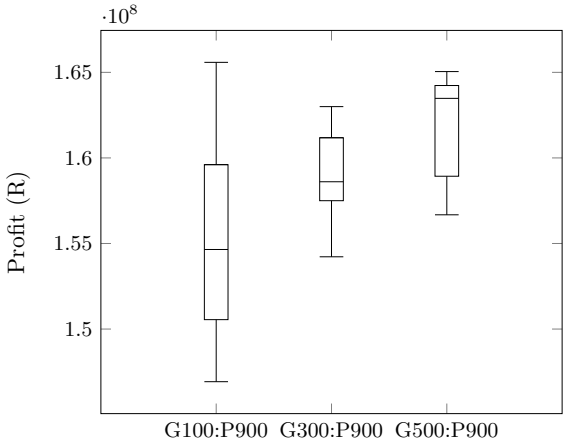
(b) Hourly interval group 2



(c) Hourly interval group 3



(d) Hourly interval group 4



(e) Hourly interval group 5

Figure 6.22: Hourly interval box plots of the profit achieved grouped by population size

Table 6.21: Nemenyi test P-values for the hourly interval

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	–	,900	,900	,900	,900	,513	,609	,900	,900	,174	,900	,706	,835	,900	,900
2		–	,900	,900	,900	,134	,900	,900	,577	,577	,900	,275	,900	,900	,900
3			–	,900	,900	,103	,900	,900	,513	,641	,900	,221	,900	,900	,900
4				–	,445	,005	,900	,900	,067	,900	,900	,015	,900	,900	,802
5					–	,900	,118	,900	,900	,012	,900	,900	,275	,900	,900
6						–	,001	,153	,900	,001	,409	,900	,002	,067	,706
7							–	,900	,008	,900	,706	,001	,900	,900	,409
8								–	,609	,545	,900	,305	,900	,900	,900
9									–	,001	,899	,900	,030	,409	,900
10										–	,247	,001	,900	,738	,078
11											–	,609	,900	,900	,900
12												–	,006	,153	,899
13													–	,900	,641
14														–	,900
15															–

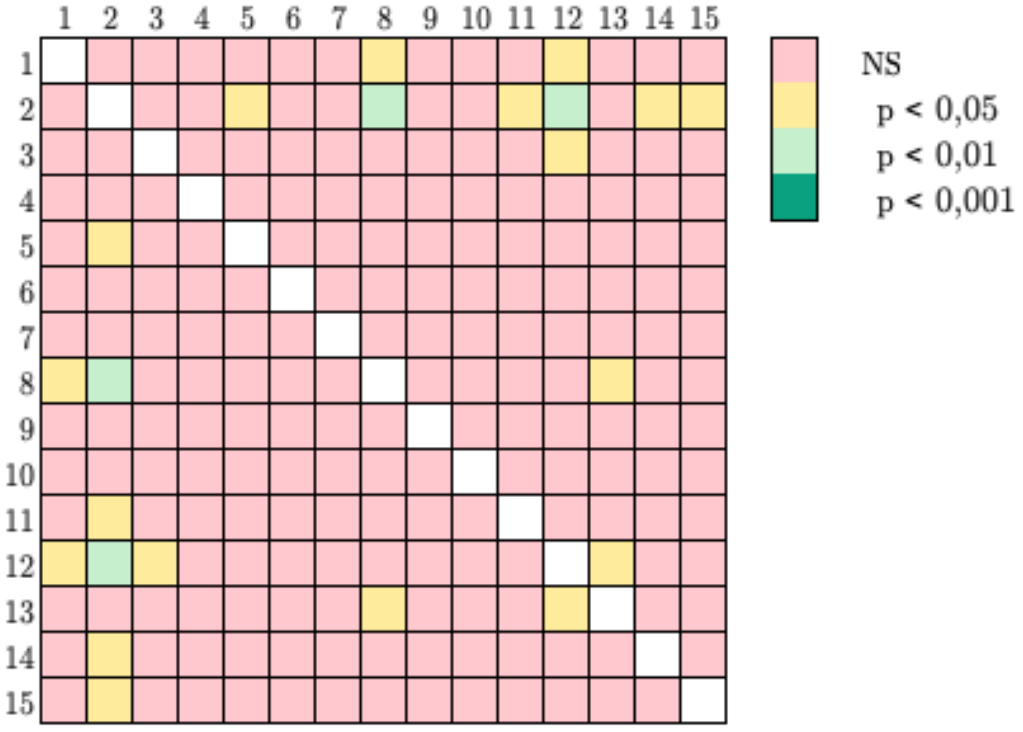


Figure 6.23: P-values for the daily interval represented in a heat map

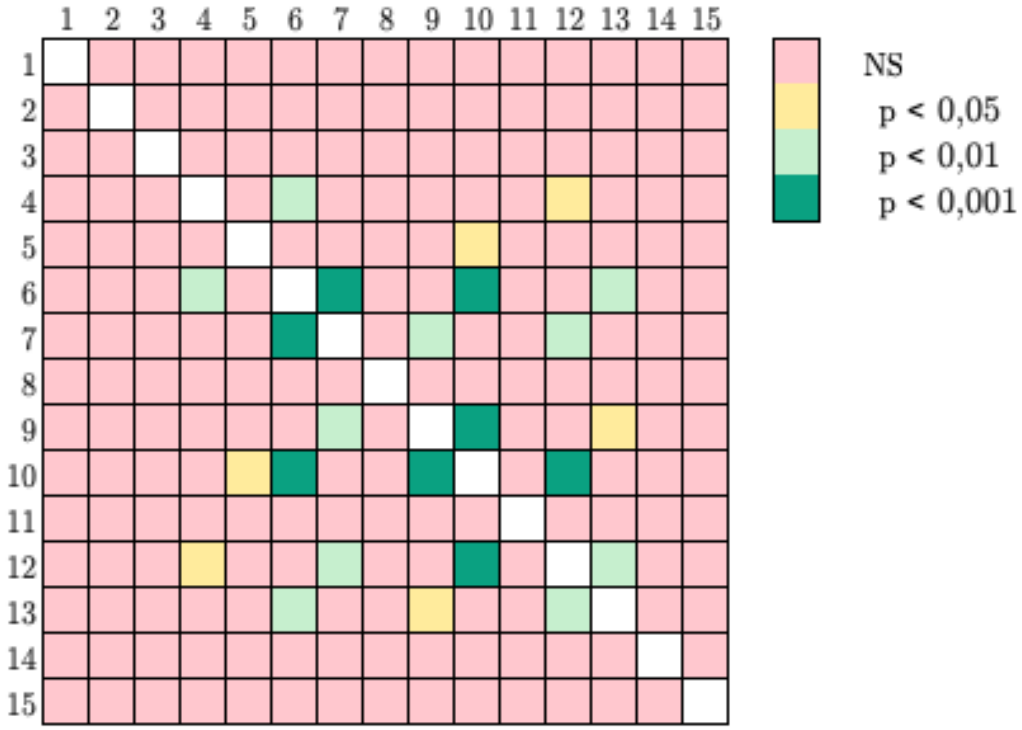


Figure 6.24: *P*-values for the hourly interval represented in a heat map

The final step to identify the best parameter group for the two intervals is to determine the statistical significance between the differences in average rank by using the critical distance (CD) (Nemenyi, 1963). The CD values are calculated using

$$CD = q_\alpha \sqrt{\left(\frac{(k)(k+1)}{6N}\right)} \tag{6.7}$$

where k is the number of blocks, N is the number of samples and q_α the critical value. The CD for both intervals are 1,751 and a succinct representation of the rank with the CD is given on CD diagrams in Figure 6.25 and Figure 6.26. The bold horizontal bars connect ranks that are statistically indistinguishable at $\alpha = 0{,}05$.

