Simulation optimisation in trailer management

Rijnen, D.J.F.

Award date:
2017

Technische Universiteit
Eindhoven
University of Technology

Department of Industrial Engineering & Innovation Sciences
Information Systems Group

Simulation optimisation in trailer management

Master Thesis

D.J.F. Rijnen BSc

Supervisors:
dr. Y. (Yingqian) Zhang
dr.ir. B.F. (Boudewijn) van Dongen
dr.ir. R.M. (Remco) Dijkman

Company supervisors:
E. (Eric) Schrover
C. (Christophe) De Ridder

1.0.0

Eindhoven, October 2017

Abstract

Simulation optimisation in trailer management
By D.J.F. (Dylan) Rijnen

The trailer flows of a big fast moving consumer goods (FMCG) company were simulated using discrete-event simulation and subsequently optimised using a simulation optimisation model. The simulation optimisation model combined meta-heuristic search (genetic algorithm), with a meta-model filter (feed-forward neural network) to optimise the configuration of the simulation model. This methodology was extended with an ensure probability that overrules the rejection of potential solutions. It was shown that a better performance could be achieved when this probability was added. Furthermore, this research also tested the impact of the configuration of the optimisation model on its effectiveness and found that parameters such as population size, filter threshold and mutation probability can have a large impact on the overall optimisation effectiveness. It was found that smaller population sizes perform better simulation optimisations due to quicker transitions to other generations which reduce the impact of weaker approximation models. Furthermore, having a threshold to allow solutions to be analysed that are slightly worse than the best solution found so far increases overall performance of the algorithm, however having the threshold too high results in costly unnecessary evaluations. A value slightly lower than the mean absolute error of the meta-model showed to have the best performance in this setup. Finally, it was found that the optimisation methodology could significantly reduce the expected costs of the trailer management process by optimising the parameters of the simulation model.

Preface

I welcome you to my thesis that I wrote for the finalisation of my Master studies Business Information Systems at the Eindhoven University of Technology. With this thesis, an end comes to my six-year long student life, which has been the stage of many of the most amazing experiences I have experienced so far. Before getting to the thesis, I would like to take a brief moment to thank all those that supported me along the journey.

First of all, I want to thank my university supervisor Yingqian Zhang. Yingqian, I want to thank you for your helpful feedback and open-mindedness in discussing my thesis. Next, I would like to thank my company supervisors, Eric Schrover and Christophe de Ridder, for their guidance and all the insights they gave me in the logistics processes at ABI. I want to thank them for the opportunity of doing my project in a company with such an interesting and dynamic environment. I would also like to thank Karin, Erik, Bart and all other colleagues I have met along the way, for the interesting discussions we had both on the subject matter of this thesis and outside of it. Aside from this, I want to thank my fellow interns and friends at ABI, which brightened the time I spend in Belgium to perform this research.

During my studies I have met a lot of amazing people, which I want to thank for making my student life amazing. In particular I would like to thank my close friends, with whom I shared the largest part of the journey. I have always been able to count on you guys and I appreciated our many travels, and all the discussions about everything or nothing that we had. Furthermore, I want to thank all the other amazing people that I met along the way, for instance at Industria or at the Honors Horizon program. Last but not least, I want to thank Julie, my amazing girlfriend. I really appreciate your support the last couple of months. You were always there for me and open to discuss with me what I had on my mind.

It is bittersweet to see the end of my time as a student approaching but I am looking very much forward to all the challenges that are coming next! I wish you a pleasant read of my thesis!

Dylan Rijnen
October 2017

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

With complexity in business problems rising, traditional optimisation techniques are becoming decreasingly capable of fully solving business problems. Simulation studies have shown to be an effective method to tackle these problems (Ingalls, 1998; Law et al., 1991). Increased computer performance has allowed the creation of advanced simulation models that closely mimic and allow analysis of the complexity of these business problems (Fu et al., 2005). Traditionally, simulations were mostly used for manually guided analyses of the problem, where the analyst had full control on the simulation configuration to explore the impact of changes. This methodology results in human errors and lack of depth, especially when the underlying behaviour of the business problem is not fully understood. With simulations often taking significant time to build and use, it is beneficial to get the most optimal model configurations in a limited time. For simulations with only a small range of possible configurations, simulation optimisation has been performed by exhaustively running simulations for all parameter combinations. This is however not possible for higher complexity simulations with a lot of different parameter combinations or variables that are continuous rather than deterministic.

Recent research on using simulations with infinite or, finite but many, parameter combinations have focused on the development of simulation optimisation algorithms that combine meta-heuristic search algorithms with function approximation models for fitness approximation. These fitness approximations are subsequently used to replace simulated fitness evaluations, or filter which solutions to simulate and which not. This research will focus on using the fitness approximation as a filter, but also briefly explores a methodology were true fitness evaluation is replaced by the fitness approximation.

By using function approximation models instead of the higher fidelity responses from the simulation model itself, it is possible to optimise models in hours or days that would, using previous optimisation techniques, take several weeks to even months to analyse. These function approximation techniques are also called meta-models, as they effectively model the outputs of the higher fidelity model. In the rest of this report, therefore the term meta-model is used.

The real-world scenario evaluated in this research is focussed on the trailer management process of AB InBev (ABI), the world's largest producer and distributor of beer with over 400 different kinds of beers in more than 50 different countries (AB InBev, 2017). Currently, they face problems with managing their transportation trailer fleet among the different production sites of ABI in Belgium. Complexities in the trailer management process include non-homogeneous stochastic demands differing both during the day and during the week, different types of trailers that are used for different kinds of shipments, the hybrid usage of external and owned trailers and multiple dependent locations managing a shared resource pool with individual needs and preferences. As a result of these complexities, situations occur in which no trailers are available when needed. When no trailers are available this negatively impact the logistic processes.

At this moment fleet sizing and trailer reallocation decisions are done ad-hoc and often when problems have already occurred. An approach is needed to prevent problems related to trailer management. In order to improve the general trailer management process a tactical decision

support tool was suggested. This tactical decision support tool will take the form of a system simulation model which will allow analysis of the potential impact of fleet sizing and other trailer management decisions on the overall process performance. The overall process performance is dependent on the availability of trailers when needed and the ability to store them on the parking (de Ridder, 2017). Subsequently, an optimal configuration of the trailer management process was needed. As an exhaustive search of the process parameters was infeasible due to the time cost of simulations and the size of the solution space, a simulation optimisation algorithm was needed to find optimal configurations.

The scope of this research is the trailer management process of the sites in Belgium. The project focuses on the three largest sites in this region (Leuven, Hoegaarden and Jupille). These sites are connected by the workflow management system (WMS), resulting in sufficient information available to accurately model the trailer flows. The other sites that are not connected by WMS are taken into account by the model as black box external processes due to unavailability of processing data on these depots and sites.

1.1 Research questions

The setup of this research project consists of designing a simulation optimisation algorithm using meta-modelling and heuristic search techniques. Subsequently, this algorithm has been applied, using different configurations, on a real world simulation model. The simulation model was built with a focus on trailer fleet management for ABI. This problem is constrained by costs, site-level resource limitations and service level goals. The performance of the algorithm was evaluated and analysed to gain better insight in the algorithm, its performance and the performance of its individual parts.

The research was structured according to the following research question:
"How can search techniques and meta-models effectively be combined to perform simulation optimisation on a real-world based simulation?"
This question was subject to the following sub-questions:

1. What kind of search techniques and meta-models are most applicable in the field of simulation optimisation?
2. What interfaces should exist between search algorithms, meta-models and simulation models such that simulation optimisation can be performed?
3. How can the effectiveness of simulation optimisation models be measured?
4. What is the impact of respectively the search and function approximation part of the algorithm on total effectiveness?
5. What factors influence the effectiveness of the simulation optimisation algorithm?

The answers of the sub-questions were combined in order to answer the main research question. The company wanted to achieve two goals with this project:

- Clarify the trailer management process by identifying causes of the current problems related to the trailer management process.
- Create an optimal fleet configuration for the amount of required trailers (of the different types (trailer and tanker) and both owned and external) at the different sites that would fulfil the transportation demand taking into account costs, trailer availability and availability of parking places.

To achieve these goals the following steps have been taken:

- Identify the transportation process with a central focus on trailer flows.

- Identify trailer availability and need factors and constraints through qualitative analysis by interviewing managers and process owners.
- Based on the identified transportation process and trailer factors, create model specifications and identify the parameters to take into account for the simulation model.
- Create the simulation model
- Create and configure the simulation optimisation algorithm
- Apply the optimisation algorithm to determine the optimal trailer fleet sizing configuration

1.2 Deliverables

The deliverables of this research project were a simulation model for the trailer management process, a simulation optimisation model and several analyses of the used technique. Lastly, an analysis of the problem domain was done using the created techniques resulting in a set of recommendations for improvements in the trailer management process.

1.3 Scientific relevance

This research aims to contribute to decades worth of simulation optimisation by analysing the different factors used in an applied simulation optimisation technique that combines a meta-heuristic search algorithm with a meta-model approximation technique used to filter which solutions to simulate or not.

By taking a look at individual parameters, the underlying dynamics and design decisions in using these kind of simulation optimisation techniques are further explored. In addition, the original technique of combining a meta-heuristic optimizer with a meta-model filter as described in April et al. (2003) is extended with an additional mechanic, an ensure probability, which forces evaluation of individuals even when having lower approximation scores. This ensure probability showed to improve performance when a low value for this ensure probability was selected and prevents solutions from being rejected indefinitely due to inaccuracies of the meta-model filter.

Furthermore, the research aims to help closing the gap between research and practice in simulation optimisation by applying the designed methodology on a real-world case study instead of toy problems, which have often been used for this area of research. This technique also contributes to the research domain of trailer & asset management using simulation-based optimisation. Even though simulations have been used to improve trailer management before, there is very limited research performed in this domain using simulation optimisation techniques. The proposed optimisation methodology has not yet been used before in this field (to the author's best knowledge), even though it shows promising results. It was found that little research has been done on the actual impact of configuration of the proposed simulation optimisation model before applying it. This research explored how different parameters of the simulation optimisation model influence the optimisation process (taking into account a limited budget for simulation and optimisation).

To summarise, the main contributions of this research are:

- Extension of simulation optimisation methodology using meta-model filter with an ensure probability to force evaluations
- Showed the importance of configuring the optimisation model by measuring the impact of individual optimisation parameters (such as population size, initial training set size, mutation probability and meta-model filter threshold) on overall optimisation effectiveness
- Contributed to trailer & asset management literature by applying a novel approach to a trailer & asset management problem

1.4 Approach & report outline

In order to answer the research question and the different sub questions posed in Section 1.1, the remainder of this research is structured as shown in Figure 1.1.

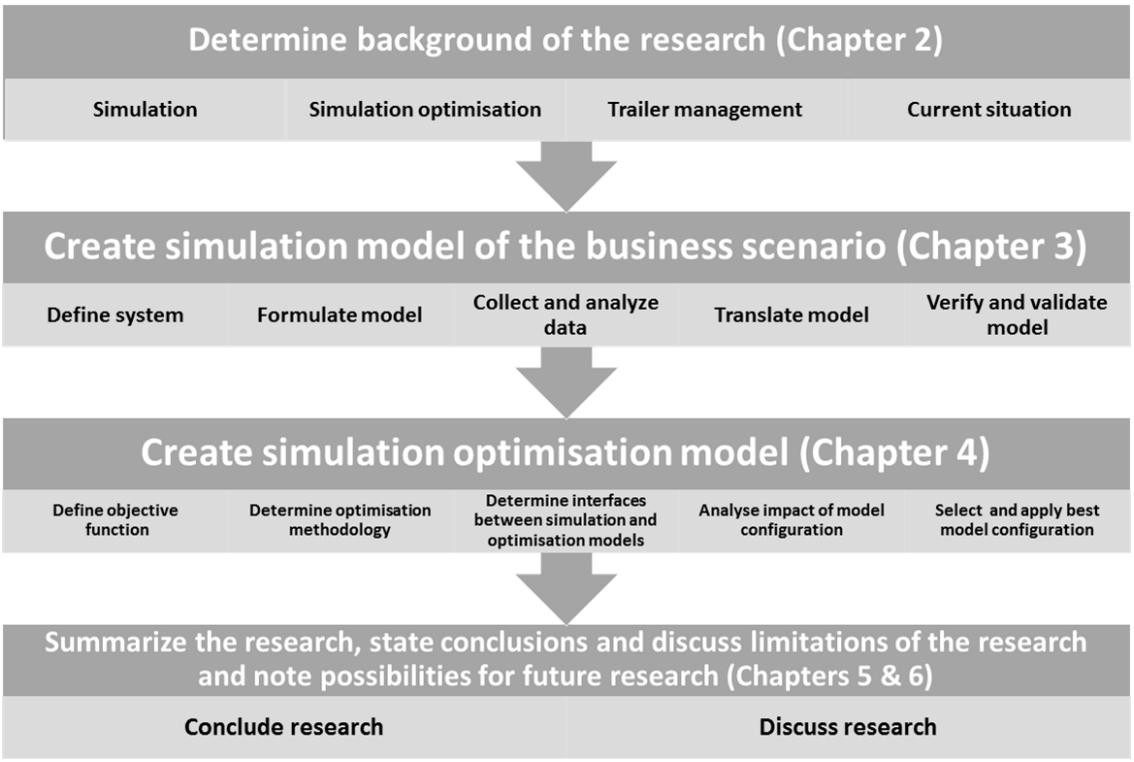

Figure 1.1: Research setup and structure of thesis

First, the necessary theoretical background including the current situation of the company are elaborated on in Chapter 2. Next, the simulation model is designed and discussed in Chapter 3. After introducing the simulation model, the simulation optimisation model is discussed in Chapter 4. This chapter starts with the definition of the objective function. Next, the general methodology is discussed. After this the specific interfaces between the optimisation and simulation part together with the implementation of the algorithm are discussed. Then, the impact of different configurations is tested and finally the best found configuration is applied to solve the optimisation problem. This research is concluded with the conclusions and a discussion in Chapters 5 and 6.