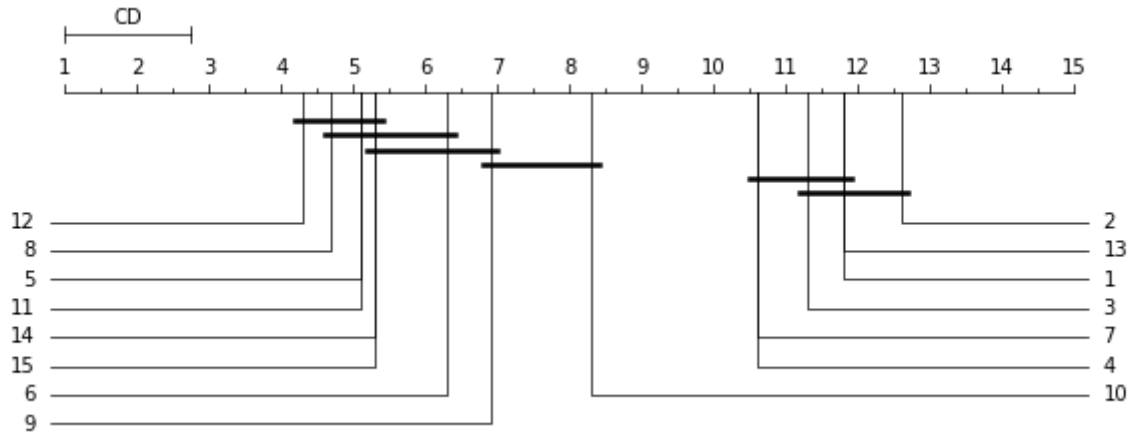


Figure 6.25: CD diagram for daily interval

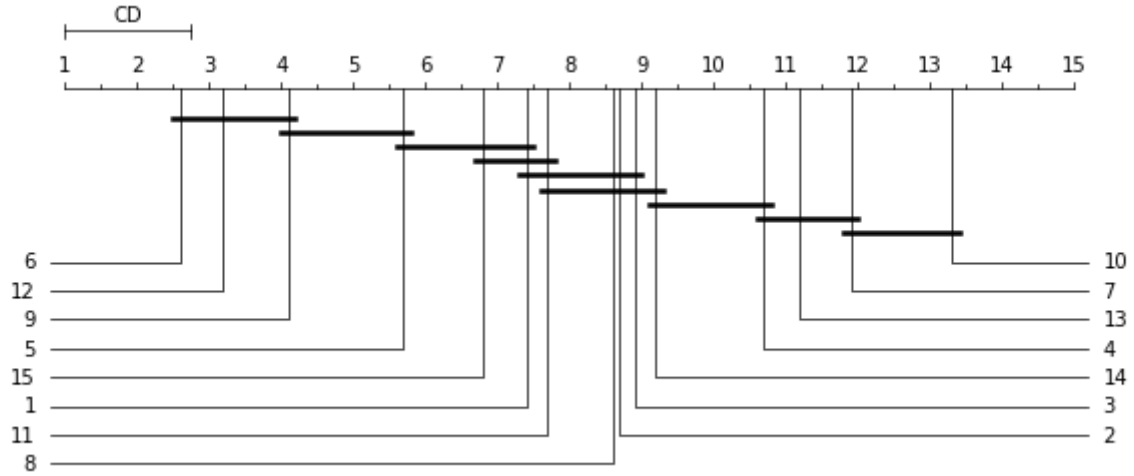


Figure 6.26: CD diagram for hourly interval

6.4.3 Parameter tuning conclusion

Based on the results, it can be concluded that groups 12, 8, 5, and 11 are the best performing parameter combination groups for the daily interval and groups 6, 12, and 9 are the best performing parameter combination groups for the hourly interval. Table 6.22 lists the statistically indistinguishable best performing groups with the associated parameter combination. The researcher would recommend using parameter combination group 12 (G500:P700) for the daily interval and parameter combination group 6 (G500:P300) for the hourly interval albeit they are statistically indistinguishable.

Table 6.22: Best performing parameter combination groups based on the Nemenyi test

Daily interval		Hourly interval	
#	Parameter combination	#	Parameter combination
12	G500:P700	6	G500:P300
8	G300:P500	12	G500:P700
5	G300:P300	9	G500:P500
11	G300:P700		

6.5 Evaluation of the decision support system

This section describes the evaluation of the decision support system. By using results validation it is possible to establish if the output data from the proposed system closely resemble the expected output (Law, 2015). This could be carried out by comparing the proposed system with an existing system, another model or expert opinion. From the available options, the only possible comparison for this study is by using expert opinion.

Result validation by expert opinion is also known as face validation. The process involves knowledgeable people subjectively comparing the system's behaviour and models to determine whether the results are reasonable (Roungas *et al.*, 2018).

Law (2015) highlight that should the expert opinion have the ability to validate the output accurately, there would be no need for a model.

Two expert opinion roles within Sasol can determine if the system's behaviour and results are reasonable. Firstly, an operations subject matter expert that has experience within the C_2 value chain and secondly, an operations researcher that has done optimisation work on the C_2 value chain.

To evaluate the system, interviews were conducted with 11 subject matter experts in operations and with operations researchers (Ethical clearance reference number: ING-2020-17314). Overall, the system's behaviour and results were positively received. A few key points were highlighted during the interviews that will be discussed next.

6.5.1 SME responses after evaluating the reports

The subject matter experts made the following observations during interviews after evaluating the reports:

- Start-up production should not be included in the profit function. The product produced during start-up is not within the product specification.
- Consider switching from decimal to integers to reduce the problem size, it could improve the results.
- Models can suggest unsustainable values nearing the end of the 14-day horizon.
- Some models in the company have taken over 10 years to reach their current maturity. Continuous changes are requested and some can cause a total re-design.
- Confidence in the system will increase over time as different scenarios are tested.
- Testing different shutdown and turnaround strategies on the system is possible.
- Near-optimal solutions in a reasonable amount of time are sufficient.
- Margin sensitivity should be tested on the system as product prices change regularly in the market.
- This is not the final version of the system but a good starting point and it will be more valuable to focus on the 14-day hourly interval.

6.5.2 SME responses after evaluating the overall system

The subject matter experts made the following observations during interviews on the overall system:

- The web-based platform is preferred over an installed application;
- The DSS looks user-friendly;
- SMEs approved the ability that additional models could be added to the same system and not have some models in different software;
- Documentation or a manual describing the system is needed.

6.5.3 Opinions on modelling and the use of models

Modelling is highly dependent on a variety of people. There are people who use the model, people that developed the model and the person or manager sponsoring the model. A change in requirements by one of them can result in the abandoning of a model. Restructuring in the company or rotating of people to different positions has caused the neglect of many models. Models are usually strongly tied to a single engineer, statistician or operations researcher. Handing over models to a different person has often been unsuccessful. Commercial off-the-shelf software has been proposed in the past to address some of these problems but this practice has left the company with a list of expensive partially used software solutions that is difficult to remove. This has many times resulted in people resorting to using spreadsheets in an attempt to find answers to the problem.

To quote an analogy from the interview on why some working models are discontinued over time, "Some models are like driving with a GPS for navigation. Once the majority of routes have been learnt, a GPS is no longer needed by the driver". It is, however, important to embed and document the corporate memory so that it is not lost or misunderstood over time.

6.5.4 Additional requirements

The SMEs recommended including additional details such as:

1. decoking of furnaces;
2. different modes of production;
3. transitions between grades;
4. warm and cold start-ups;
5. production schedules per product.

This concludes evaluation of the decision support system, which will be followed by an improvement recommended by the SMEs.

6.6 Decision variable recommendation

Based on the recommendation of the subject matter experts, switching from using decimals to integers in the decision variables could improve the results. The assumption is that if the search space is reduced, the algorithms could have improved results

by achieving a higher profit or shorter run times. A non-parametric test such as the Mann-Whitney-Wilcoxon test is used for this comparison as it allows for pairwise comparison of the median values of two distributions (Derrac *et al.*, 2011). The two hypotheses considered for this experiment are
H_0 : Switching from decimals to integers has no effect,
H_a : Switching from decimal to integers improved the performance.

6.6.1 Decision variable experimental setup

The highest performing parameters listed in Table 6.22 were used to compare the daily and hourly interval over 40 replications. The level of significance used in the experiment is set at $\alpha = 0{,}05$. The mean (μ_W) can be defined as:

$$\mu_W = \frac{n_1(n_1 + n_2 + 1)}{2} \tag{6.8}$$

where n_1 is the count of the decimal sample and n_2 is the count of the integer sample. The standard deviation (σ_W) can be defined as

$$\sigma_W = \sqrt{\frac{n_1 n_2(n_1 + n_2 + 1)}{12}}. \tag{6.9}$$

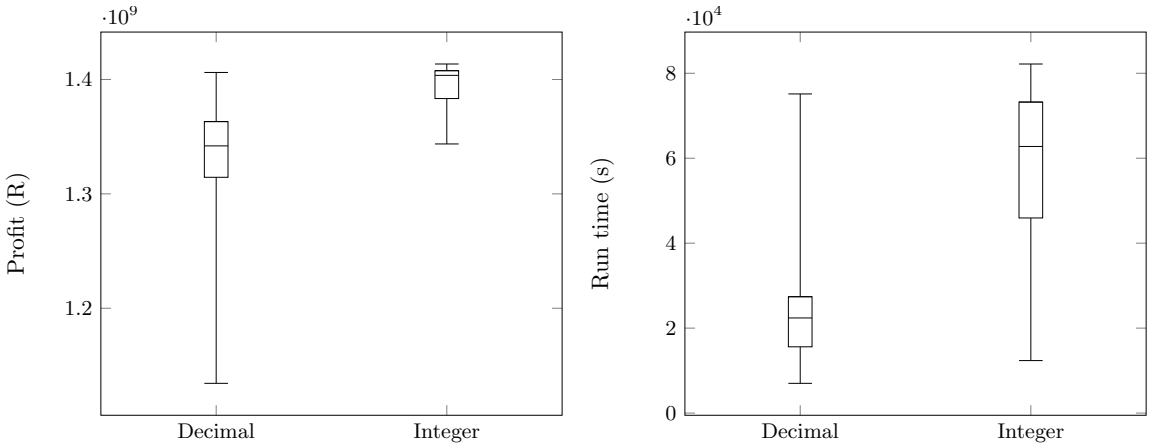
The Z-value test statistic can be defined as

$$Z = \frac{W - \mu_W}{\sigma_W} \tag{6.10}$$

where W is the rank sum for the decimal sample.

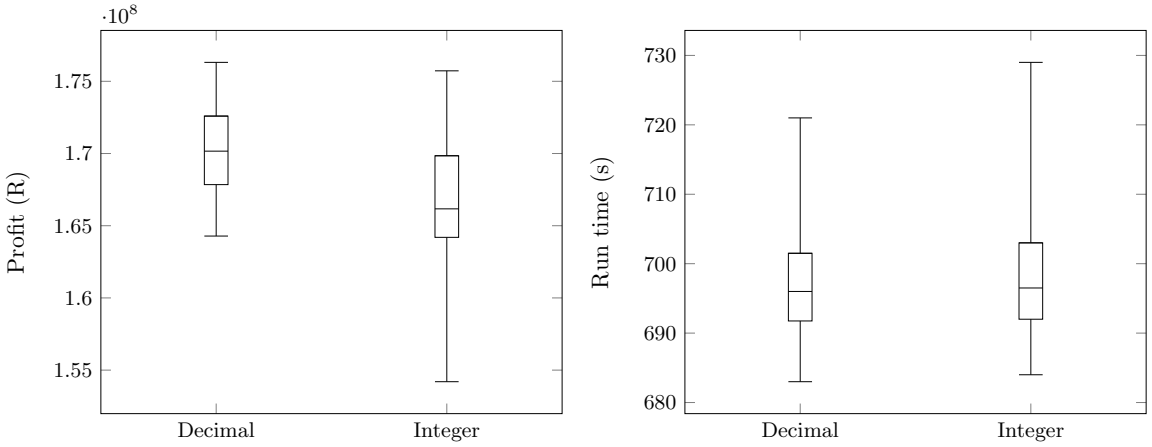
6.6.2 Decision variable experiment results

The results for the decimal and integer experiment can be graphically summarised in box plots in Figure 6.27 and Figure 6.28. Note that when interpreting the box plots, a higher profit or a lower run time is considered an improvement.



(a) Daily interval profit comparison (b) Daily interval run time comparison

Figure 6.27: Daily interval box plots of the profit achieved grouped by population size



(a) Hourly interval profit comparison (b) Hourly interval run time comparison

Figure 6.28: Hourly interval box plots of the profit achieved grouped by population size

The Mann-Whitney-Wilcoxon test is used for a pairwise comparison. The samples are first stacked, then ranked within the stack and then the sum of the ranks of the two samples is calculated. The sums of the ranks are listed in Table 6.23. Note that when interpreting the sum of ranks table, a higher sum of ranks on both profit and run time is considered to be superior.

Table 6.23: Sum of ranks for the four experiments

	Daily interval		Hourly interval	
Sample	Profit	Run time	Profit	Run time
Decimal	953	2 255	2009	1 651,5
Integer	2 287	985	1231	1 588,5

For the daily interval profit comparison, the Z-value is -6,418 with a P-value of $6{,}894 \times 10^{-11}$. The researcher rejects the null hypothesis (H_0) as the data indicates an increase in profit when switching from decimals to integers. Considering the run time comparison, the Z-value is 6,11 with a P-value of 1. The researcher fails to reject the null hypothesis (H_0) as there is not a significant increase in run time.

For the hourly interval profit comparison, the Z-value is 3,743 with a P-value of 1. The researcher fails to reject the null hypothesis (H_0) as the data does not indicate an increase in profit when switching from decimals to integers. The sum of ranks indicates the contrary; the profit has decreased. Considering the run time comparison, the Z-value is 0,303 with a P-value of 0,619. The researcher fails to reject the null hypothesis (H_0) as there is not a significant decrease in run time.

6.6.3 Decision variable experiment conclusion

The data supports the recommendation to switch from decimals to integers in order to improve the results, although only for the daily interval and at a significant increase in run time. It is a noteworthy outcome that a reduction in the search space can increase the run time. It is not apparent what caused the run time to increase and a detailed profiling of the algorithm could provide further insight. For the hourly interval, the researcher deduced from the data that due to the proposed encoding scheme, the search space had already been reduced indirectly. The data suggests that the encoding scheme decoupled the interaction between the search space and the algorithm.

6.7 Synthesis of results

From the two test scenarios, it is evident that the genetic algorithm (GA) paired with a single-solution base consistently achieved higher profits than the other algorithms. The size of the problem presented challenges throughout the study. First, the local search (LS), tabu search (TS) and simulated annealing (SA) initially performed poorly due to the size of the problem and the evaluation process. Evaluating all the

neighbouring solutions is computationally expensive and time-intensive. The greedy search (GS) was added to accept an improvement move before evaluating all the neighbours. After profiling the performance of the algorithms, implementing caching of the functions used in the algorithms and evaluating the neighbourhood over multiple processor cores, the performance increased greatly. Additional profiling of the performance of the code could further improve any one of the algorithms.

Secondly, a change or error in the code could have a significant impact, from requiring the implementation of the change on 20 algorithms to testing everything thoroughly.

Thirdly, the algorithm parameters setting plays a critical role in the performance and it is recommended that parameter tuning is done on the final developed and optimised state of the algorithms. Changes in the code can influence the parameter settings and in some cases reduce or increase their sensitivity.

The last challenge was on the required changes in the function to make use of caching or parallel processing. For caching, this entailed splitting functions to separate random, non-hashable data as functions with randomisation cannot be cached. For parallel processing, careful design of the function is required to ensure that only the data that is needed is passed to all the processors. For large optimisation problems, splitting the data preparation to only parallel process the tasks that are computationally expensive, will reduce the overhead that can cause memory leaks.

Based on the results, the researcher can make the following conclusions:

- The top 10 results for the daily and hourly intervals are within 10% of each other and any of the algorithms in the top 10 could be used. If only one algorithm should be selected to be used for future work, the researcher would select the GA with a GS for both hourly and daily intervals. The flexibility and customisation that is possible with GAs allows for a wide range of changes that can lead to further improvements. Furthermore, the exploitation of the GS complements the GA well for this problem.

- The problem in this study can be adapted to include energy consumption as a second criterion to provide additional information when making decisions.

- Even though the hybrid GA with a multilayer perceptron (MLP) neural network did not obtain favourable results, the integration of combinatorial optimisation and machine learning should be further explored. When combined, the two research fields can benefit from the state-of-the-art algorithms, theoretical guarantees and other performance incentives (Bengio *et al.*, 2021).

- Developing the algorithms to make use of caching or parallel processing decreases the run time significantly.
- The experiment on the decision variable recommendation provided two insights when working with a large search space; a reduction of the search space can yield higher profit and the proposed encoding scheme indirectly reduces the search space.

6.8 Chapter summary

The chapter started by specifying the tests that are needed to verify the code used in the study. The performance of each of the algorithms was then compared with the others in relation to the questions listed in Chapter 1. Then, an analysis of the parameters used with the best performing algorithms was carried out, followed by the results from parameter tuning. Thereafter, an evaluation of the decision support system (DSS) by subject matter experts and subsequently, an experiment based on a recommendation from the subject matter experts to conclude the verification and evaluation of the study. The study is concluded in the next chapter.

Chapter 7

Conclusion and recommendations

This chapter consists of three sections. First, a summary of the study is provided, then contributions that emanated from this study are presented and lastly opportunities for future work are discussed.

7.1 Summary of the study

Chapter 1 provided a background to the complexities of chemical manufacturing. The term "value chain" was introduced and a simplified C_2 value chain was presented.

The study was done at the Sasol organisation and a brief introduction was given to the company. The focus of the study, the C_2 value chain, was explained in the context of Sasol. The explanation identified two problems in the C_2 value chain and mentioned that these had been addressed in Sasol with varying success. The decision-making on these problems is needed with two time horizons, 14 days and 90 days. The 14-day time horizon decision-making is time-sensitive, requiring hourly oversight. This raised the need for a decision support system (DSS) that integrates the feedstock distribution problem and the intermediate feedstock distribution problem for the two time horizons using daily and hourly time intervals.

From the stated need, a research task was identified and objectives formulated to achieve this task. The research task was formulated as follows:

Develop an integrated decision support system that maximises profit in the Sasol C_2 value chain for hourly and daily decision-making.

Chapter 2 introduced scheduling and presented a chronological review of scheduling literature in the process industry that is relevant to the two problems. Two challenges arising from the literature were introduced and the succeeding literature posi-

tioned single-objective optimisation, specifically metaheuristics, as a possible method for addressing the large industrial-sized problems in the process industry. Literature on the second challenge was focused on the time representation and a discrete-time representation was selected for the study. A discrete-time single non-uniform grid time representation was proposed as a time-based decomposition for the 14-day hourly interval time horizon. The chapter addressed Objective 1 and 2.

Chapter 3 started with literature on single-objective optimisation to partially fulfil Objective 3. To complete Objective 4, metaheuristic algorithms were selected and a detailed review was presented on the selected algorithms. A brief introduction to multi-objective optimisation followed to complete Objective 3 with a review of the selected algorithm to achieve Objective 5. A review of hybrid metaheuristics literature with a specific focus on classification, grammar, parallel metaheuristics and machine learning concluded the chapter.

The knowledge gained from **Chapter 2** and **Chapter 3** enabled the model construction and implementation of the algorithms. This was documented in **Chapter 4** to achieve Objective 6 and 7. The chapter started with a list of the algorithms that were developed for this study, followed by an explanation of the variables needed. A novel encoding scheme was proposed for the hourly interval problem. The objective functions were presented before describing the three solution evaluation processes; namely, the daily interval evaluation process, the hourly interval evaluation process and the combined evaluation process. The evaluation process contained a balancing period that contributed uniquely to the successful implementation of the algorithms. A detailed description was then given of the parallel model implementations and hybrid metaheuristic implementations.