Chapter 2

Theoretical background

This section forms the theoretical background of this research. It starts with exploring the literature relevant to the research project and ends with a qualitative analysis of the current situation at ABI. The aim of this chapter is to bring the reader up to speed on the relevant literature performed in this field of study and subsequently give more insight in the current situation at ABI.

First, a brief introduction on simulation is done in Section 2.1. Next, the field and techniques of simulation optimisation are covered in Section 2.2. As simulation optimisation forms the main focus of this research, this section aims to introduce different optimisation techniques and discusses previous applications of the proposed simulation optimisation technique incorporated in this research. The theoretical background also includes some previous research done in the field of trailer management related to this research 2.3. The qualitative analysis of the current situation at ABI is discussed in Section 2.5.

2.1 Simulation

Simulation has long been a proven tool in the modelling of complex systems such as supply chains where the full complexity of problems cannot be incorporated in mathematical optimisation models (Ingalls, 1998; Law et al., 1991). In order to make successful, meaningful simulation models, they need to be both validated and perceived credible by stakeholders (Law, 2005). In (Law, 2005) seven main steps are discussed to create credible and valid simulation models. These steps are: formulate the problem, collect information and data and construct a conceptual model; validate the conceptual model; program the model; validate the programmed model; design, conduct and analyse experiments; document findings. When one of these steps is not achieved return to the problem formulation and data collection step and iterate until valid models are achieved. Execution of these steps is context dependent, (Law, 2005) describes a variety of specific methodologies for each of the different steps. Involving the stakeholder at regular intervals in the model design is crucial in achieving valid models as well as modelling the right problem.

Sargent (2005) categorises and discusses tools to validate and verify simulation models. Relevant for this research is the validation and verification of models with unobservable outputs. Whilst not technically unobservable, the outputs of the simulation model, as will be discussed in Chapter 3, are not explicitly measured at the moment. In this case Sargent (2005) suggests validating the modelling logic by involving stakeholders at different stages. Furthermore, it is advised to visualise the simulations such that face validity can be established. It helps to quantify results of the simulation model and walk stakeholders (and process owners) through the model. Furthermore, data validity is required in order to achieve valid simulation models. Data must be appropriate, accurate and sufficient and this can be achieved by developing good procedures for collecting and maintaining data, testing the collected data and screening the data for outliers and determine their validity (Sargent, 2005).

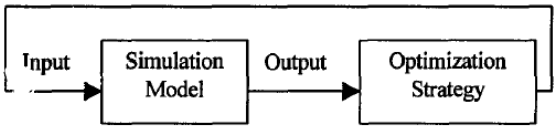

Figure 2.1: Simple simulation optimisation setup (Carson and Maria, 1997)

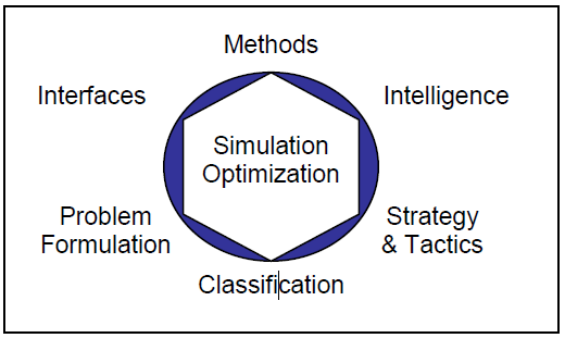

Figure 2.2: Domains of Simulation optimisation (Bowden and Hall, 1998)

2.2 Simulation optimisation

In Carson and Maria (1997) simulation optimisation is defined as "the process of finding the best input variable values from among all possibilities without explicitly evaluating each possibility". In course, the main goal of simulation optimisation can be described as optimising the amount of information obtained from simulations, subject to resources spent. Figure 2.1 shows a simple example of the feedback loop required in simulation optimisation (Carson and Maria, 1997). The inputs of the simulation model translate to simulation outputs that are used by the optimisation strategy to generate new inputs to be simulated.

An effective optimisation strategy is dependent on several facets. Bowden and Hall (1998) describe six simulation optimisation domains that can be used to describe these facets. The six domains are shown in Figure 2.2

The Problem Formulation domain focuses on the construction of objective functions and constraints. In the Methods domain, the focus lies in developing and applying optimisation methods. The Classification domain focuses on classifying and analysing given simulation optimisation models with the goal of selecting effective strategies in solving them. Factors to consider in classification are the number and type of decision variables and simulation output characteristics. The Strategy and Tactics domain focuses on selecting the right methods and using additional techniques to improve efficiency of the optimiser as well as improving search efficiency and accuracy. In the Intelligence domain, the intelligence inside the solver is considered. This includes selecting the right strategy for a given problem and how the employment of different tools is guided during optimisation. The interface domain considers the interfaces between the user, the optimiser and the simulation model. Each of these domains have to be considered when using simulation optimisation techniques (Bowden and Hall, 1998). Furthermore, Bowden and Hall (1998) show the important interaction between the different parts of simulation optimisation research. As many of the parts either depend on or strengthen each other. This paper also claims that historically most research was focused on only one of these domains at the time.

Currently there is a gap between academic solvers and commercial solvers (Hong and Nelson, 2009). In the research community there has been a focus on convergence properties, statistical guarantees and the design of simple algorithms, where commercial solvers focus on heuristics with robust performance regardless of statistical or convergence guarantees (Hong and Nelson, 2009).

2.2.1 Simulation optimisation methods

Azadivar (1999) extensively discussed simulation optimisation methodologies and the importance of effective optimisation techniques. Simulation optimisation methodologies were categorized based on the type of optimisation (single versus multi-objective), input parameter domains (continuous and very large or countably infinite parameter domains versus discrete small parameter domains). The biggest difference between the two categories is the possibility to either evaluate all options or not (Jalali and Nieuwenhuyse, 2015). In the next sections these methodologies are shortly introduced for each of the input parameter domains. As there are countably infinite parameter input combinations possible for the simulation model used in this research, the literature review is focused on optimisation methods for these type of simulations.

2.2.2 Simulation optimisation problems having discrete inputs with small domains

These optimisation problems are often characterised by the ability to exhaustively search all solutions (Jalali and Nieuwenhuyse, 2015). Optimisation is done by comparing each of the different outputs. Typically used examples include Ranking & Selection algorithms and Multiple Comparison algorithms (Jalali and Nieuwenhuyse, 2015).

2.2.3 Simulation optimisation of stochastic and large to countably infinite parameter input space

(Robinson, 2005) describes the methods of meta-modelling and response surface methodology to approximate relationships between inputs and outputs. These methods have been used to optimize complex simulation optimisations with large solution spaces. The advantage of using (validated) meta-models rather than running full simulations has been described in several studies (Hurrion, 2000; Laguna and Marti, 2002; April et al., 2003; Barton and Meckesheimer, 2006). In each of these studies the meta-model was used either as a filter to allow or reject simulations of solutions, or as a replacement of the simulation as a fitness evaluator. In Carson and Maria (1997) asynchronous team (A-team) methodologies are mentioned as "combining several problem solving strategies so that they can interact synergistacilly". Optimisation models using meta-heuristics and meta-models can be regarded as A-team methodologies. A-team techniques aim to be both fast and robust and have since then been applied in a variety of ways.

April et al. (2003) explains how meta-models can be used to reject likely non-optimal solutions in a simulation optimisation set-up. In this paper a general set-up is described where the simulation model generates response $f(x)$ for a given input x. These responses in term are used by a meta-heuristic optimiser to determine new inputs. These inputs are checked using the meta-model to generate meta-model responses $\hat{f}(x)$. The difference with the best solution x^* so far is calculated $(\hat{f}(x) - f(x^*))$ which gives value d. When d is larger than a certain threshold the potential solution is discarded. This general approach is illustrated in Figure 2.3. Using meta-models in meta-heuristic optimisation comes with three main challenges: defining the type and architecture of the meta-model, data collection and training frequency of the meta-model and filtering rules of estimated inputs. Besides discussing the methodology April et al. (2003) also summarizes the biggest problems in simulation optimisation using traditional methods. These are neighbourhood definition, defining the exploration strategy (and how to incorporate past findings), determining what is the best solution and the computational burden of function estimates relative to searching.

Another example were a meta-model filter was used was in Laguna and Marti (2002). The authors perform online simulation optimisation by using a neural network as function approximator and a scatter search algorithm for finding solutions. In order to create the neural network all input data was normalised using the minimum and maximum values in the data set. The researchers used a neural network with one hidden layer and two different activation functions were evaluated. Furthermore, the set of weights found by training the model were subsequently used for linear regression to improve the model. The key focus of the algorithm was to reduce the number of

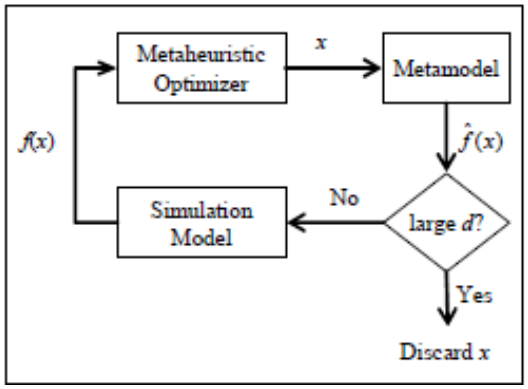

Figure 2.3: General hybrid simulation optimisation setup (April et al., 2003)

simulations by removing expected weak solutions. This methodology allowed online simulation optimisation as the optimisation occurred whilst running simulations.

Daughety and Turnquist (1981) applied response surface estimation on a constrained simulation study by estimating a response surface after each stage of the simulation using all previously obtained simulation observations. The created surface indicates an optimum operation point and this becomes the next starting point for further experiments. This would continue until a stopping criterion was met or the simulation budget was depleted.

In Hurrion (2000) the results from a previous validated discrete-event simulation model were used to train a neural network meta-model using back propagation. The author used a cross-validation approach to select a meta-model. It used 120 random configurations (80 for training, 20 for testing and 20 for cross-validation) of the simulation model to do so. The result was a meta-model with outcomes differing less than 10% of the outcome of running simulations. Subsequently this meta-model was used as fitness evaluation in a simple combinatorial search program.

This technique where meta-models are trained at the start of the optimisation cycle and subsequently optimising the meta-model using traditional optimisation techniques. This technique, also called meta-model based optimisation, has been extensively described in Barton and Meckesheimer (2006). Barton and Meckesheimer (2006) enhances this methodology with local iterated meta-models that aim to generate the local response surfaces, to create a global meta-model that aims to describe the response surface on the entire domain. In creating local iterated meta-models, the concept of locality is crucial for success. As the size of local regions impacts its accuracy. When local regions are defined too tight they may be heavily influenced by the variation of simulation outputs, while when the region is defined as too large it is not possible to map the region using linear or quadratic approximations (Barton and Meckesheimer, 2006). This stresses the importance of either using (ensembled) localised meta-models in optimisation, or allowing re-training (and selection) of a meta-model to improve local approximations. Another advantage of meta-models is that the trained meta-model can afterwards be used in processes with limited decision time frames. However, the biggest problem lies in the degree of abstraction. As a meta-model is effectively an abstraction of an abstraction of the reality, when this abstraction is not accurate enough, meta-model based optimisation is effectively optimising the wrong model (Barton and Meckesheimer, 2006).

The role of meta-heuristic search strategies, such as genetic algorithms, tabu search and simulated annealing in simulation optimisation have been discussed in Robinson (2005). These specific techniques have been extensively described in literature such as Pham and Karaboga (2012) and Davis (1987). These meta-heuristics iteratively explore and improve parameter settings by analysing the outcomes of previous simulation runs and by maintaining or adjusting parameters accordingly. The advantage of meta-heuristic search strategies, is that they generally find good results. A disadvantage of these methodologies is that usually many different simulation runs have to be performed to find an optimum, with the advantage however, that they require little

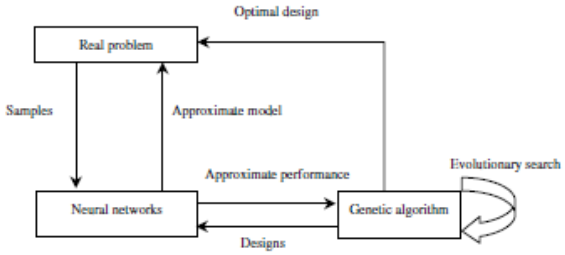

Figure 2.4: Framework of the hybrid GA-NN strategy (Wang, 2005)

human interaction to do so (Robinson, 2005). Furthermore, the property of provable convergence is not always maintained, as some meta-heuristics will completely reject solution regions without evaluating these regions (Fu et al., 2000).