Chapter 5 satisfied Objective 8 and 9 by presenting a DSS design and implementation. First, the DSS architecture was presented and this was followed by a discussion of the components and implementation of a model-driven DSS.

The verification and evaluation of the DSS were presented in **Chapter 6** to achieve Objective 10. This entailed performing unit and integration testing on the code and achieving face-validation by interviewing subject matter experts. An experiment was conducted based on a recommendation from the subject matter experts (SMEs) and the data supports the change from decimal to integers for the daily interval but not for the hourly interval.

Experiments were conducted on two scenarios to fulfil Objective 11. The results and an analysis of them were presented in **Chapter 6** in partial fulfilment of Objective 12. The analysis and synthesis of the results included the performance of the algorithms using profit and run time as performance measures. The results of the non-dominated sorting genetic algorithm II (NSGA-II) were presented followed by an

analysis of parameters used in the study. Finally, a discussion of parameter tuning to identify the best performing parameter combinations followed. Future work on this problem is presented in Section 7.3 in further fulfilment of Objective 12.

To conclude, the objectives that were formulated to achieve the research task were fulfilled.

7.2 Contributions

The contributions that emanated from this study are, on a macro level, a novel model-based decision support system (DSS) that was designed and documented using a high-level architecture. The architecture provides the flexibility to include different algorithms, optimisers and solvers. The architecture can be scaled to accommodate large deployments. A demonstrator of the model was developed, implemented and tested with subject matter experts.

A metaheuristics approach was proposed and implemented to schedule the feedstock distribution for a chemical value chain. This is, to the best of the researcher's knowledge, the first implementation of such an approach. Additionally, 16 single-objective algorithms over two time intervals was compared on different scenarios.

On a micro level,

- the two problems representing two planning time horizons identified in the study can be integrated and the constructed DSS provides near-optimal solutions,
- a novel step encoding (subsection 4.2.1.2) was proposed that enabled the algorithms to be applied on an hourly interval problem,
- a balancing period was proposed as part of the solution evaluation process to balance the network at each time interval, and
- parameter tuning identified the preferred parameter combinations for this problem.

The single-objective optimisation problem in this study was expanded to the multi-objective optimisation (MOO) domain and formulated as a bi-objective optimisation problem. The NSGA-II delivered satisfactory results and an experiment provided the confidence intervals for the hyper area performance difference indicator for this problem.

The work in this study indicates that it is possible to use metaheuristics for this class of problem as an alternative to the methods used currently in the industry. The

study found the GA to be a good base algorithm. A hybrid GA can increase the solution quality and parallel execution can shorten the execution time.

The summary of the study and conclusions have led to opportunities for future work that will be discussed next.

7.3 Future work

The work done in the study provides several opportunities for future work.

The following suggestions to the single-objective algorithms can be considered:

- The hourly interval horizon can be extended from 14 days to 90 days. The hourly solution can be aggregated to provide the solution for the daily interval.
- Implement other evolutionary algorithms (EAs) on the problem such as ant colony optimisation (ACO) and particle swarm optimisation (PSO).

The following suggestions to the bi-objective algorithm can lead to performance improvements. The following can be considered:

- Investigation of parameter tuning of the NSGA-II will improve the performance.
- Implement strength Pareto evolutionary algorithm 2 (SPEA2) on the problem, as noted by Zitzler *et al.* (2001), SPEA2 could have a better Pareto set solution distribution compared with NSGA-II.

Additional aspects can be included to increase the detail of the decision-making:

- Warm start-up times are significantly shorter than cold start-up times. Keeping a plant warm comes at an hourly cost. This trade-off increases decision complexity.
- Products are produced at different efficiencies. Including the product being produced can increase the accuracy of the expected output.

The following future work is suggested on a macro level:

- Daily production schedules for the final plants in the value chain are based on the feedstock received to meet customers demand. The plants produce multiple products/grades that have sequence-dependent changeovers between different grades and changeovers produce products that do not meet the product specification. This is a well-researched area that can be incorporated into the problem

in this study as illustrated in Figure 7.1. Since this showed that it is possible to integrate the first two problems, future work can investigate if all three problems can be integrated.

- The short run time that is possible with neural networks remains an attractive advantage. In supervised learning, the following can be considered:
  - Use a metaheuristic to find the best design and configuration for the multilayer perceptron (MLP) neural network.
  - Replace the MLP with a long short-term memory (LSTM) neural network or graph neural network (GNN).
  - Alternatively, in the reinforcement learning (RL) field, a multi-agent reinforcement learning (MARL) approach can be considered where each decision variable is represented by an agent.

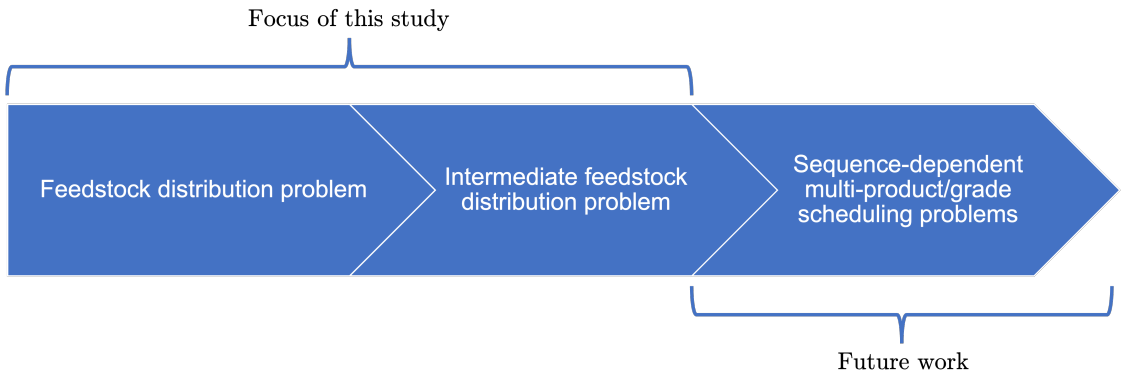


Figure 7.1: Three problems proposed as future work

7.4 Chapter summary

This chapter concludes the research study in which it was shown that an integrated decision support system can be used to provide decision support for hourly and daily interval decision-making on the Sasol C_2 value chain. The chapter provided a concise summary of the research study with contributions that emanated from the study and provided several opportunities for future work.

References

Derya Eren Akyol and G Mirac Bayhan. A review on evolution of production scheduling with neural networks. *Computers & Industrial Engineering*, 53(1):95–122, August 2007. ISSN 03608352. doi: 10.1016/j.cie.2007.04.006. URL https://linkinghub.elsevier.com/retrieve/pii/S0360835207000666. 17, 48, 49

Charles Audet, Jean Bigeon, Dominique Cartier, Sébastien Le Digabel, and Ludovic Salomon. Performance indicators in multiobjective optimization. *European Journal of Operational Research*, 292(2):397–422, July 2021. ISSN 03772217. doi: 10.1016/j.ejor.2020.11.016. URL https://linkinghub.elsevier.com/retrieve/pii/S0377221720309620. 109

James E Baker. *Reducing bias and inefficiency in the selection algorithm.* L. Erlbaum Associates Inc., Cambridge, Massachusetts, USA, 1987. ISBN 0805801588. doi: 10.5555/42512.42515. URL https://dl.acm.org/doi/10.5555/42512.42515. 37

Philipp Baumann and Norbert Trautmann. A hybrid method for large-scale short-term scheduling of make-and-pack production processes. *European Journal of Operational Research*, 236(2):718–735, July 2014. ISSN 03772217. doi: 10.1016/j.ejor.2013.12.040. URL https://linkinghub.elsevier.com/retrieve/pii/S0377221713010242. 17

Peter C Bell. A Decoupling Inventory Problem with Storage Capacity Constraints. *Operations Research*, 28(3-part-i):476–488, June 1980. ISSN 0030-364X. doi: 10.1287/opre.28.3.476. URL http://pubsonline.informs.org/doi/abs/10.1287/opre.28.3.476. 14

Peter C Bell. The effect of restricted access to intermediate product inventory in two-stage production lines. *International Journal of Production Research*, 21(1):75–85, January 1983. ISSN 0020-7543. doi: 10.1080/00207548308942338. URL http://www.tandfonline.com/doi/abs/10.1080/00207548308942338. 14, 28

Peter C Bell, Arshad A Taseen, and Paul F Kirkpatrick. Visual interactive simulation modeling in a decision support role. *Computers & Operations Research*, 17(5):447–456, January 1990. ISSN 03050548. doi: 10.1016/0305-0548(90)90049-D. URL https://linkinghub.elsevier.com/retrieve/pii/030505489090049D. 14

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, April 2021. ISSN 03772217. doi: 10.1016/j.ejor.2020.07.063. URL https://linkinghub.elsevier.com/retrieve/pii/S0377221720306895. 130

Bloomberg Finance L.P. USD to ZAR Exchange Rate - Bloomberg Markets, 2020. URL https://www.bloomberg.com/quote/USDZAR:CUR. 155

Christian Blum, Andrea Roli, and Enrique Alba. *An Introduction to Metaheuristic Techniques*, chapter 1, pages 1–42. John Wiley & Sons, Ltd, 2005a. ISBN 9780471678069. doi: 10.1002/0471739383.ch1. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/0471739383.ch1. 44