In Xu et al. (2015) the need for multi-fidelity simulation optimisation was expressed, in particular for engineering design questions. Fidelity in simulation is described as the degree to conformance with reality (Xu et al., 2015). Multi-fidelity simulation optimisations combine high fidelity models, which are realistic but expensive to compute, with low-fidelity models which are more abstract but easier to compute (Xu et al., 2015). Meta-models could be effective low-fidelity substitutes of high-fidelity models. In Xu et al. (2014) an efficient algorithm called Multi-fidelity optimisation with Ordinal Transformation & Optimal Sampling was developed that partitions solutions in groups of similar performance on the low-fidelity model. Then it selects n samples from each group and simulates them using the high fidelity model. The main advantage of this methodology is that the grouping uses the lower in-group variances (Xu et al., 2014).

Wang (2005) shows the general effectiveness of combining genetic algorithms (GAs) and neural networks (NNs) for simulation optimisation in engineering design, in this paper the GA was used for generation evaluation using approximations made by neural networks. The focus was a robust algorithm that could be used for a variety of simulation models. In Figure 2.4 a schematic model is shown of the GA-NN hybrid setup used in (Wang, 2005).

In Magnier and Haghighat (2010) an ANN and GA were combined for multi-objective optimisation using simulation for the building design process. The neural network was trained using simulated cases, these cases were defined using Latin Hypercube sampling and the GenOpt optimisation engine. After evaluation the NN was implemented in the NSGA-II algorithm for quick evaluation. Due to the complexity of the problem, optimisation would not have been feasibly when using only the meta-heuristic for optimisation as the simulation time would exceed three years.

Bierlaire (2015) reviews the simulation optimisation literature focused on transport systems. This paper splits possible solutions using the concept of black box and white box optimisation techniques. The black box uses no model specific information in the optimisation techniques and aims to find the optimal solution using the inputs and outputs of the model. White box optimisation takes into account the characteristics of the system by using a physical (meta-)model. Both methods are not extensively being used in the transport literature, and even less so in practice, even though they have been shown to be effective (Bierlaire, 2015).

2.2.4 Genetic algorithms

Genetic algorithms use the concept of evolution as a means for optimisation. It does so by creating generations of individuals, where each individual represents a solution, and then using cross-overs between and mutations of fit individuals to create subsequent generations of solutions. Each solution is represented as a set of genes where each gene is a value for one of the variables. The fitness of each individual is measured using a predefined, goal-specific, fitness function that takes into account the genes.

(Spears et al., 1992) discusses the relative importance of cross-over and mutation in GA's. Cross-over speeds up the process of convergence and can be used to optimise objectives faster,

with the pitfall that it may converge to a local optimum. This unlike mutation which enables deviation from the current gene pool of the population. This shows the importance of the role of mutation in exploring possible solutions. It is however less fitting to use for exploitation due to the randomness and disregard of history involved in the mutation process. Rudolph (1994) analysed the convergence in canonical genetic algorithms and concluded that true global convergence is impossible to achieve without any form of elitism. Elitism ensures that the fittest individuals will remain in the population.

When focusing on cross-over operations in GA's often a distinction is made between ordered and non-ordered genes. For ordered genes the solution order is important. This is for instance the case for travelling salesperson problems where a solution would be a set of locations that should be visited in that order. The solution space of the problem of this research is however not ordered. Therefore, cross-over operations for ordered genes are disregarded. The cross-over operations that are regarded involve single-point, uniform and uniform elitist cross-over.

The single-point cross-over switches the genes from a certain point from two parents and swaps them with each other. The uniform cross-over operation randomly takes a percentage (often 50%) of the different genes from both parents and combines them in the children.

Different kind of selection methodologies have been used in Genetic Algorithms. Most commonly used techniques involve elitism and tournament selection. The concept of elitism keeps the best performing individuals from the generation in subsequent generations to improve convergence. Tournament selection takes n random samples of k individuals and keeps the best individual. The size of n equals the population size so after the n tournaments the new population has been defined. This new population is then being subjected to the different GA operations such as mutation and cross-over. Every value can be sampled multiple times which results in fitter individuals being selected more often for next generations. Tournament selection is the main form of selection used, combined with a little bit of elitism to ensure the best solutions will stay in the population.

2.2.5 Interfaces

(Le Riche et al., 2003) describes two different ways to interface simulation models with optimisers. External interfaces consist of at least two executable programs, the optimiser and simulator, that communicate through files. This often requires a translator program or function that translates simulation outputs to suitable optimiser inputs. Likewise a translation may be needed for (sets of) selected decision variables to simulation inputs. The main advantage of using an external interface is that the simulation and optimiser are completely decoupled. The main downside of external interfaces however is that they often require reloading of the simulation model for all (batches of) simulation runs.

Internal interfaces require that the optimiser and simulation are embedded in the same program. The programs share internal interfaces and can therefore easily transfer information. The main advantage of using internal interfaces is that the simulation can easily remain in a loaded state between replications (Le Riche et al., 2003). This would however require that the simulation and optimisation can be done in the same program, which limits the choice of (often specialised) software.

2.3 Trailer management

As this research focuses on applying simulation optimisation in a real world situation, a brief literature review was performed on the problem simulated, namely trailer management. Trailer management can be divided into three categories: strategic management concerning long term decisions such as in- or outsourcing of the fleet, tactical management including empty vehicle balancing policies and fleet sizing and operational management concerning real-time (specific) decisions such as real-time resource allocation and empty vehicle re-positioning (Crainic and Laporte, 1997).

In Dejax and Crainic (1987) the following problem defining criteria were proposed for handling the empty vehicle reallocation problem: type of model (policy versus operational), type of flow (only empty flows, both empty and full), transportation mode (single transport mode i.e. only trucks or trains versus multi-mode transport), fleet homogeneity (different types of vehicles in the fleet) and the type of company (freight carrier versus industrial company managing an own or rented fleet). Furthermore, Dejax and Crainic (1987) proposes a variety of factors to be taken into account in the methodology such as the time domain (static versus dynamic) and whether the problem is (partly) deterministic or stochastic. Dejax and Crainic (1987) summarise previous approaches used such as algebraic and analytic stochastic models and simulation models. They also mention previous solution techniques such as mathematical programming, network algorithms, stochastic optimisation, simulation. These techniques can be applied both to tactical and operational models (Dejax and Crainic, 1987).

Furthermore, Dejax and Crainic (1987) explain the importance of appropriate and reliable information systems on empty vehicle availability and demand as these are a necessary input for vehicle allocation models. In Song (2005) a mathematical model for optimal empty vehicle repositioning and fleet sizing was made for a two-depot service system. It suggests thresholds for decentralised empty vehicle dispatching as well as optimising initial fleet sizes. Song (2005) also briefly described ways to extend their approach to systems with more depots such as hub-and-spokes systems.

Legato and Mazza (2001) used discrete event simulation for the simulation and optimisation of the berth planning and resource management processes at a container terminal by modelling the system as a closed queuing network. This problem is similar to the trailer management process as resource availability as well as site (or in this case terminal) capacities have to be taken into account.

In Köchel et al. (2003) an evolutionary simulation optimisation methodology was developed for the fleet sizing and allocation problem (FSAP). In this research, a genetic algorithm was used to optimise simulation experiments for general transportation networks. The subsequent simulation outputs were then used in industry specific approximation functions to estimate the true performance of solutions. This methodology was very dependent on the assumptions that were made such as homogeneous fleets, instant employment of rides and infinite transporters as this was required for the approximation models.

2.4 Contribution of this research to existing literature

As discussed in the previous parts of this chapter, there has been done excessive research on simulation optimisation techniques, where these techniques take on many different forms and complexities. There are several subjects that remain less explored in the discussed literature. While it was shown that a simulation optimisation approach, using a genetic algorithm and neural network meta-model, has shown to be effective in various studies (Hurrion, 2000; Laguna and Marti, 2002; April et al., 2003; Barton and Meckesheimer, 2006), little research has been done on the parametrization of the different parts of the optimisation model on the effectiveness of the model. This specific research focuses on an online optimisation algorithm that uses the meta-model as a filter for solutions, rather than replacing the evaluation function all together.

Köchel et al. (2003) discussed the usage of simulation in fleet sizing and allocation problems. This research improves upon this earlier research by applying a simulation model with added complexity that more closely resembles a specific real-world situation (the on-site processing and waiting processes). Besides the added on-site complexity the assumption of homogeneous fleet size is dropped as our model involves both owned (homogeneous fleet) and eternal vehicles (non-homogeneous). As this assumption does not hold in many real-life systems, this would make a more realistic model. Furthermore, Köchel et al. (2003) did not use an approximation algorithm to define which cases to simulate or not, which could ultimately prevent inefficient simulations to be performed. This research tries to solve a similar problem but embeds a meta-model of the simulation (representing the input-output relation) to improve approximation accuracy.

The literature review of Abo-Hamad and Arisha (2011) shows there is a gap in the usage of simulation optimisation techniques in supply chain management compares to research done specifically on supply chain management or general simulation optimisations. For example in 2009, only 2.9% (20 out of 689 papers) on Supply Chain Management used simulation optimisation techniques and only 1.8% (20 out of 1097) of the papers on simulation optimisation focused on Supply Chain Management. This while simulation optimisation has been shown to be an effective methodology in optimisation problems of supply chains due to the uncertainties involved in their objective functions (Abo-Hamad and Arisha, 2011). For supply chain related simulation optimisation studies, meta-model optimisation approaches and meta-heuristic search processes have shown to be applied most often. The review did however not mention one case where meta-model and meta-heuristics were combined as described in this paper. To the best of the author's knowledge this approach has not been used in a similar context. Therefore, this research aims to test the effectiveness in using this approach for problems with this context.

The main concern of the general hybrid simulation optimisation methodology described in April et al. (2003), is the lack of convergence property and the relative high chance that optimal solutions get rejected due to statistical noise in the input-output relations of the simulation model and meta-model accuracy. In order to reduce this phenomenon, this research extends the general methodology with a simple mechanic that forces some solutions to be evaluated regardless of their approximated fitness score. This prevents completely removing solutions from the solution space and therefore prevents possible discarding of the optimal solution. This research also discusses how this simple mechanic may improve the exploration in the optimisation process.

To conclude, this research aims to apply a simulation optimisation methodology that has been proven in other fields, in a field where it has not been used (extensively), furthermore, this research aims to improve upon the existing methodology by analysing the impact of different parameters of both the genetic algorithm and the neural network on the optimisation effectiveness and adding a simply, but mechanism to improve solution exploration and reduce original global convergence problems of the algorithm.

2.5 Current situation

In this section the current transportation situation is described. The main focus is the transportation planning process and trailer unavailability.

2.5.1 Transportation planning process

In the current planning process only the next two days are taken into account. The initial planning is based on the production schedule, which is dependent on sales and stock transfer orders (STO) between locations, but often changed due to uncertain circumstances. The most common circumstances in which the transport planning changes are: trailers not yet being loaded (because of out-of-stocks or trailer unavailability), transporters being delayed (or late) and to an extend defects (on trucks or trailers). Some of these problems also occur because of last minute changes in the production schedule, as demand from customers may change in a late stadium or a production line breaking down (Reynaert, 2017) Whenever delays occur, the transportation schedule is adapted accordingly, in which customer orders are prioritised (to maintain high service levels) (Caignau, 2017). Therefore, one small delay potentially changes the entire planning of the day, resulting in high workloads and a lot of ad hoc decision making to streamline process execution as much as possible.

It is important to consider that the different depots and plants have an unbalanced trailer need, this inevitably results in empty trailers that require to be relocated. Current relocation is however done intuitively rather than looking at explicit (future) needs (Schrover, 2017). It would be helpful to be able to quantify the potential impact of re-locations and streamlining this process accordingly.

2.5.2 Trailer unavailability

Transportation is done using trailers and tankers. One goal of ABI is to minimize the time that products are in inventory and therefore production and transportation are aligned as much as possible. During direct loading, beer (after being packaged) is immediately loaded from the production line onto transport and for this process it is critical that trailers and tankers are available. When there are no trailers it means that the goods have to be stocked which results in multiple additional interactions. Also, as no goods are able to be loaded, shipments get delayed and the problems mentioned before get amplified, where many changes have to be made in the transportation planning (Bastiaens, 2017).

There are various reasons why trailers are not available. Transportation need changes over time and deliveries are not always spread over time. Therefore, peak demands occur which results in many trailers being used at once and not being available for a certain amount of time. This problem can very clearly be seen after weekends, where a lot of pre-loading happens during the weekend, resulting in trailer availability problems on the Monday morning (Lonjaret, 2017). Also, in the current situation it is assumed that for every trailer out, there will be one trailer in. However due to delays when a trailer is outbound it could take some significant time for a trailer to return, potentially resulting in unavailability of trailers at the original location. Furthermore, as there have been other problems in the past where trailers were not loaded on time, they currently try to pre-load as much as possible (Caignau, 2017). The latter could give availability problems when a trailer has already been loaded for a lower priority shipment as unloading the shipment again is very costly and time intensive and is therefore avoided.

2.5.3 Trailer parking capacity

The brewery in Leuven is rapidly expanding which results in less space for trailers (and trucks) to be parked on-site, increasing the need for a streamlined logistic process (Schrover, 2017). On the other side there is an increased need for trailers as more shipments are performed per day. More trailers, in course, results in the need for more parking places.

At the moment there are 107 parking spots for trailers and 9 smaller parking spots for tankers in Leuven (Bastiaens, 2017). In Jupille the parking capacity is 54, but this is currently being expanded to 59 places (Langouche, 2017) and the parking capacity for Hoegaarden is 30 places (Cosemans, 2017). It is important to note however that this is designated parking space. In reality trailers are often also parked at remaining parking spaces such as at the entrance of the site where drivers arrive, and on the docks when production capacity is not required or plentiful (Cosemans, 2017).

2.5.4 Other challenges in trailer management

Previous projects in trailer management showed the importance of trailer processing being spread over the day. The reason for this is the long times trailers are on the road. Spreading the processing of trailers over the day ensures that the amount of needed and returning trailers fluctuates less over time (Lonjaret, 2017; Verhelst, 2017). Unfortunately, while significant improvements have been made in spreading the processing of trailers over the day and the weekend, the actual transportation of the trailers still depends on the transporters and often results in peak demand periods.

Data in the system is not always correct as different sites use different procedures regarding trailers. Seeing as data is collected at different touch points in the internal system, there is a lack of information regarding trailers when there are no touch points, for example at customers or smaller sites and depots without automated systems. Human and technical errors also result in mistakes in the system. An example of this is when trailers are shown to be at multiple locations according to the system. This makes correct management of trailers very difficult (Verhelst, 2017; Richelle, 2017).

Delays are often caused by trailers that are not yet loaded, which can be because of a variety of reasons such as, out-of-stocks, transporters being delayed (when the transporter comes with own trailer) or unavailability of trailers. Furthermore, delays occur because of transporters that are late or other transports have changed and therefore the planning has been adapted based on priority (Valckenborg, 2017; Parijs, 2017).

One large problem with trailer management is the limited amount of control you have on external truck drivers (which handle the majority of rides). It happens regularly that they leave empty trailers at external locations, without notice, as this may fit their own schedules better. This results in trailers being in remote locations where they are not needed, requiring costly extra (empty) rides to return these trailers (Cleton, 2017; Richelle, 2017) as well as trailer availability problems.

Planned shipments arrive from the Business Service Center (BSC) in Prague. Pilots couple the planned shipments to trailers. This coupling process takes into account the type of trailer and the shipment priority. After the shipment and trailer have been linked they will be unloaded and loaded. Before unloading starts or loading completes a check is done. The loading and unloading of trailers can happen in parallel in order to save time (Bastiaens, 2017).

Chapter 3

Simulation model

This section elaborates on the simulation model created to improve trailer management at ABI. ABI currently faces problems with effectively managing their trailer fleet, taking into account fleet availability, parking capacity and costs. For this a simulation model was built that simulates how trailers (both internal and external) move within and between sites. The aim of the tool is to identify the impact of changes in fleet configuration as well as allowing analysis of future scenarios. The simulation model was subsequently used in a simulation optimisation algorithm to find the optimal fleet configuration that minimises costs whilst complying with parking capacity and trailer availability constraints.

This chapter is structured as follows: first the approach used to create the simulation model is discussed in Section 3.1. Next the simulation system is defined as a function of its scope, assumptions made and the required decision variables, parameters and outputs of the model (Section 3.2. Based on this system definition the model is formulated using process models and a description of the model objects and characteristics (Section 3.3). In Section 3.4 the collection and analysis of the input data for the simulation model is discussed. Subsequently, the specific implementation of the simulation model is briefly discussed in Section 3.5. This chapter is concluded with the verification and validation of the simulation model in Section 3.6

The simulation optimisation methodology used and its result are described in Chapter 4. The structure of this chapter is displayed in Figure 3.1.

Figure 3.1: Structure of Chapter 3: Simulation

3.1 Approach

The approach taken for creating this simulation model followed the steps as described in (Law, 2005). These steps are shown in Figure 3.2.

This approach uses iterative stakeholder validation meetings for both conceptual and programmed models to ensure model validity and acceptance. This iterative approach also allows adaptation to changing stakeholder needs. The latter advantage proved crucial for this project as the scope changed multiple times due to data quality problems, changing stakeholders and a shift in approaching the original problem by involved parties. Examples of scope changes include focusing only on Belgium instead of Belgium and the Netherlands, including shipments performed on external trailers, including parking capacity as a focus area and decreasing the timespan from one year to one quarter.

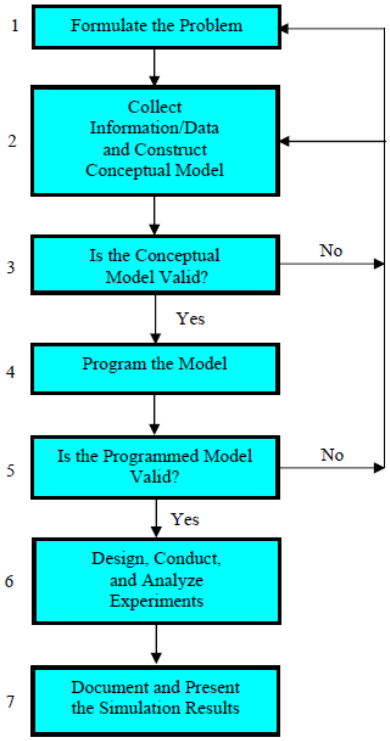

Figure 3.2: 7 steps to create build valid and credible simulation models (Law, 2005)

3.2 System definition

In this section the system simulated and its boundaries are defined. The scope is defined and the assumptions are presented and their impact discussed.

3.2.1 Scope

The system simulated consists of the trailer movements within and between Jupille, Leuven and Hoegaarden, the sites connected to the work flow management system (WMS). As there was insufficient data to fully model the flows of the automatic transport loading system (ATLS) trailers it was decided to not explicitly include them in the model. The duotank processes are not comparable to the other vehicle types and therefore excluded from the model. Furthermore, it was also recognized that problems related to the duotank are not caused by fleet sizing (Caignau, 2017). As no data of the processes outside of WMS (such as operations on depots, customer deliveries etcetera) was readily available, it was decided to regard these processes to be outside the scope of the simulation and treated as a black box. The main business focus is on the original

3.2.2 Level of abstraction

The trailer management process is simulated with a relatively high level of abstraction. As data quality could not be ensured for each individual process (see Section 3.4) step it was decided to focus on a higher level of abstraction. An example of this is the `processing` process. The individual events for the underlying process (`call to dock`, `start unload`, `end unload` and `release from dock`) are available at most plants. However, due to the individual processes being partially automated and partially manual it occurs that some events get skipped and take zero time. As there are multiple sites, multiple vehicle types and each of them having specific process steps, it would require a lot of additional effort for minimal return. As data was however accurate and

comparable for arrival time, departure time and the calling to the dock and departure from the dock times it was possible to model a higher level version of the transportation processes. These processes are described in Section 3.3.1.

3.2.3 Assumptions

In this section the assumptions are discussed that have been made with regards to the simulation model.

1. External trailers have priority in being processed before owned trailers. In reality, a priority system is used based on a variety of factors such as shipment type and planned departure date. Generally, this results in external trailers having priority over owned trailers. The impact is therefore expected to be negligible.

2. External vehicles do not use parking places, instead a fixed amount of parking spots is reserved for the external vehicles that are used for pre-loading. This assumption mostly holds for live-loaded shipments as they usually arrive at the reception and leave directly after processing. As there is a variable amount of external vehicles used for pre-loading, and there is no precise data available for this, it is difficult to include these trailers in a more specific way.

3. The actual transportation contents of a transportation order are not taken into account. Orders are only distinguished on the required vehicle type. The contents of trailers can impact the transportation processes when the actual contents cannot be loaded and in the time required to process the trailer. In practice, however, re-planning will be done when contents cannot be loaded to prevent lost capacity and transport penalties. In reality, processing times do not differ significantly. Therefore, it is assumed that the impact of this assumption will minimally impact performance.

4. Resources (parking spots and docks) have a fixed capacity for the duration of the simulation. In the real-world, capacity may change based on scheduled workforce or construction on site.

5. Individual trailers are only distinguishable from each other by types. In reality there is a distinction between trailers. Some more expensive models can be loaded and unloaded quicker. However, the only available data was the general vehicle type (trailer/tanker). But as there is no distinction between trailers in the fitting of the distributions, this property is partly included in the fitted distributions.

6. No explicit distinction is made between processing orders (loading/unloading etc.). Even though typing of process orders is not included explicitly, it is included implicitly as certain types have shorter process times. The variability in process times is taken into account by fitting the distribution of all orders.

7. When the parking is full after a trailer is just processed, it will wait on the dock before departure until a parking spot becomes available. When this happens in the real world, either the trailer leaves directly (if there is transport available) or it will be parked "illegally" outside of a parking spot. When both things are not possible the trailer will remain on the dock.

3.2.4 Decision variables

In this section the decision variables of the simulation model are discussed. Decision variables are those variables that can be changed in the real processes. The optimisation part of this research aims to optimise the configuration of these decision variables to find an optimal solution.

In Table 3.1, an overview of the decision variables is given, together with a definition and the total number of variables of that type. The number of variables is derived from the amount of combinations resulting from its indexes. There are three plants (i) (Leuven, Jupille and Hoegaarden)

and two vehicle types (v) (trailers and tankers). The feasible parameter ranges for each of these decision variables have been established together with stakeholders and can be found in Appendix B. These feasible parameter ranges are based on the current operations of the different plants. The parameter ranges also reduce the complexity of the optimisation problem as it greatly reduces the solution space.

The parameters that can be changed in the set-up are the amount of vehicles (trailers and tankers) at each of the sites, the amount of parking places at each of the sites and the amount of shipments performed on external vehicles. This latter decision variables came from the question of ABI on how the performance and costs of transport would change when the focus of fleet management would shift more towards using external trailers rather than using owned trailers. The advantages of external trailers are that they spend less time on the site, they are easier to up and downscale in advance and there is less responsibility for damages and maintenance (Schrover, 2017). The down sides are that they cost more, there is more dependence on the transportation company for arriving on time and it is more difficult to apply direct loading techniques. The latter is difficult as this would require a strong fit of transportation and production, which has been difficult due to many production changes in the past (Schrover, 2017).

Due to the high complexity of maintaining the trailer fleet it could be interesting to consider changing to using more external trailers. The stakeholders were interested in seeing, especially when demand would grow, what would be the bottleneck in the trailer process. For that the simulation model needed a way to adjust the amount of shipments done on external vehicles. To achieve this decision variable $m_{i,v}$ was created which allows the up and down-scaling of the total amount of shipments done on external trailers.

Table 3.1: Decision variables

Decision variable	Definition	Number of variables	Type
$r_{veh,i,v}$	The number of owned vehicles of type v at plant i	6	Discrete
$r_{park,i}$	The number of parking places at plant i	3	Discrete
$m_{i,v}$	Multiplication factor for external vehicle arrival rate for vehicle type v at plant i	5	Continuous

3.2.5 Model parameters

To simulate the trailer flows, the following model parameters (Table 3.2 are identified.

Table 3.2: Model parameters

Parameter	Definition
$\lambda_{call,i,v}$	Inter-arrival rate for calls to be processed at location i for vehicle type v
$\lambda_{ext,i,v}$	Inter-arrival rate for external vehicles with type v arriving at plant i
$proc_{i,v,o}$	Processing time for a (not) owned o vehicle with type v at plant i
$P_{i,j,v}$	Transition probability that an owned trailer of type v goes from plant i to plant j.
$out_{i,j,v}$	Expected out time between location i and j for vehicle type v
$wait_{i,v}$	Expected wait time before departure for vehicles of type v at plant i

Processing times, out times and wait for departure times have been determined by fitting theoretical distributions that resemble their empirical distribution function (Sections 3.4.4 to 3.4.7. The inter-arrival times of calls and external trailers were modelled based on their real-life behaviour and is further explain in Section 3.4.3. The transition probabilities are further explain in Section 3.4.8.

3.2.6 Performance measurements

The following statistics will be measured during the simulation:

- Number of shipments on external vehicles per vehicle type
- Number of occurrences that an order has to wait at least 12 hours before being processed per location (representing unavailability of vehicles)
- Number of occurrences that the parking is completely occupied per location (representing unavailability of parking)

As the goal of the project is to reduce trailer availability problems by adapting the fleet configuration, it is important to know how many vehicles are being used (both owned and external) and how big the availability problem is. Availability problems can be derived when trailers are not available when they are needed. When an order has to wait for a significant time (defined as 12 hours) it may be assumed that there was an availability problem. On the other side, parking problems can be expected whenever the parking is completely full. Besides the availability, it is also important to know how big the parking capacity problem is, in order to include it in any decision making regarding the trailer management problem. Hence why the number of occurrences that the parking is full is tracked.