Christian Blum, Maria Jose Belsa Aguilera, Andrea Roli, and Michael Sampels. *Hybrid Metaheuristics*, volume 114 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-78294-0. doi: 10.1007/978-3-540-78295-7. URL http://link.springer.com/10.1007/978-3-540-78295-7. 30, 43

Stephan Blum, Romanas Puisa, and Marc Wintermantel. Adaptive Mutation Strategies for Evolutionary Algorithms. *2nd Weimar Optimization and Stochastic Days*, pages 1–13, 2005b. URL https://www.dynardo.de/fileadmin/Material{_}Dynardo/WOST/Paper/wost2.0/AdaptiveMutation.pdf. 35

Jason Brownlee. Clever Algorithms: Nature-Inspired Programming Recipes. In *Climate Change 2013 - The Physical Science Basis*, page 436. Lulu.com, 1st edition, 2011. ISBN 978-1-4467-8506-5. URL http://www.cleveralgorithms.com. 30

Braulio Brunaud, Hector D Perez, Satyajith Amaran, Scott Bury, John Wassick, and Ignacio E Grossmann. Batch scheduling with quality-based changeovers. *Computers & Chemical Engineering*, 132:106617, January 2020. ISSN 00981354. doi: 10.1016/j.compchemeng.2019.106617. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135419300560. 16

Frada Burstein and Clyde W Holsapple. *Handbook on Decision Support Systems 1*. Number 1, Basic themes in Handbook on Decision Support Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-48712-8. doi: 10.1007/978-3-540-48713-5. URL http://link.springer.com/10.1007/978-3-540-48713-5. 74

Jenna Carr. An Introduction to Genetic Algorithms. Technical report, Whitman College, WA, 2014. URL https://www.whitman.edu/Documents/Academics/Mathematics/2014/carrjk.pdf. 35

Pedro M Castro, Ignacio E Grossmann, and Qi Zhang. Expanding scope and computational challenges in process scheduling. *Computers & Chemical Engineering*, 114: 14–42, June 2018. ISSN 00981354. doi: 10.1016/j.compchemeng.2018.01.020. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135418300449. 12, 14, 16, 23

V Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1): 41–51, January 1985. ISSN 0022-3239. doi: 10.1007/BF00940812. URL http://link.springer.com/10.1007/BF00940812. 32

David Gideon Conradie. *Scheduling coal handling processing using metaheuristics*. PhD thesis, University of Pretoria, 2007. URL https://repository.up.ac.za/bitstream/handle/2263/24046/dissertation.pdf?sequence=1. 2

Pedro J Copado-Méndez, Gonzalo Guillén-Gosálbez, and Laureano Jiménez. Rigorous computational methods for dimensionality reduction in multi-objective optimization. In *Computer Aided Chemical Engineering*, volume 30, pages 1292–1296. Elsevier, 2012. ISBN 1570-7946. doi: 10.1016/B978-0-444-59520-1.50117-2. URL https://linkinghub.elsevier.com/retrieve/pii/B9780444595201501172. 39

Carlos Cotta-Porras. A Study of Hybridisation Techniques and Their Application to the Design of Evolutionary Algorithms. *AI Communications*, 11(3,4):223–224, 1998. ISSN 0921-7126. doi: 10.5555/1216122.1216125. URL https://linkinghub.elsevier.com/retrieve/pii/S0920996404001860. 44

CSIMarket. Chemicals - Plastics & Rubber Industry Profitability, 2020. URL https://csimarket.com/Industry/industry{_}Profitability{_}Ratios.php?ind=102. 155

Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Archives of Computational Methods in Engineering*, 27(4):1071–1092, September 2020. ISSN 1134-3060. doi: 10.1007/s11831-019-09344-w. URL http://link.springer.com/10.1007/s11831-019-09344-w. 47

Kenneth De Jong. Parameter Setting in EAs: a 30 Year Perspective. In *Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence*, volume 54, pages 1–18. Springer, Berlin, Heidelberg, 2007. ISBN 978-3-540-69432-8. doi: 10.1007/978-3-540-69432-8_1. URL http://link.springer.com/10.1007/978-3-540-69432-8{_}1. 35

Kenneth Alan De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, USA, 1975. URL https://hdl.handle.net/2027.42/4507. 35

Kalyanmoy Deb and Himanshu Jain. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, August 2014. ISSN 1089-778X. doi: 10.1109/TEVC.2013.2281535. URL http://ieeexplore.ieee.org/document/6600851/. 40

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002. ISSN 1089778X. doi: 10.1109/4235.996017. URL http://ieeexplore.ieee.org/document/996017/. 39, 40, 41

Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, March 2011. ISSN 22106502. doi: 10.1016/j.swevo.2011.02.002. URL https://linkinghub.elsevier.com/retrieve/pii/S2210650211000034. 116, 127

Pedro A Diaz-Gomez and Df Hougen. Initial Population for Genetic Algorithms: A Metric Approach. *Proceedings of the 2007 International Conference on Genetic and Evolutionary Methods*, pages 43–49, 2007. URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.7571{&}rep=rep1{&}type=pdf. 34

Thomas G Dietterich. Machine Learning for Sequential Data: A Review. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 2396, pages 15–30. Springer, Berlin, Heidelberg, 2002. ISBN 3540440119. doi: 10.1007/3-540-70659-3_2. URL http://link.springer.com/10.1007/3-540-70659-3{_}2. 101

WA Du Plessis. *Strategic repositioning of Safripol in the South African polymer industry*. PhD thesis, North-West University, South Africa, 2010. URL https://dspace.nwu.ac.za/bitstream/handle/10394/5566/DuPlessis{_}WA.pdf?sequence=2. 5

Fabian Dunke and Stefan Nickel. Neural networks for the metamodeling of simulation models with online decision making. *Simulation Modelling Practice and Theory*, 99: 102016, February 2020. ISSN 1569190X. doi: 10.1016/j.simpat.2019.102016. URL https://linkinghub.elsevier.com/retrieve/pii/S1569190X19301479. 49

Tarek A El-Milhoub, Adrian A Hopgood, Lars Nolle, and Alan Battersby. Hybrid Genetic Algorithms: A Review. *Engineering Letters*, 13(3):124–137, December 2006. ISSN 00165107. URL http://hdl.handle.net/2086/1173. 34

T Elmihoub, A A Hopgood, L Nolle, and A Battersby. Performance of Hybrid Genetic Algorithms Incorporating Local Search. *Proceedings of 18th European Simulation Multiconference*, pages 154–160, 2004. URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.6860{&}rep=rep1{&}type=pdf. 34, 35

Christodoulos A Floudas and Xiaoxia Lin. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering*, 28(11):2109–2129, October 2004. ISSN 00981354. doi: 10.1016/j.compchemeng.2004.05.002. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135404001401. 20

Fred Glover and Manuel Laguna. *Tabu Search Background*, pages 1–24. Springer US, Boston, MA, 1997a. ISBN 978-1-4615-6089-0. doi: 10.1007/978-1-4615-6089-0_1. URL http://link.springer.com/10.1007/978-1-4615-6089-0{_}1. 25, 26, 31, 32, 33

Fred Glover and Manuel Laguna. *Tabu Search*. Springer US, Boston, MA, 1997b. ISBN 978-0-7923-8187-7. doi: 10.1007/978-1-4615-6089-0. URL http://link.springer.com/10.1007/978-1-4615-6089-0. 31

Fred Glover and Manuel Laguna. Tabu Search. In *Handbook of Combinatorial Optimization*, pages 3261–3362. Springer New York, New York, NY, 2013. doi: 10.1007/978-1-4419-7997-1_17. URL http://leeds-faculty.colorado.edu/glover/449-TabuSearchinAnalyticsandComputationalScience.pdf. 31

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN 9780262035613. URL http://www.deeplearningbook.org. 46, 47

Google Maps. Distance between Sasolburg and Secunda, 2016. URL https://www.google.co.za/maps/dir/-26.820763,+27.838793/-26.52995,29.174134/. 3

Stanley Gotshall and Bart Rylander. Optimal population size and the genetic algorithm. In *Proceedings On Genetic And Evolutionary Computation Conference*, pages 1–5, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.2431{&}rep=rep1{&}type=pdf. 34, 35

Stephen C Graves. A Review of Production Scheduling. *Operations Research*, 29(4): 646, 1981. ISSN 0030364X. URL https://www.jstor.org/stable/170383. 12, 13, 17

Michael Greenberg. Market Update - September 21st, 2020 - theplasticsexchange.com, 2020. URL https://www.theplasticsexchange.com/research/weeklyreview.aspx. 155

Jacomine Grobler, Andries P Engelbrecht, Graham Kendall, and V S S Yadavalli. Alternative hyper-heuristic strategies for multi-method global optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, July 2010. ISBN 978-1-4244-6909-3. doi: 10.1109/CEC.2010.5585980. URL http://ieeexplore.ieee.org/document/5585980/. 35

Ignacio Grossmann. Enterprise-wide optimization: A new frontier in process systems engineering. *AIChE Journal*, 51(7):1846–1857, July 2005. ISSN 0001-1541. doi: 10.1002/aic.10617. URL http://doi.wiley.com/10.1002/aic.10617. 16