3.3 Model formulation

In this section the model is formulated based on the system definition. The process is displayed using a conceptual process model (modelled using coloured petri-nets). Concepts such as network characteristics are explained. Finally, the model objects and their characteristics are elaborated on.

3.3.1 Process models

Trailers arrive and are placed in a queue waiting to be called for processing. Processing orders, arrive in the system and are placed in a queue to wait for an available trailer of the required type. The arrival distributions for both external trailers and processing orders is elaborated on in Section 3.4.3. When there is both a trailer and an order available of the same type, then they will be matched together. In this step, the information of the order is copied to the trailer. While not explicitly modelled in the CPN model, the order determines the different process delays such as processing times, wait time for departure and out time for the trailer. The different process delays are discussed in Section 3.4.5. When a dock is available, the vehicle will claim the dock and processing will start. After processing, two scenarios may occur: when a parking spot is available, the trailer will be moved to the parking and the dock will be released. Then it will wait for a determined time before departing. When no parking is available, the vehicle will instead wait on the dock, until there is a parking spot available after which it will be moved to the parking. When a vehicle starts waiting on the dock it will only wait for the remainder of the time when it claims a parking spot. When it waits for the entire duration before a parking spot becomes available the trailer will leave the system without claiming a parking spot and it will release the dock when it leaves. If the trailer claimed a parking spot, it will release it, otherwise no parking resource was used. The time a trailer waits before departure is predetermined using historical data. This simulates the effect of pre-loading where a trailer is loaded directly from the production line, but is shipped hours later. Based on whether the trailer is owned or not it will either depart and leave the system or be assigned a destination after which it will depart to this next destination. In Section 3.4.8 it is explained how these destinations are determined.

The conceptual process model can be seen in Figure 3.3. The conceptual process model was modelled using coloured petri-nets (Ratzer et al., 2003).

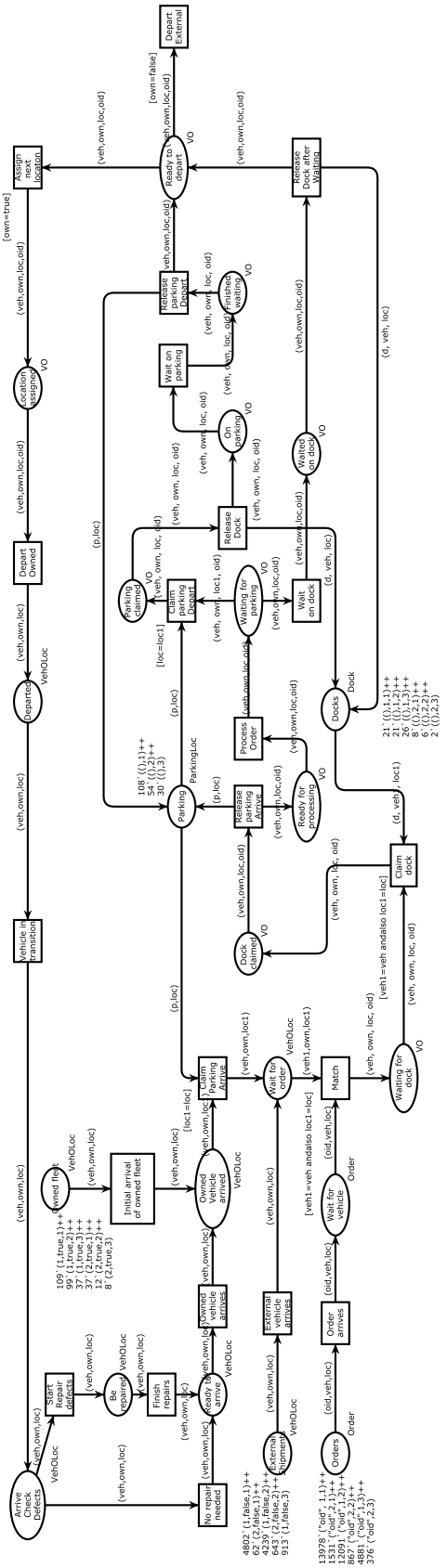

Figure 3.3: Conceptual process model

The conceptual model is created as a coloured petri net according to the specifications defined in Ratzer et al. (2003). It is to be interpreted as follows. Places (the round shapes) are states in which an object of a specific type can be. An object can transition from one state to another through the means of transitions (the square shapes). Places and transitions are connected with each other through arcs (the arrows). The inscription of the arcs determine which attributes of an object are passed to the next place whenever a transition is fired (terminology used for an activated transition). The type of object of a place is denoted in the bottom right corner of the place. For example the place `Owned vehicle arrived` is a place for an object with type `VehOLoc`. `VehOLoc` is a custom defined object, that is combined of the attributes vehicle type, owned/not-owned attribute and a location attribute. In petri-nets these attributes can be represented by so called colour sets. In Figure 3.4 the different colour sets are shown used by this conceptual model. An object of type `VehOLoc`, a combination of a vehicle type, owned/not-owned attribute and a location, represents a real-life vehicle (trailer or tanker) that is either owned or not that is currently at one of the locations.

The behaviour of the model is subsequently dependent on the attributes of the objects at the places, how these places are connected with each other by transitions, the inscriptions on the transition (called guards which represent a logical check) and which inscriptions are on the arcs that combine these transitions and places. A transition can only be fired when at each of the places that are connected to it with an incoming arc, an object is available that complies with all logical demands of the transition to fire (i.e. right attribute, right amount of objects etc.). For example, the transition `Claim parking arrive` can only fire when a `parkingLoc` object is available in place `parking` and a `VehOLoc` object is available in place `Owned vehicle arrived` and both objects have the same value for `location`. This transition will only fire (for the two specific objects) if each of these demands hold. A more specific definition of coloured petri nets can be found in Ratzer et al. (2003).

```
colset Owned = BOOL;
colset Location = INT;
colset Parking = UNIT;
colset OID = STRING;
colset ParkingLoc = product Parking * Location;
var Own: Owned;
colset VehType = INT;
colset Dock = product UNIT * VehType * Location;
colset Order = product OID * VehType * Location;
colset VehOLoc = product VehType * Owned * Location;
colset VO = product VehType * Owned * Location * OID;
var oid: OID;
var veh: VehType;
var veh1: VehType;
var own: Owned;
var loc: Location;
var loc1: Location;
var p: Parking;
var d: UNIT;
```

Figure 3.4: Defined colour sets

3.3.2 Model objects

The main model objects are trailers, orders, departures, docks and parking spots. Trailers and orders were modelled as entities in the system while parking spots and docks were modelled as resources. In Table 3.3 an overview is given of the different model objects.

Table 3.3: Model objects

Object	Definition	States
Trailer	Transportation vehicles of any vehicle type (trailer or tanker)	{waiting for processing, processing, waiting for departure, out}
Call	Call for processing for a specific order	{waiting to be started, started}
Dock	Location where trailers get processed	{available, unavailable}
Parking spot	Location where trailers get parked before and after processing	{available, unavailable}

Even though both parking spots and docks are modelled as resources it is important to note that parking spots do not have a clearly defined total capacity. In reality only certain spaces are reserved for parking trailers, however when a situation occurs that all dedicated parking spots are filled they are instead parked anywhere possible on the sites. This is an unwanted situation but nevertheless it is not correct to assume that the transportation process would be disrupted too much when this situation occurs (to a certain degree). Especially in the weekends, when there is less transport activity, the parking fills up often, but due to less transport activity it does not give significant problems (van Hulst, 2017). Regardless, parking spots are modelled as resources (as this allows for easier tracking of statistics) with an excessive amount. This excessive amount was determined with stakeholders and should be perceived as the situation that serious problems will occur. When this capacity is reached, arriving trailers are asked to come back later (Caignau, 2017). This was modelled as a fixed delay of 1 hour. On a sample of 75 working days it was found that on average 10.0 external trailers are available for pre-loading with a standard deviation of 5.5 trailers.

3.3.3 Entity characteristics

In the simulation two different kinds of entities will be used namely entities depicting trailers and entities depicting shipment orders. The trailer entity represents a physical trailer which moves from location to location, the shipment represents a virtual entity which provides information for what the shipment is supposed to do.

Trailers: represents a trailer in the real world system. It has the following characteristics

- TrailerType: This describes the type of trailer (SR26, Road Tanker) that is used for the shipment.
- OwnedTrailer: Describes if the trailer is owned or not

Call: represents a call for processing

- TrailerType: This describes the type of trailer (SR26, Road Tanker) that is used for the shipment.
- NextPlant: Determines the location where the trailer will go afterwards (only for owned trailers). This NextPlant is one of the WMS locations: Leuven, Jupille or Hoegaarden.

3.3.4 Network characteristics

When simulating queuing networks it is important to make a distinction between open and closed queuing networks. An open queuing network assumes entities entering the system, getting serviced and then leaving the system. A closed network assumes that there is a fixed quantity of entities in the system that do not leave the system, and there are no other entities entering the system. A combination can occur in the form of mixed networks where part of the system contains entities that enter and leave the system and part of the system contains entities that stay in the system.

(Walrand, 1988). The latter most closely describes the trailer management process as the process contains both owned trailers (which stay inside the network) and external trailers that enter and leave the network. The mixed network approach could also be used for sizing the owned trailer fleet up and down as this will make owned trailers enter and leave the system. It was therefore decided to create the simulation model as a mixed queuing network.

The network can be defined as a set of nodes and arcs that span these nodes. Each of the nodes presents a possible origin and/or destination location and the arcs present a connection between these locations. In Section 3.4.8 the specific transitions between the locations are elaborated on.

Each of the locations has a pool of resources that constrain on site processes. There are two types of resources: parking spots and docks. Parking spots are physical places where any type of trailer can be parked when it is either waiting for processing or waiting for departure. Docks are places where trailers can be processed (unloaded and loaded). As different vehicles are processed different, there are different types of docks to process them on. These docks cannot be interchanged. There are docks for trailers and for tankers. In Table 3.4 an overview is given of the resources per site.

Table 3.4: Resources per site

Site	Trailer Docks	Tanker Docks	Parking
Leuven	21	8	108
Jupille	21	6	54
Hoegaarden	26	2	30

3.4 Data collection and analysis

The simulation model that was created to analyse and optimise the trailer management process is created based on the data from the internal WMS system. The advantage of this is that there was a lot of tracked data available, allowing a data-driven approach to analysing the problems. The downside was that the WMS system had not been fully deployed at all sites. WMS was deployed at production sites Leuven, Jupille and Hoegaarden. Other sites, such as depots, smaller production sites and external warehouses are not (fully) connected to this system. In this section the quality of the data is discussed and outliers identified. Furthermore, data transformations and additional calculated fields are discussed.

ABI uses a work-flow management system (WMS) in which all different plannings and movements come together. This system is used by employees to manage the different processes and provides data related to process execution. Of particular interest for this project is data related to shipments. Shipments represent the effective events taken by a trailer when they are at a site (arrival, call to dock, free from dock etc.). Each shipment line also contains information about the trailer itself such as type, plate number and whether it is an ABI trailer or not and it also contains information on the unload customer and load customer. As this project required the process times rather than event time stamps, these times had to be derived. This is discussed in Section 3.4.1. In Appendix D several example data rows are given from the WMS data.

Data for shipments is readily available for the period 2014 until 2017 for the plants Leuven, Jupille, Hoegaarden and Dommelen. As the focus lies on the Belgian fleet, Dommelen was left outside the scope of this research.

3.4.1 Data enrichment

In this section the relevant calculated fields are discussed.

Processing time: The processing time is calculated from the moment a trailer is called to the dock until the moment the dock gets declared free. When loading and unloading are split

(which can be derived from the SPLIT_LOAD field) the process time is calculated as the sum of time to load ((FREE_DOCK_UNLOAD − CALL_TO_DOCK_UNLOAD) + (FREE_DOCK_LOAD − CALL_TO_DOCK_LOAD). Furthermore, it is checked what kind of shipment it is (unloading and loading, only unloading, only loading, nothing).

Plant: Originally not included in the WMS data is the plant at which the shipment takes place. This was added when the shipment data of the individual sites was consolidated.

Out time: This is only calculated for owned trailers and is calculated using a VBA script which can be found in Appendix D. The out time is defined as the time it takes between departure of a trailer at one WMS site until its next arrival at another WMS site.

Wait before departure time: This is calculated as the time between finishing processing (loading, unloading or both) and the actual departure of the trailer. It is calculated as follows: ((DEPARTURE − FREE_DOCK_(UN)LOAD where it depends on the type of transport whether the departure gets deducted by the free dock load or free dock unload time.

3.4.2 Data quality and outliers

Data quality is one of the most common problems in data-driven research projects such as simulation studies. Therefore, the data was critically assessed to determine its quality. Unfortunately, there were some significant problems with the data quality which can influence the simulation validity.

First of all, nearly half of the data from Hoegaarden in 2015 was missing. This was due to some problems with the WMS export when it was made back then. The data were unfortunately not possible to retrieve. Most of the 2015 data of Hoegaarden can be used, but due to the missing shipments it is not possible to accurately determine the out times (as this assumes interdependency between shipments). It was therefore decided to filter the out times of Hoegaarden 2015 and instead only use 2014 and 2016. Another problem that was found in Hoegaarden was the processing times of tankers. For Leuven and Jupille these times were accurate, but for Hoegaarden no times were measured or near zero-times. These shipments cannot be ignored as they form up to $10-15\%$ of all road tanker shipments. The processing times of tankers in Hoegaarden will instead be substituted by the processing times in Leuven as the processes are comparable at these plants. Automatic Transport Loading System (ATLS) trailers form an important part of the transportation strategy of ABI. Unfortunately, due to its recent implementation there was very limited data available and the data that was available gives an inaccurate view on the situation due to initial implementation problems. ATLS trailers will be deployed on specific flows and will mostly replace the trailers for those flows. The difference between ATLS trailers and normal trailers are that they can only be used with a specific installation (present at the sites and the target customers) and that it allows significantly faster loading and unloading. This is achieved by an automated system. Only 1.45% of shipments (done on owned trailers) in 2016 was done on ATLS trailers. But this will like become 10-20% of all owned trailer shipments in 2017 when the ATLS processes have been improved, more ATLS trailers are made available and the ATLS systems being implemented at other sites.

Another big problem is the high degree of short distance shipments. These shipments consist of shipments within for example two parts of a site (i.e. warehouse and production). For instance in Jupille or Hoegaarden, trailers leave one part of the site and drive to another part of the site using the public road. This means that the departure time of one shipment (almost) equals the arrival time of the next shipment. Whilst, this should not be a problem, it was found that a small measuring difference at leaving and subsequently entering the site, results in many negative out times. As this is a very common occurrence (34,116 cases) filtering would result in many rejected, otherwise valid cases. Instead these cases were identified and replaced by 0.

Any other negative values are likely the result of human errors or deviations from the original business processes. On an average day two to five trailers (circa one percent) would be misplaced according to the WMS system, requiring manual intervention in the system to be used again for further shipments (Caignau, 2017). This and the recounting activities that have to be performed due to the known problems in the system, result in unnecessary costs and more inefficiency. It

was therefore decided to exclude the related shipments. On the contrary there were also some shipments with exceptionally long times (processing, waiting for departure and out times). While some of these outliers have some validity (for instance, trailers have been known to get lost for multiple weeks to months), often they are due to human errors (manually entered dates and times) or context specific behaviour. An example of this is trailers staying at a dock for several hours when it is a day with few shipments. Normally a trailer gets moved immediately to the parking, but when there is no priority in doing so it may be decided to just keep it standing on the dock (de Ridder, 2017) In either case they skew the expected process significantly. In a process that is already highly variable these outliers only increase the variability. Process owners provided expert knowledge which was used to remove outliers by defining minimum and maximum value bounds. The value bounds for the outliers, together with the number of rejected cases, are shown in Table 3.5.

Table 3.5: Outlier bounds and removals

	Minimum time	Maximum time	Cases rejected	Percentage rejected
Out time	0 h	120 h	5135	2.25 %
Processing time	0 h	12 h	3861	1.72%
Wait time before departure	0 h	72 h	3335	1.48%

3.4.3 Arrival distribution

A significant part of the complexity of this simulation model can be found in the arrival patterns of the different entities (calls and arrivals of external trailers). The reason for this is the arrivals show a high degree of non-homogeneity during the day and week. For example, the Leuven plant produces 24/7 both during the week and in the weekend. However, due to operating hours of the other plants, customers and transporters, transportation is very limited in the weekends. This is important for the trailer management as many flows are temporarily blocked during these periods. A common hypothesis of many trailer problems include the imbalance of supply and demand during the day (Lonjaret, 2017; Schrover, 2017; Bastiaens, 2017). Problems also occur at the transitions from weekends to weekdays as loading (Caignau, 2017; Schrover, 2017).

In order to simulate this complex demand process the expected arrivals at different time arrivals are calculated using the percentage of arrivals during specific periods. A classification can be made on a time of day and type of day basis. The day can be split in three parts, night (22:00-5:59), morning (6:00-13:59) and afternoon (14:00-21:59) and a distinction is made between weekdays and weekends. This split represents the actual arrivals of vehicles during the day and the week.

In total this gives $92*3 = 276$ different time intervals per three months which can have $2*3 = 6$ different classifications and this would have to be calculated for both calls and the arrivals of external vehicles, seeing as there are also 3 different locations for which these distributions behave different, it was instead decided to derive expected arrivals by determining the amount of shipments at each location per quarter. Then using ratios per time of day and the type of day to determine the expected number of arrivals per time interval. Historical WMS data was used to derive these ratios as well as the expected number of shipments over the entire period.

The JaamSim software used to simulate the problem (See Section 3.5) is limited to the use of a Non Stationary Exponential Distribution for complex variable distributions. This uses a Non Homogeneous Poisson Process implementation (JaamSim Development Team, 2017). In order to use this, a time series with expected arrivals per interval is required. This time series has been generated using a Python script that generated the expected cumulative arrivals N at time t using Equation 3.1. In this formula m depicts a scale multiplier (to easily up- and downscale the expected amount of shipments), $E[S]$ is the expected total amount of shipments for the total period, r_d and r_w are respectively the time of day and type of day ratio. q_{size} depicts the size of

the quarter ($365/4 \approx 92 days$) and w_{size} is the size of the type of day, effectively 5 for weekdays and 2 for weekends.

$$N_t = N_{t-1} + m\frac{E[S]r_d r_w}{q_{size} w_{size}} \tag{3.1}$$

The expected cumulative arrivals at time t ($N(t)$) are determined for 8 hour intervals (the size of one shift).

The expected transportation need is directly relatable to the amount produced at each of the sites. Additional filling lines therefore have a big impact on the expected transportation need as each filling line increases productivity substantially (de Ridder, 2017). The expected number of shipments for the base case were therefore derived from the actual number of shipments in the period of April 2017 until July 2017 as this most accurately describes the current demand process. Due to additional filling lines installed at the different sites the previous years taking a (weighted) average of the previous years would not give an accurate estimate. Furthermore, including previous months would not be desirable either as it was made clear that the simulated system should simulate the high season as this is where historically most of the problems occurred (de Ridder, 2017). The expected number of shipments per plant and vehicle type are shown in Table 3.6. In this table a split has also been made on shipments on owned versus on external vehicles. The expected number of shipments on external vehicles will be used in modelling the arrival of external vehicles.

Table 3.6: Expected number of shipments for a period of 3 months

Plant	Vehicle Type	Shipments on external vehicles	Shipments on owned vehicles	Total shipments
Leuven	Trailer	4,802	9,176	13,978
Leuven	Road Tanker	62	1,469	1,531
Jupille	Trailer	4,239	7,852	12,091
Jupille	Road Tanker	643	224	867
Hoegaarden	Trailer	913	3,968	4,881
Hoegaarden	Road Tanker	0	376	376

3.4.4 Distribution fitting

In this section the approach taken to fit distributions with certain characteristics to the different simulation parameters is discussed. Due to the many different parameters that require distribution fitting the author aimed to create a clear and (mostly) standardised methodology for fitting these distribution. This methodology, which performs both qualitative and quantitative analysis has been inspired by (Delignette-Muller et al., 2015) and uses the `fitdistrplus` package in R Studio (RStudio Team, 2015). The following steps have been performed in order to define the distributions of the parameters:

1. Filter outliers based on expert opinion of process owners
2. Plot Cullen & Frey graph to get an indication of skewness and kurtosis of the empirical distribution
3. Fit gamma, lognormal and normal distributions to the empirical data using Maximum Likelihood Estimation (MLE)
4. Plot the histogram, cumulative distribution function, quantile-quantile(Q-Q)-plot and probability-probability(P-P)-plot for the empirical data and the candidate distributions

5. Calculate the goodness-of-fit values

6. Based on the plots and goodness-of-fit values, select the model distribution

The different distributions are noted in the form $Lognormal(\mu, \sigma^2)$, $Gamma(\alpha, \beta)$ and $N(\mu, \sigma^2)$vc for the lognormal, gamma and normal distribution respectively. In Appendix A an example case of how the distribution fitting has been performed is shown.

3.4.5 Processing times

As different vehicle types use different processes and processes have not been fully standardised between plants, it was not possible to assume a single processing time for vehicles. Instead, processing times were assumed to be vehicle type and plant specific. Furthermore, it was found that external trailers were generally processed faster than owned trailers. This is likely due to the higher importance of having short lead times for external trailers (due to penalties for delays) and a different mix of shipment types (drop-offs and pick-ups generally take less time than combined drop-offs and pick-ups). Due to this it was decided to make the processing times also dependent on whether it is an owned trailer or not. This results in 12 different processing times. In Table 3.7 the fitted distributions are shown. The notation used is $proc_{loc,vt,own}$ where loc represents the location (lv=Leuven, jup=Jupille and hoe=Hoegaarden), vt represents the vehicle type (trail=trailer and tank=tanker) and o represents whether it involves owned (yes) or external (no) trailers.

It can be seen that the processing times in general have a high variability. This can for a big part be explained by the different types of shipments. The different shipments types (deliveries, pick-ups and shuttles) each take a different amount of time. However it was decided not to model this as well as this would make the model significantly more difficult to manage.

Table 3.7: Processing times

Parameter	Distribution	Mean	Variance
$proc_{lv,trail,yes}$	$Gamma(1.155, 0.624)$	1.851	2.965
$proc_{lv,trail,no}$	$Gamma(1.800, 1.447)$	1.244	0.859
$proc_{lv,tank,yes}$	$Gamma(1.026, 0.216)$	4.731	21.808
$proc_{lv,tank,no}$	$Lognormal(0.033, 0.665)$	1.290	0.928
$proc_{jup,trail,yes}$	$Lognormal(0.102, 1.058)$	1.940	7.782
$proc_{jup,trail,no}$	$Lognormal(-0.080, 0.626)$	1.122	0.606
$proc_{jup,tank,yes}$	$Lognormal(0.480, 0.736)$	2.120	3.237
$proc_{jup,tank,no}$	$Lognormal(0.345, 0.438)$	1.554	0.512
$proc_{hoe,trail,yes}$	$Lognormal(-0.807, 1.633)$	1.691	38.325
$proc_{hoe,trail,no}$	$Lognormal(-0.484, 0.763)$	0.824	0.538
$proc_{hoe,tank,yes}$	$Lognormal(-0.563, 0.738)$	0.747	0.404

3.4.6 Out times

The out time is defined as the time that a trailer is not physically at one of the main sites, (i.e.), Leuven, Jupille or Hoegaarden. The out times are logically dependent on the travel times between two sites but they are also dependent on the type of vehicle and the amount of external locations they visit before reaching a main site. The latter is dependent on the start location and also the next destination. Therefore the out time is determined per vehicle type per destination for each of the plants.

Only the out flows of owned trailers are modelled. The reason for this is that external trailers may be used for several clients outside ABI. It was decided to model the external trailer flow as an open queuing network where the entities enter the system based on their arrival rates and where they leave the system after finishing their onsite operations.

In order to determine the out time distributions it is important to note that there are different types of shipments and scenarios to consider when a trailer is "out". These different scenarios heavily influence the actual time it takes for the trailer to arrive at the next plant. It could be that while a trailer is out that in reality it is visiting several clients or depots before reaching the next plant. Another scenario could be a direct trip between plants, for instance to transport packaging materials. It is therefore difficult to correctly model out times using a single theoretical distribution. The empirical distribution of out times often showed a very irregular distribution with multiple spikes at different values. Instead of fitting one distribution for the out time for each case, it was decided to split the total range of values into different value ranges based on the empirical distribution of the out times. For each of these ranges a theoretical distribution was fitted.

The simulation samples out time values from the different fitted distributions with a chance proportional to the amount of cases that were in the accompanying value range. The result is a more accurate representation of the out process in the simulation. Regardless, the out times are still treated mostly as a black-box. In order to improve the simulation in the future it would be advisable to track information (for instance using GPS) to see which processes really occur when the trailer is out.

In Table 3.8 the derived out time distributions with the respective value ranges used for fitting the distribution are shown for the individual cases. The notation used is $out_{or,dest,vt}$ where or represents the origin location (lv=Leuven, jup=Jupille and hoe=Hoegaarden), $dest$ represents the destination location (lv=Leuven, jup=Jupille and hoe=Hoegaarden) and vt represents the vehicle type (trail=trailer and tank=tanker).

For the out times for trailers from Hoegaarden to Hoegaarden it was not possible to fit a distribution due to problems with the derivation of the out times. Nearly all cases had negative or zero values. The impact of this parameter value in the simulation is that in reality the average availability of trailers (specifically in Hoegaarden) would be lower and less parking problems would occur in real-life than in the simulation as it takes longer for trailers to re-enter the site and in course filling the parking.