Ignacio E Grossmann. Advances in mathematical programming models for enterprise-wide optimization. *Computers & Chemical Engineering*, 47:2–18, December 2012. ISSN 00981354. doi: 10.1016/j.compchemeng.2012.06.038. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135412002220. 16, 21

Ignacio E Grossmann and Kevin C Furman. Challenges in Enterprise Wide Optimization for the Process Industries. In *Springer Optimization and Its Applications*, volume 30, pages 3–59. Springer International Publishing, 2009.

doi: 10.1007/978-0-387-88617-6_1. URL http://link.springer.com/10.1007/978-0-387-88617-6{_}1. 16

Gonzalo Guillén-Gosálbez and Ignacio E Grossmann. A global optimization strategy for the environmentally conscious design of chemical supply chains under uncertainty. *Submitted to Comput. Chem. Eng*, 2008. URL https://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1{&}type=pdf{&}doi=10.1.1.206.2457. 16

Pierre Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Congress on numerical methods in combinatorial optimization, Capri, Italy*, pages 70–145, 1986. 31

Pierre Hansen, Nenad Mladenović, Jack Brimberg, José A. Moreno Pérez, Michel Gendreau, and Jean-Yves Potvin. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer US, Boston, MA, 2010. ISBN 978-1-4419-1663-1. doi: 10.1007/978-1-4419-1665-5. URL http://link.springer.com/10.1007/978-1-4419-1665-5. 27, 34, 35, 36

George Harik, Erick Cantú-Paz, David E Goldberg, and Brad L Miller. The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations. *Evolutionary Computation*, 7(3):231–253, September 1999. ISSN 1063-6560. doi: 10.1162/evco.1999.7.3.231. URL https://direct.mit.edu/evco/article/7/3/231-253/853. 35

Iiro Harjunkoski, Christos T Maravelias, Peter Bongers, Pedro M Castro, Sebastian Engell, Ignacio E Grossmann, John Hooker, Carlos Méndez, Guido Sand, and John Wassick. Scope for industrial applications of production scheduling models and solution methods. *Computers & Chemical Engineering*, 62:161–193, March 2014. ISSN 00981354. doi: 10.1016/j.compchemeng.2013.12.001. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135413003682. 14, 15, 21, 101

Randy L Haupt and Sue Ellen Haupt. *Practical Genetic Algorithms*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2nd edition, May 2003. ISBN 0471455652. doi: 10.1002/0471671746. URL http://doi.wiley.com/10.1002/0471671746. 35

Yaohua He and Chi-Wai Hui. Genetic algorithm for large-size multi-stage batch plant scheduling. *Chemical Engineering Science*, 62(5):1504–1523, March 2007. ISSN 00092509. doi: 10.1016/j.ces.2006.11.049. URL https://linkinghub.elsevier.com/retrieve/pii/S0009250906007573. 17

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://direct.mit.edu/neco/article/9/8/1735-1780/6109. 48

Christian D Hubbs, Can Li, Nikolaos V Sahinidis, Ignacio E Grossmann, and John M Wassick. A deep reinforcement learning approach for chemical production scheduling. *Computers & Chemical Engineering*, 141:106982, October 2020. ISSN 00981354. doi: 10.1016/j.compchemeng.2020.106982. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135420301599. 18

Zora Neale Hurston. Technology in Motion, 2013. URL http://www.sasol.com/sites/sasol/files/content/files/Sasol{_}Fuels{_}technology{_}brochure{_}V18{_}0.pdf. 2

ISO. ISO/PAS 19450: 2015 - Automation systems and integration-Object-Process Methodology, 2015. URL https://www.iso.org/standard/62274.html. 75

Jennifer R Jackson, Jeanna Hofmann, John Wassick, and Ignacio E Grossmann. A nonlinear multiperiod process optimization model for production planning in multi-plant facilities. Technical report, Carnegie Mellon University, PA, 2003. URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.19.4147{&}rep=rep1{&}type=pdf. 15, 59

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castañeda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, May 2019. ISSN 0036-8075. doi: 10.1126/science.aau6249. URL https://www.sciencemag.org/lookup/doi/10.1126/science.aau6249. 47

Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, January 2005. ISSN 1432-7643. doi: 10.1007/s00500-003-0328-5. URL http://link.springer.com/10.1007/s00500-003-0328-5. 49

Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen. Data-Driven Evolutionary Optimization: An Overview and Case Studies. *IEEE Transactions on Evolutionary Computation*, 23(3):442–458, June 2019. ISSN 1089-778X. doi: 10.1109/TEVC.2018.2869001. URL https://ieeexplore.ieee.org/document/8456559/. 49

Jinghui Zhong, Xiaomin Hu, Jun Zhang, and Min Gu. Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 2, pages 1115–1121. IEEE, 2005. ISBN 0-7695-2504-0. doi: 10.1109/CIMCA.2005.1631619. URL http://ieeexplore.ieee.org/document/1631619/. 37

Josef Kallrath. Planning and scheduling in the process industry. *OR Spectrum*, 24 (3):219–250, August 2002. ISSN 0171-6468. doi: 10.1007/s00291-002-0101-7. URL http://link.springer.com/10.1007/s00291-002-0101-7. 14, 30, 52

M Kannegiesser and H-O Günther. An integrated optimization model for managing the global value chain of a chemical commodities manufacturer. *Journal of the Operational Research Society*, 62(4):711–721, April 2011. ISSN 0160-5682. doi: 10.1057/jors.2010.18. URL https://www.tandfonline.com/doi/full/10.1057/jors.2010.18. 2

Rahul B Kasat, Ajay K Ray, and Santosh K Gupta. Applications of Genetic Algorithm in Polymer Science and Engineering. *Materials and Manufacturing Processes*, 18(3):523–532, January 2003. ISSN 1042-6914. doi: 10.1081/AMP-120022026. URL http://www.tandfonline.com/doi/abs/10.1081/AMP-120022026. 33

Jong Suk Kim and Thomas F Edgar. Optimal scheduling of combined heat and power plants using mixed-integer nonlinear programming. *Energy*, 77:675–690, December 2014. ISSN 03605442. doi: 10.1016/j.energy.2014.09.062. URL https://linkinghub.elsevier.com/retrieve/pii/S0360544214011190. 61

S Kirkpatrick, C D Gelatt, and M P Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May 1983. ISSN 0036-8075. doi: 10.1126/science.220.4598.671. URL https://www.sciencemag.org/lookup/doi/10.1126/science.220.4598.671. 32

Averill M Law. *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, New York, NY, USA, 5th edition, 2015. ISBN 9780073401324. URL http://public.eblib.com/choice/PublicFullRecord.aspx?p=6327699. 124

In Lee and Yong Jae Shin. Machine learning for enterprises: Applications, algorithm selection, and challenges. *Business Horizons*, 63(2):157–170, March 2020. ISSN 00076813. doi: 10.1016/j.bushor.2019.10.005. URL https://linkinghub.elsevier.com/retrieve/pii/S0007681319301521. 46

Jay H Lee, Joohyun Shin, and Matthew J Realff. Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, 114:111–121, June 2018. ISSN 00981354. doi: 10.1016/j.compchemeng.2017.10.008. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135417303538. 46

David Levine. Commentary - Genetic Algorithms: A Practitioner's View. *INFORMS Journal on Computing*, 9(3):256–259, August 1997. ISSN 1091-9856. doi: 10.1287/ijoc.9.3.256. URL http://pubsonline.informs.org/doi/abs/10.1287/ijoc.9.3.256. 35

B Lin and D C Miller. Tabu search algorithm for chemical process optimization. *Computers & Chemical Engineering*, 28(11):2287–2306, October 2004. ISSN 00981354. doi: 10.1016/j.compchemeng.2004.04.007. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135404001231. 18, 27, 32

Ying Liu, Haibo Dong, Niels Lohse, and Sanja Petrovic. A multi-objective genetic algorithm for optimisation of energy consumption and shop floor production performance. *International Journal of Production Economics*, 179:259–272, September 2016. ISSN 09255273. doi: 10.1016/j.ijpe.2016.06.019. URL https://linkinghub.elsevier.com/retrieve/pii/S092552731630127X. 18

Fernando G Lobo and David E Goldberg. The parameter-less genetic algorithm in practice. *Information Sciences*, 167(1-4):217–232, December 2004. ISSN 00200255. doi: 10.1016/j.ins.2003.03.029. URL https://linkinghub.elsevier.com/retrieve/pii/S0020025504000441. 34, 35

Michael Lones. Sean Luke: essentials of metaheuristics. *Genetic Programming and Evolvable Machines*, 12(3):333–334, September 2011. ISSN 1389-2576. doi: 10.1007/s10710-011-9139-0. URL http://link.springer.com/10.1007/s10710-011-9139-0. 35, 37

Looxix. Scheme of a neuron, 2003. URL https://commons.wikimedia.org/wiki/File:Neurone.png. 47

Heikki Maaranen, Kaisa Miettinen, and Antti Penttinen. On initial populations of a genetic algorithm for continuous optimization problems. *Journal of Global Optimization*, 37(3):405–436, January 2007. ISSN 0925-5001. doi: 10.1007/s10898-006-9056-6. URL http://link.springer.com/10.1007/s10898-006-9056-6. 34

B L Maccarthy and Jiyin Liu. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, 31(1):59–79, January 1993. ISSN 0020-7543. doi: 10.1080/00207549308956713. URL http://www.tandfonline.com/doi/abs/10.1080/00207549308956713. 14