Table 3.8: Out times

Parameter	Distribution	Min (of data)	Max (of data)	Probability
$out_{lv,lv,trail,[0,2]}$	$Gamma(0.136, 0.303)$	0	2	0.070
$out_{lv,lv,trail,[2,12]}$	$Lognormal(1.668, 0.356)$	2	12	0.694
$out_{lv,lv,trail,[12,72]}$	$Lognormal(3.186, 0.473)$	12	72	0.211
$out_{lv,lv,trail,[72,120]}$	$Lognormal(4.499, 0.147)$	72	120	0.023
$out_{lv,jup,trail,[0,1]}$	$Gamma(0.54, 1.441)$	0	1	0.009
$out_{lv,jup,trail,[1,3.5]}$	$Lognormal(0.463, 0.23)$	1	3.5	0.589
$out_{lv,jup,trail,[3.5,72]}$	$Lognormal(2.886, 0.772)$	3.5	72	0.338
$out_{lv,jup,trail,[72,120]}$	$Lognormal(4.518, 0.151)$	72	120	0.063
$out_{lv,hoe,trail,[0,2.5]}$	$Lognormal(-0.213, 0.384)$	0	2.5	0.376
$out_{lv,hoe,trail,[2.5,14]}$	$Lognormal(1.911, 0.344)$	2.5	14	0.350
$out_{lv,hoe,trail,[14,35]}$	$Lognormal(3.037, 0.222)$	14	35	0.172
$out_{lv,hoe,trail,[35,72]}$	$Lognormal(3.945, 0.182)$	35	72	0.067
$out_{lv,hoe,trail,[72,120]}$	$Lognormal(4.465, 0.147)$	72	120	0.033
$out_{lv,lv,tank,[0,3.5]}$	$Lognormal(-1.915, 2.196)$	0	3.5	0.018
$out_{lv,lv,tank,[3.5,7]}$	$Lognormal(1.643, 0.14)$	3.5	7	0.358
$out_{lv,lv,tank,[7,40]}$	$Lognormal(2.679, 0.454)$	7	40	0.507
$out_{lv,lv,tank,[40,72]}$	$Lognormal(4.01, 0.175)$	40	72	0.062
$out_{lv,lv,tank,[72,120]}$	$Lognormal(4.516, 0.153)$	72	120	0.052
$out_{lv,jup,tank,[0,0.5]}$	$Lognormal(-1.848, 0.721)$	0	0.5	0.010
$out_{lv,jup,tank,[0.5,4]}$	$Lognormal(0.478, 0.233)$	0.5	4	0.908
$out_{lv,jup,tank,[4,72]}$	$Lognormal(2.414, 0.674)$	4	72	0.080
$out_{lv,hoe,tank,[0,3]}$	$Lognormal(-0.166, 0.36)$	0	3	0.868
$out_{lv,hoe,tank,[3,72]}$	$Lognormal(1.807, 0.299)$	3	72	0.131
$out_{jup,lv,trail,[0,1]}$	$Lognormal(-0.775, 0.664)$	0	1	0.040
$out_{jup,lv,trail,[1.3]}$	$Lognormal(0.356, 0.204)$	1	3	0.461
$out_{jup,lv,trail,[3,72]}$	$Lognormal(2.737, 0.809)$	3	72	0.434
$out_{jup,lv,trail,[72,120]}$	$Lognormal(4.52, 0.152)$	72	120	0.063
$out_{jup,jup,trail,[0,1]}$	$Lognormal(-3.13, 1.967)$	0	1	0.361
$out_{jup,jup,trail,[1,12]}$	$N(6.155, 2.37)$	1	12	0.437
$out_{jup,jup,trail,[12,72]}$	$Lognormal(3.282, 0.48)$	12	72	0.161
$out_{jup,jup,trail,[72,120]}$	$Lognormal(4.519, 0..15)$	72	120	0.039
$out_{jup,hoe,trail,[0,3]}$	$Gamma(5.675, 5.25)$	0	3	0.479
$out_{jup,hoe,trail,[3,15]}$	$Lognormal(2.038, 0.374)$	3	15	0.176
$out_{jup,hoe,trail,[15,72]}$	$Lognormal(3.404, 0.471)$	15	72	0.267
$out_{jup,hoe,trail,[72,120]}$	$Lognormal(4.52, 0.147)$	72	120	0.076
$out_{jup,lv,tank,[0,3]}$	$Gamma(9.984, 6.968)$	0	3	0.663
$out_{jup,lv,tank,[3,10]}$	$Lognormal(1.626, 0.186)$	3	10	0.248
$out_{jup,lv,tank,[10,72]}$	$Lognormal(3.074, 0.576)$	10	72	0.079
$out_{jup,lv,tank,[72,120]}$	$Lognormal(4.431, 0.125)$	72	120	0.008
$out_{jup,jup,tank,[0,3]}$	$Lognormal(-1.102, 0.733)$	0	3	0.87
$out_{jup,jup,tank,[3,10]}$	$Lognormal(1.712, 0.205)$	3	10	0.09
$out_{jup,jup,tank,[10,72]}$	$Lognormal(3.015, 0.496)$	10	72	0.04
$out_{jup,hoe,tank,[0,2]}$	$N(1.023, 0.277)$	0	2	0.787
$out_{jup,hoe,tank,[2,72]}$	$Lognormal(2.224, 0.799)$	2	72	0.212
$out_{hoe,lv,trail,[0,2.5]}$	$Gamma(6.878, 7.278)$	0	2.5	0.341
$out_{hoe,lv,trail,[2.5,13]}$	$Gamma(9.673, 1.227)$	2.5	13	0.316
$out_{hoe,lv,trail,[13,72]}$	$Lognormal(3.154, 0.44)$	13	72	0.305
$out_{hoe,lv,trail,[72,120]}$	$Lognormal(4.502, 0.143)$	72	120	0.036
$out_{hoe,jup,trail,[0,3.5]}$	$Lognormal(0.327, 0.315)$	0	3.5	0.460
$out_{hoe,jup,trail,[3.5,14.5]}$	$Lognormal(2.056, 0.375)$	3.5	14.5	0.163
$out_{hoe,jup,trail,[14.5,72]}$	$Lognormal(3.357, 0.479)$	14.5	72	0.280
$out_{hoe,jup,trail,[72,120]}$	$Lognormal(4.536, 0.149)$	72	120	0.095
$out_{hoe,hoe,trail,[0,120]}$	0^*	0	120	1
$out_{hoe,lv,tank,[0,1.5]}$	$Lognormal(-0.147, 0.277)$	0	1.5	0.767
$out_{hoe,lv,tank,[1.5,72]}$	$Lognormal(1.653, 0.817)$	1.5	72	0.232
$out_{hoe,jup,tank,[0,72]}$	$Lognormal(0.46, 0.669)$	0	72	1
$out_{hoe,hoe,tank,[0,3]}$	$Lognormal(0.417, 0.638)$	0	3	0.018
$out_{hoe,hoe,tank,[3,14.5]}$	$Lognormal(1.783, 0.316)$	3	14.5	0.395
$out_{hoe,hoe,tank,[14.5,72]}$	$Lognormal(3.415, 0.388)$	14.5	72	0.481
$out_{hoe,hoe,tank,[72,120]}$	$Lognormal(4.531, 0.138)$	72	120	0.104

3.4.7 Wait time before departure

Between processing an order and the departure of this order there is a delay defined in this report as the wait time before departure. There are two main reasons for orders to wait before departure. Firstly, most of the loading done is direct loading, meaning that produced goods are directly loaded after packaging to save unnecessary warehouse operations. Secondly, orders are loaded in advance to ensure availability of transport orders. Because of this it often happens that trailers

get placed multiple hours on the parking before leaving. Owned trailers generally wait longer before departure as, external trailers are often used in live-loading and penalties apply for waiting too long. It was assumed that the wait time was also different for different vehicle types.

In Table 3.9 the wait time distributions are shown. The notation used is similar to the one used for processing times, namely $wait_{loc,vt,own}$.

Table 3.9: Wait before departure times

Parameter	Distribution	Mean	Variance
$wait_{lv,trail,yes}$	$Gamma(0.577, 0.065)$	8.864	136.172
$wait_{lv,trail,no}$	$Gamma(0.368, 0.133)$	2.763	20.747
$wait_{lv,tank,yes}$	$Gamma(0.731, 0.074)$	9.792	131.091
$wait_{lv,tank,no}$	$Gamma(0.375, 0.239)$	1.569	6.555
$wait_{jup,trail,yes}$	$Gamma(0.237, 0.032)$	7.324	226.051
$wait_{jup,trail,no}$	$Lognormal(0.291, 0.255)$	1.382	0.129
$wait_{jup,tank,yes}$	$Gamma(0.427, 0.081)$	5.276	65.142
$wait_{jup,tank,no}$	$Gamma(2.115, 8.916)$	0.237	0.026
$wait_{hoe,trail,yes}$	$Gamma(0.113, 0.020)$	5.600	275.91
$wait_{hoe,trail,no}$	$Gamma(0.573, 1.065)$	0.538	0.505
$wait_{hoe,tank,yes}$	$Gamma(0.476, 0.117)$	4.073	34.816

3.4.8 Transition probabilities

In this section the transition probabilities are derived for each vehicle type for each plant. Correct transition probabilities are crucial in maintaining a sufficient trailer balance all over the network. Even slight biases can result in serious shortages and overflows at sites. The transition probability was derived from the data as the percentage of shipments of vehicle type v leaving from plant i of which the next arrival would be at j. This percentage effectively equals the transition probability $Pr(j|i) = P_{i,j}$.

In order to achieve network balance, the arrival rate λ_i should equal departure rate μ_i for each of the sites i. The arrival rate at a site is a function of the arrival rates of external ($\lambda_{i,ext}$) and owned trailers ($\lambda_{i,int}$). Similarly, the departure rate is a function of the departures rates of external $\mu_{i,ext}$ and owned trailers $\mu_{i,int}$.

The arrival rate at every plant can be derived by the transition probability and the departure rate which can be denoted as $\lambda_{i,int} = \sum_{j=1}^{n} P_{j,i}\mu_{j,int}$ with n equals the number of sites.

The difference between the owned arrival rate and departure rate for each vehicle was validated using the script described in Appendix F in Listing 6. Initially, there was a slight difference for the measured transition probabilities. Therefore, probabilities were adjusted to minimise the difference as much as possible.

The corrected transition probabilities are displayed per vehicle type in Tables 3.10.

Table 3.10: Transition probabilities

		Destination					
		Trailers			Tankers		
		Leuven	Jupille	Hoegaarden	Leuven	Jupille	Hoegaarden
Origin	Leuven	0.710	0.217	0.073	0.710	0.128	0.162
	Jupille	0.239	0.697	0.065	0.873	0.005	0.122
	Hoegaarden	0.190	0.104	0.706	0.643	0.088	0.269

3.4.9 Vehicle reallocation

Even though the transition probabilities have been corrected it is important to note that this results in a theoretically balanced network over a long period of time. It is still possible that single simulation runs are imbalanced due to the effect of sampling. In reality these imbalances also occur but are subsequently balanced by manual interventions aiming to prevent trailer unavailability. When too many trailers stock up at one location, or a significant (future) gap in need and availability is identified, trailers will be moved from one location to the other. There is however no standardised process in place for this, making it difficult to simulate this. However due to the high variability in the processes and probabilistic sampling for defining the routing, it was found that the model was unstable without any control process.

It was therefore decided to add a re-balance process that when the amount of trailers at one plant exceeds a certain threshold, the trailer is sent to the plant with the highest need for trailers. The need is defined as the number of open calls minus the actual trailers available on-site. The transferring of one trailer to another site takes approximately one hour. This process resembles the manual actions taken when there are shortages and overcapacity at individual plants. One challenge of this methodology is that it gives an additional parameter that needs to be defined, namely the threshold for the maximum amount of available vehicles at a plant before they get redistributed to other plants. As this process is not standardised, no real threshold like this exist. Instead it was discussed what would be a reasonable number to set as a maximum. To do so the maximum numbers of shipments per vehicle type and plant in one shift were derived from the data. As this gives an idea of the maximum required trailers over an 8-hour period. Seeing as trailers, both external and owned trailers continuously arrive, this value is much higher than what would logically be needed. It was therefore decided to take 75% of the maximum amount of shipments as the threshold for trailers. For tankers it was decided to make the thresholds the same as the expected ratios as tankers are often idle for long times at the different plants and lowering this number would result in many unnecessary repositionings. This results in the thresholds as shown in Table 3.11.

Table 3.11: Repositioning thresholds

Plant	Vehicle Type	Threshold
Leuven	Trailer	96
Leuven	Tanker	37
Jupille	Trailer	72
Jupille	Tanker	12
Hoegaarden	Trailer	47
Hoegaarden	Tanker	8

3.4.10 Defects

In the current situation it is estimated that 5-10% of the owned fleet is unavailable due to defects, repairs and maintenance Schrover (2017). In an earlier study, which measured the actual percentage of defects over the course of a month, a percentage of 6.98% of the fleet being defect was found (Richelle, 2017). Therefore, it is important to take into account defects when optimising the trailer fleet. Unfortunately, no data was available on how long trailers generally are defect. The only historical information available is the amount of defects per month, and the aforementioned percentage. This makes it difficult to fully model the defects process. The defect process instead is modelled as a probability P_d that a trailer will be delayed with time t_d. Probability P_d is estimated by dividing the average number of defects per year by the average number of shipments per year. This gives the probability that a defect occurs per shipment. It is assumed that the probability of a defect is unrelated to the vehicle type and the location. The time delay t_d is estimated by calculating the effective unavailability time per year and dividing this by the total

number of defects. As on average 6.98% of the fleet is unavailable this means that every vehicle is on average $8760 * 6.98\% = 611.448$ hours per year defect. With 668 defects per year on 355 trailers this results in 1.88 defects per trailer per year, taking an average time of 12.74 days to resolve.