Sandeep U Mane and M R Narasinga Rao. Many-objective optimization: Problems and evolutionary algorithms - a short review, 2017. ISSN 09739769. URL https://www.ripublication.com/ijaer17/ijaerv12n20{_}72.pdf. 39

Christos T Maravelias. General framework and modeling approach classification for chemical production scheduling. *AIChE Journal*, 58(6):1812–1828, June 2012. ISSN 00011541. doi: 10.1002/aic.13801. URL https://onlinelibrary.wiley.com/doi/10.1002/aic.13801. 14

Pablo A Marchetti, Ignacio E Grossmann, Wiley Bucey, and Rita A Majewski. A Multiproduct Feedstock Optimization Model for Polymer Production. In *Computer Aided Chemical Engineering*, volume 32, pages 583–588. Elsevier B.V, 2013. ISBN 1570-7946. doi: 10.1016/B978-0-444-63234-0.50098-1. URL https://linkinghub.elsevier.com/retrieve/pii/B9780444632340500981. 15, 16

Pablo A Marchetti, Miguel A Zamarripa, Juan A Reyes-Labarta, Ignacio E Grossmann, Wiley Bucey, and Rita A Majewski. Optimal planning and feedstock-mix selection for multiproduct polymer production. *Computers & Chemical Engineering*, 95:182–201, December 2016. ISSN 00981354. doi: 10.1016/j.compchemeng.2016.09.002. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135416302800. 15, 16

Yailen Martínez Jiménez. *A Generic Multi-Agent Reinforcement Learning Approach for Scheduling Problems*. PhD thesis, Vrije Universiteit Brussel, 2012. URL https://ai.vub.ac.be/wp-content/uploads/2019/12/

A-Generic-Multi-Agent-Reinforcement-Learning-Approach-for-Scheduling.-Problems.pdf. 18

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, December 1943. ISSN 0007-4985. doi: 10.1007/BF02478259. URL http://link.springer.com/10.1007/BF02478259. 47

Melanie Mitchell. Genetic algorithms: An overview. *Complexity*, 1(1):31–39, September 1995. ISSN 10762787. doi: 10.1002/cplx.6130010108. URL https://onlinelibrary.wiley.com/doi/10.1002/cplx.6130010108. 33

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. Technical report, Deepmind, December 2013. URL https://dblp.org/rec/journals/corr/MnihKSGAWR13. 47

Pablo Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical report, California Institute ofTechnology, 1989. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.9474{&}rep=rep1{&}type=pdf. 42

Wiem Mouelhi-Chibani and Henri Pierreval. Training a neural network to select dispatching rules in real time. *Computers & Industrial Engineering*, 58(2):249–256, March 2010. ISSN 03608352. doi: 10.1016/j.cie.2009.03.008. URL https://linkinghub.elsevier.com/retrieve/pii/S0360835209000953. 49

R F H Musier and L B Evans. An approximate method for the production scheduling of industrial batch processes with parallel units. *Computers & Chemical Engineering*, 13(1-2):229–238, January 1989. ISSN 00981354. doi: 10.1016/0098-1354(89)89020-9. URL https://linkinghub.elsevier.com/retrieve/pii/0098135489890209. 14, 17

Gerrit Stephanus Nel. *A hyperheuristic approach towards the training of artificial neural networks*. PhD thesis, Stellenbosch University, 2021. URL http://hdl.handle.net/10019.1/109792. 116

Peter Bjorn Nemenyi. *Distribution-Free Multiple Comparisons*. PhD thesis, Princeton University, 1963. URL https://catalog.princeton.edu/catalog/9920813653506421. 122

Duyquang Nguyen and Miguel Bagajewicz. Optimization of Preventive Maintenance in Chemical Process Plants. *Industrial & Engineering Chemistry Research*, 49(9): 4329–4339, May 2010. ISSN 0888-5885. doi: 10.1021/ie901433b. URL https://pubs.acs.org/doi/10.1021/ie901433b. 27, 36

Ivana Nižetić, Krešimir Fertalj, and Boris Milašinović. An Overview of Decision Support System Concepts. *18th International Conference on Information and Intelligent Systems*, 2007. URL https://www.bib.irb.hr/305146. 74

Nasimul Noman, Danushaka Bollegala, and Hitoshi Iba. Differential evolution with self adaptive local search. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*, page 1099, New York, New York, USA, 2011. ACM Press. ISBN 9781450305570. doi: 10.1145/2001576.2001725. URL http://doi.acm.org/10.1145/2001576.2001725http://portal.acm.org/citation.cfm?doid=2001576.2001725. 42

Tatsuya Okabe, Yaochu Jin, and Bernhard Sendhoff. A critical survey of performance indices for multi-objective optimisation. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 2, pages 878–885. IEEE, 2003. ISBN 0-7803-7804-0. doi: 10.1109/CEC.2003.1299759. URL http://ieeexplore.ieee.org/document/1299759/. 109

Oxford English Dictionary. Definition: Decision support system, 2011. URL http://www.oed.com.ez.sun.ac.za/. 7

Joseph F Pekny. Algorithm architectures to support large-scale process systems engineering applications involving combinatorics, uncertainty, and risk management. *Computers & Chemical Engineering*, 26(2):239–267, February 2002. ISSN 00981354. doi: 10.1016/S0098-1354(01)00744-X. URL https://linkinghub.elsevier.com/retrieve/pii/S009813540100744X. 38

Humyun Fuad Rahman, Ruhul Sarker, and Daryl Essam. A genetic algorithm for permutation flow shop scheduling under make to stock production system. *Computers and Industrial Engineering*, 90:12–24, 2015. ISSN 03608352. doi: 10.1016/j.cie.2015.08.006. URL http://dx.doi.org/10.1016/j.cie.2015.08.006. 27

Günther R Raidl. A Unified View on Hybrid Metaheuristics. In *Hybrid Metaheuristics*, pages 1–12. Springer, Berlin, Heidelberg, 2006. ISBN 9783540463849. doi: 10.1007/11890584_1. URL http://link.springer.com/10.1007/11890584{_}1. 44

Günther R Raidl and Jakob Puchinger. Combining (Integer) Linear Programming Techniques and Metaheuristics for Combinatorial Optimization. In *Computers & Operations Research*, volume 64, pages 31–62. Springer Berlin Heidelberg, December 2008. doi: 10.1007/978-3-540-78295-7_2. URL http://link.springer.com/10.1007/978-3-540-78295-7{_}2. 43, 44

Manojkumar Ramteke and Rajagopalan Srinivasan. Novel genetic algorithm for short-term scheduling of sequence dependent changeovers in multiproduct polymer plants. *Computers and Chemical Engineering*, 35(12):2945–2959, 2011. ISSN 00981354. doi: 10.1016/j.compchemeng.2011.05.002. 18

A Reisman, A Kumar, and J Motwani. Flowshop scheduling/sequencing research: a statistical review of the literature, 1952-1994. *Engineering Management, IEEE Transactions on*, 44(3):316–329, 1997. ISSN 0018-9391. 14

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 1939-1471. doi: 10.1037/h0042519. URL http://doi.apa.org/getdoi.cfm?doi=10.1037/h0042519. 47

Bill Roungas, Sebastiaan A Meijer, and Alexander Verbraeck. A framework for optimizing simulation model validation & verification. *International Journal On Advances in Systems and Measurements. In Press*, 11, 2018. URL http://www.iariajournals.org/systems{_}and{_}measurements/2018,. 124

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. ISSN 0028-0836. doi: 10.1038/323533a0. URL http://www.nature.com/articles/323533a0. 47

Ihsan Sabuncuoglu and Souheyl Touhami. Simulation metamodelling with neural networks: An experimental investigation. *International Journal of Production Research*, 40(11):2483–2505, January 2002. ISSN 0020-7543. doi: 10.1080/00207540210135596. URL http://www.tandfonline.com/doi/abs/10.1080/00207540210135596. 49

A Sadegheih. Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance. *Applied Mathematical Modelling*, 30(2):147–154, February 2006. ISSN 0307904X. doi: 10.1016/j.apm.2005.03.017. URL https://linkinghub.elsevier.com/retrieve/pii/S0307904X05000521. 17

Sasol. Sasol Segment overview, 2014a. URL https://www.sasol.com/sites/default/files/content/files/SegmentOverviewDocument{_}18Feb1015.pdf. 2

Sasol. Redacted AEL Polymers Sasolburg, 2014b. 4, 5

Sasol. Redacted AEL Polymers Secunda, 2015a. 4, 5

Sasol. Redacted AEL Synfuels Secunda, 2015b. 4

Sasol. Sasol Ltd 20-F Annual Report 2015, 2015c. URL https://www.sasol.com/sites/default/files/financial{_}reports/SasolLtd20-FWebsite{_}v1{_}2{_}0.PDF. 9

Sasol. Sasol Sasol Products - Fuels and Oils, Chemicals and Gas, 2019. URL https://products.sasol.com/pic/products/home/index.html. 2, 4, 5

D Saygin, M K Patel, E Worrell, C Tam, and D J Gielen. Potential of best practice technology to improve energy efficiency in the global chemical and petrochemical sector. *Energy*, 36(9):5779–5790, September 2011. ISSN 03605442. doi: 10.1016/j.energy.2011.05.019. URL https://linkinghub.elsevier.com/retrieve/pii/S0360544211003446. 39

Deger Saygin, Martin K Patel, Cecilia Tam, and Dolf J Gielen. Chemical and Petrochemical Sector. Technical report, IEA, Paris, 2009. URL https://www.iea.org/reports/chemical-and-petrochemical-sector. 61

Haitham Seada and Kalyanmoy Deb. U-NSGA-III: A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives: Proof-of-Principle Results. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9019, pages 34–49. Springer, 2015. ISBN 9783319158914. doi: 10.1007/978-3-319-15892-1_3. URL http://link.springer.com/10.1007/978-3-319-15892-1{_}3. 40

Pratik Shukla and Roberto Iriondo. Main Types of Neural Networks and its Applications, 2020. URL https://medium.com/towards-artificial-intelligence/main-types-of-neural-networks-and-its-applications-tutorial-.734480d7ec8e. 47

E A Silver. An overview of heuristic solution methods. *Journal of the Operational Research Society*, 55(9):936–956, September 2004. ISSN 0160-5682. doi: 10.1057/palgrave.jors.2601758. URL https://www.tandfonline.com/doi/full/10.1057/palgrave.jors.2601758. 25

Kenneth Sörensen, Marc Sevaux, and Fred Glover. A History of Metaheuristics. In *Handbook of Heuristics*, pages 1–18. Springer International Publishing, Cham, 2018. ISBN 0521804701. doi: 10.1007/978-3-319-07153-4_4-1. URL http://arxiv.org/abs/1704.00853http://link.springer.com/10.1007/978-3-319-07153-4{_}4-1. 30

N Srinivas and Kalyanmoy Deb. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, September 1994. ISSN 1063-6560. doi: 10.1162/evco.1994.2.3.221. URL https://direct.mit.edu/evco/article/2/3/221-248/1396. 40

Statistics South Africa. Statistical release: Gross Domestic Product, Fourth Quarter 2015. *Gross domestic product*, P0441(December):1–17, 2015. URL http://www.statssa.gov.za/publications/P0441/P04414thQuarter2015.pdf. 2

Mario Stobbe, Thomas Löhl, Christian Schulz, and Sebastian Engell. Planning and Scheduling in the Process Industry. Technical Report April, University of Dortmund, 2000. 21

Arul Sundaramoorthy and Christos T Maravelias. Computational Study of Network-Based Mixed-Integer Programming Approaches for Chemical Production Scheduling. *Industrial & Engineering Chemistry Research*, 50(9):5023–5040, May 2011. ISSN 0888-5885. doi: 10.1021/ie101419z. URL https://pubs.acs.org/doi/10.1021/ie101419z. 19, 22

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2017. URL https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf. 46, 47

E G Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564, 2002. ISSN 13811231. doi: 10.1023/A:1016540724870. URL https://link.springer.com/article/10.1023/A:1016540724870. 43, 44, 45

E G Talbi. *Metaheuristics: From Design to Implementation*. Numerical Methods & Algorithms. John Wiley & Sons, Inc., May 2009. ISBN 978-0-470-49690-9. URL https://www.wiley.com/en-us/Metaheuristics{%}3A+From+Design+to+Implementation+-p-9780470496909. 20, 25, 26, 27, 30, 32, 33, 35, 36, 39, 42, 43, 49, 60, 109

E G Talbi. A Unified Taxonomy of Hybrid Metaheuristics with Mathematical Programming, Constraint Programming and Machine Learning. In *Hybrid Metaheuristics*, volume 434, pages 3–76. Springer, 2013. doi: 10.1007/978-3-642-30671-6_1. URL http://link.springer.com/10.1007/978-3-642-30671-6{_}1. 44

United States Department of Energy. Ethane Storage and Distribution Hub in the United States. Technical Report November, United States Department of Energy, Washington, 2018. URL https://www.energy.gov/sites/prod/files/2018/12/f58/Nov2018DOEEthaneHubReport.pdf. 2, 4

Sara Velez and Christos T Maravelias. Multiple and nonuniform time grids in discrete-time MIP models for chemical production scheduling. *Computers & Chemical Engineering*, 53:70–85, June 2013. ISSN 00981354. doi: 10.1016/j.compchemeng.2013.01.014. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135413000410. 19, 20

Sara Velez and Christos T Maravelias. Theoretical framework for formulating MIP scheduling models with multiple and non-uniform discrete-time grids. *Computers & Chemical Engineering*, 72:233–254, January 2015. ISSN 00981354. doi: 10.1016/j.compchemeng.2014.03.003. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135414000702. 22

Kefeng Wang, Thomas Löhl, Mario Stobbe, and Sebastian Engell. A genetic algorithm for online-scheduling of a multiproduct polymer batch plant. *Computers & Chemical Engineering*, 24(2-7):393–400, July 2000. ISSN 00981354. doi: 10.1016/S0098-1354(00)00427-0. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135400004270. 17, 34

Zhenwu Wang, Chao Qin, Benting Wan, and William Wei Song. A Comparative Study of Common Nature-Inspired Algorithms for Continuous Function Optimization. *Entropy*, 23(7):874, July 2021. ISSN 1099-4300. doi: 10.3390/e23070874. URL https://www.mdpi.com/1099-4300/23/7/874. 116

John M Wassick. Enterprise-wide optimization in an integrated chemical complex. *Computers & Chemical Engineering*, 33(12):1950–1963, December 2009. ISSN 00981354. doi: 10.1016/j.compchemeng.2009.06.002. URL https://linkinghub.elsevier.com/retrieve/pii/S0098135409001574. 15, 16

Thomas Weise. *Global Optimization Algorithms–Theory and Application*, volume 1. 2009. doi: 10.1.1.64.8184. URL http://www.it-weise.de/projects/book.

pdf{%}5Cnhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.64.8184{&}rep=rep1{&}type=pdf. 25, 28, 31, 32

David P Williamson and David B Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, Cambridge, June 2012. ISBN 9780511921735. doi: 10.1017/CBO9780511921735. URL http://ebooks.cambridge.org/ref/id/CBO9780511921735. 25, 30

Christopher Wilt, Jordan Thayer, and Wheeler Ruml. A comparison of greedy search algorithms. In *Proceedings of the 3rd Annual Symposium on Combinatorial Search, SoCS 2010*, pages 129–136, 2010. ISBN 9781577354819. URL www.aaai.orghttps://www.aaai.org/ocs/index.php/SOCS/SOCS10/paper/viewFile/2101/2515. 38

Paraskevi Th Zacharia, Elias K Xidias, and Nikos A Aspragathos. Task scheduling and motion planning for an industrial manipulator. *Robotics and Computer-Integrated Manufacturing*, 29(6):449–462, December 2013. ISSN 07365845. doi: 10.1016/j.rcim.2013.05.002. URL http://dx.doi.org/10.1016/j.rcim.2013.05.002https://linkinghub.elsevier.com/retrieve/pii/S0736584513000410. 27

Xu Zhang, Hua Zhang, and Jin Yao. Multi-Objective Optimization of Integrated Process Planning and Scheduling Considering Energy Savings. *Energies*, 13(23): 6181, November 2020. ISSN 1996-1073. doi: 10.3390/en13236181. URL https://www.mdpi.com/1996-1073/13/23/6181. 18

Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, June 2000. ISSN 1063-6560. doi: 10.1162/106365600568202. URL https://direct.mit.edu/evco/article/8/2/173-195/868. 39

Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, 2001. doi: 10.1.1.28.7571. URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.7571{&}rep=rep1{&}type=pdf. 39, 135

Danielle Zyngier and Jeffrey D Kelly. UOPSS: A New Paradigm for Modeling Production Planning & Scheduling Systems. *Symposium on Computer Aided Process Engineering*, 17:20, 2012. URL https://booksite.elsevier.com/9780444594310/downloads/ESC.35-UOPSSAnewparadigmformodelingproductionplanning.pdf. 15, 27, 60

Appendix

A Product margins

To protect the confidentiality of proprietary data, the product margins have been obtained from public data. United States of America (USA) spot prices are used in United States Dollar (USD) per pound and are converted to a gross margin in South African Rand (ZAR) per metric ton. Table A.1 lists the margin values converted from the USA spot prices.

Table A.1: Margins inferred from public data

	$/lb [1]	USD/ZAR [2]	lb/kg	R/t	Margin [3]
LLDPE	$0,55	R16,95	0,4536	R20 459	R6 956
LDPE	$0,58	R16,95	0,4536	R21 580	R7 337
Ethylene	$0,24	R16,95	0,4536	R8 968	R3 049
Ethane	$0,08	R16,95	0,4536	R2 915	R991

Notes on the assumptions in Table A.1:

1. Linear low-density polyethylene (LLDPE) and low-density polyethylene (LDPE) can be sold in film and injection mould grades which are sold for different prices. The US spot market provides a low and high price. Using the average spot market price and the average price between film and injection mould grades can indicate the US market price. The plastic exchange provides spot prices in $/lb for LLDPE, LDPE, Ethylene and Ethane (Greenberg, 2020).

2. The exchange rate for United States Dollar (USD) to South African Rands (ZAR) (Bloomberg Finance L.P, 2020).

3. A gross margin of 34% is listed for the second quarter of 2020 in the chemicals, plastic and rubber industry (CSIMarket, 2020).