3.5 Model translation

In this section the designed simulation model is implemented using specific simulation software. The software used to create the simulation model is the open source discrete event simulation software JaamSim (JaamSim Development Team, 2017). This Java-based simulation package offers a customisable simulation tool with a visual user interface, allowing rapid development of simulation models as well as the possibility to animate the simulations. A visual user interface was chosen as this allows for easier debugging as well as stakeholder model acceptance and adoption (Rohrer, 2000). The software has a lightweight executable (< 8MB) that interprets config files (.cfg) and uses unique syntax which allows models to also be created and adapted without the user interface. Furthermore, the software can also be called by other programs, and inputs and outputs of the simulation model can be communicated. The latter has been used to interface the simulation model with an optimisation algorithm written in Python. An elaboration on this can be found in Section 4.3. Simulations can also be performed without the visual user interface which allows for shorter simulation times when running batches of simulations.

The specific JaamSim implementation is elaborated on in Appendix C, however the different parts of the JaamSim implementation are briefly discussed here. The implementation follows the process models as described in Section 3.3.1. The JaamSim implementation adds specific process steps to link the different model parameters. Trailers are created, their types assigned and then sent to an arrival process. In this process, shown in Figure C.1, it is checked whether there is a surplus of trailers at the destination of the trailer. If so, the trailer is send to another site with the highest need for trailers. This need is defined as the number of open calls minus the current number of trailers at the site. Transferring involves a fixed delay of one hour. If there is no surplus the trailer arrives at the parking of its intended destination.

3.6 Model verification and validation

Verification of the model tests whether the implementation of the simulation model is correct and does what it aimed to do in the conceptual model (Sargent, 2005). The Graphical User Interface (GUI) of the simulation software and the ability to animate simulations helped in verifying the simulation implementation as it allowed additions and changes to the simulation logic to be visualised immediately.

Model validation focuses on whether the simulation model actually represents the real system. Seeing that there was no real life data available for many of the output data it was difficult to compare a real scenario with the simulated scenario. (Sargent, 2005) names a couple of validation methods that can be performed when validation data is not widely available. First of all face validity can be achieved by showing the simulation and its results to process owners and experts. As they are knowledgeable of the real processes they should be able to determine whether the simulation and its results are seemingly valid or not. In order to test face validity, one simulation case has been simulated which represented the current situation. The availability of trailers, tankers and parking places at each of the sites was visualised inside the simulation software.

In Figure 3.5 an example is shown of the availability of trailers, generated by the simulation model. In this graph, The Y-axis shows the amount of trailers (at snapshots every hour), either available for processing (at the sites (Leuven (red), Jupille (blue), Hoegaarden (green)) or on the road when Out (black). The X-axis depicts the total simulation time of one simulation run (3 months). In Figure 3.6, a similar figure is shown but then for tankers, instead of trailers.

Looking at Figure 3.5, several phenomena can be seen. First of all there is a clear, periodic drop in the number of trailers that are out. These drops closely resemble the drop in trailer leaving

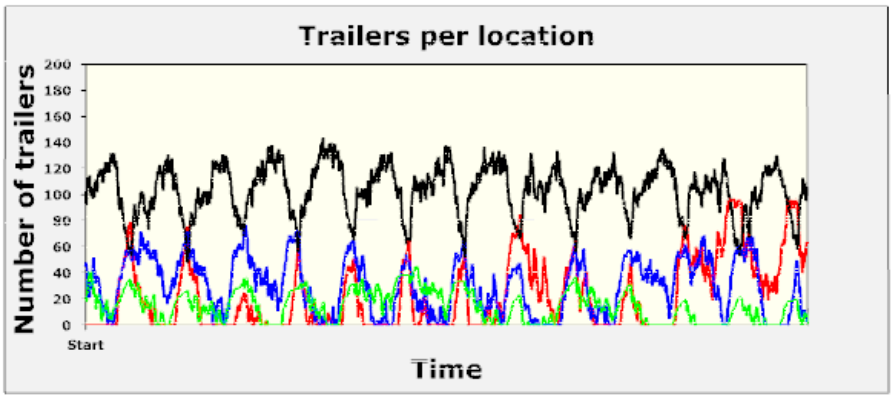

Figure 3.5: Example trailer availability - interval 1 hour - time window 3 months

during weekends, as less loading and shipments are performed during the weekend, a similar rise in the amount of available trailers can be seen during the weekend at the different sites. Whenever, one of the sites has no available trailers, it can be said that there is an availability problem. It can be seen that at each of the sites this happens occasionally.

Figure 3.6 shows similar dips in the amount of vehicles out (depicting the weekends) but in this figure it is not always as clear when a weekend is occurring. The reason for this is that there are normally less tankers out, than for instance trailers. Therefore, there is a smaller difference in the maximum and minimum amount of tankers that are out. Furthermore, this image clearly shows that there is overcapacity of tankers. In reality, tankers rarely give availability problems, and when this occurs it is almost exclusively caused by mismanagement of the tanker fleet (Caignau, 2017).

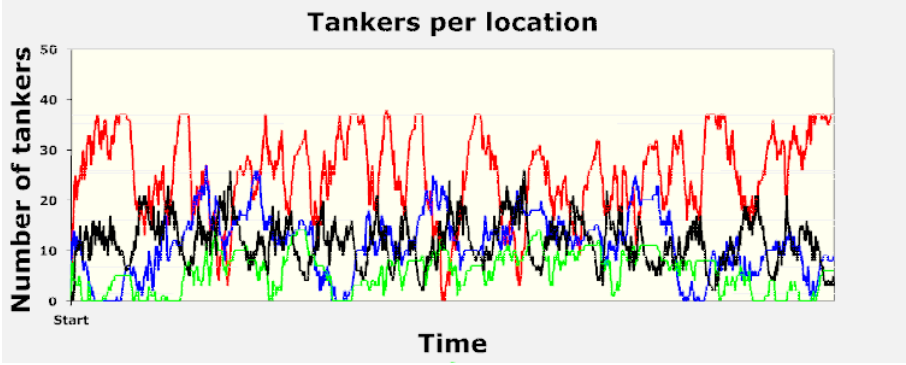

Figure 3.6: Example tanker availability - interval 1 hour - time window 3 months

In Figure 3.7 the parking occupation in the simulation is shown for Leuven (red), Jupille (blue) and Hoegaarden (green). The Y-axis shows the number of parking spots occupied, whilst the X-axis shows the simulation time (of 3 months). The values are the number of occupied parking spots (snapshots every hour). As there is only limited parking capacity available, it can be seen in the graph when full parking lots occur. In this case the amount of used parking spaces stops increasing. It can be seen that for each of the locations, the graph rounds off at different places. Indicating multiple times that the parking is full.

Running the simulations for the situation from April to July 2017. Table 3.12 shows the outputs of the simulation of the current situation. Looking at these results it can be seen that on average 9.1% of all orders get delayed. At first this seemed a bit high, but realising that unavailability of trailers occur sometimes multiple times a week and sometimes for hours on an end, one moment of insufficient trailers can often result in many shipments being delayed. It is interesting to note that the variability of the outcomes, specifically on plant level, are very high. This variability are

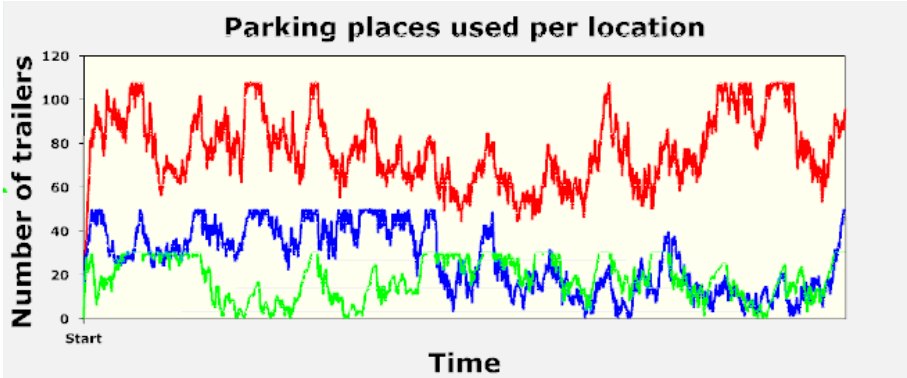

Figure 3.7: Example parking usage - interval 1 hour - time window 3 months

likely caused by the high variability in processing, waiting and out times, but also by the routing of vehicles between plants. However, as the optimisation part will use the aggregated statistics (as the methodology is focused on improving the complete scope rather than specific plants) this is not a very large problem. Regardless, there is still a relative large amount of variability in, specifically, the number of delayed orders.

Table 3.12: Outputs current situation per location

Statistic	Leuven	Jupille	Hoegaarden	Total
Average delayed orders	1,570.6	790.9	723.3	3084.8
St. dev. Delayed orders	847.1	764.0	487.9	1070.0
Total orders	15,509	12,958	5,257	33,724
Percentage of total (delays)	10.1%	6.1%	13.8%	9.1%
Average times parking full	272.6	412.0	537.1	1,221.7
St. dev. times parking full	126.1	111.1	182.1	186.5

With the visualisations of the trailer, tanker and parking availability, combined with showing several simulation runs and explaining what was happening in the simulation model, and the quantified results of the simulation run, face validity was achieved.

Chapter 4

Optimisation

This chapter focuses on the design, implementation and evaluation of the optimisation part of the simulation optimisation approach. This chapter is structured as shown in Figure 4.1.

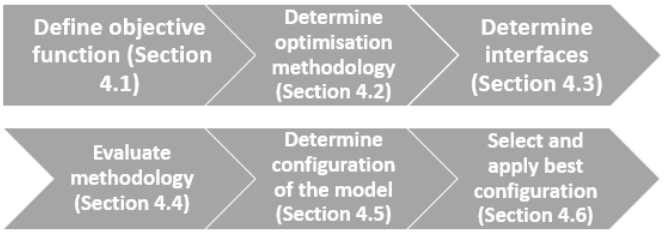

Figure 4.1: Structure of Chapter 4: Optimisation

First the objective function is defined in Section 4.1. This forms the basis of the optimisation model as it defines the true optimisation goal. Next the optimisation methodology is defined and elaborated on in Section 4.2. In Section 4.3 the interfaces between the simulation model and different parts of the optimisation model are defined and the implementation is discussed.

After this, the main parts (meta-model and genetic algorithm) of the optimisation model are evaluated in Section 4.4. In Section 4.5 the impact of the configuration of the optimisation model is tested by testing different configurations and analysing the impact of the configuration on overall optimisation. This chapter is concluded Section 4.6, with the selection and application of the best configuration based on the performed experiments.

4.1 Objective function

In order to optimise any simulation or function it is important that the objective function is well defined and its constraints determined. The trailer management process depends on several factors and each of them should be taken into account when optimising the size of the trailer fleet. The factors that will be focused on in the objective function are:

- Trailer availability
- Trailer parking availability
- Trailer related costs
- Parking related costs

The objective function aims to minimise the trailer and parking related costs while ensuring trailer and trailer parking availability. Trailer related costs are defined as the estimated trailer

costs of external shipments, the rent cost of owned trailers and the costs of defects. Parking related costs include the estimated cost of additional parking places.

Trailer and trailer parking availability can be considered constraints in the optimisation process. Higher unavailability of either trailers or trailer parking results in additional costs in the logistics process by delaying the process, slowing down the process or disrupting other processes (Schrover, 2017). The simulation model measures the time frequency of orders waiting for more than 12 hours to be linked to a trailer. As it can be assumed that linking trailers and orders takes no time, this effectively means that an order could not start processing due to lack of availability in trailers.

Constrained optimisation problems have been approached in a variety of ways. Coello (2002) made an extensive review on constraint handling in genetic algorithms. These techniques often involve some way to represent the constraints in an unconstrained domain (such as penalty functions), use repair algorithms to gain feasible solutions from infeasible ones or separating constraints and objectives and using multi-objective techniques to solve them. In this research, a penalty function methodology has been used. The reasoning for this is that the constraints (missed orders and parking shortages) are both constraints on the outputs of the simulation model and they are not easily repaired, as this would require either approximation of the outputs (reducing accuracy of the methodology) or running new simulations (which increases simulation time). The general idea of penalty functions is that the degree of infeasibility of a solution is important information that can be used for optimisation. The penalty function has a very big impact on the optimisation process as it implicitly reflects the importance of different constraints on the quality of a solution. Penalty functions however come with the difficulty that real life penalties are not always easily defined. In the case of the trailer management process there has never been a full study on the costs of a missed order or the costs related to having an overfull parking lot. In many cases, it can even be said that there are no direct costs as many of the cases either have no direct consequences due to mitigation of problems by human intervention or the consequences are difficult to relate to this problem. Both scenarios have, however, been identified as work-flow disrupting, making them highly unfavourable. For this research, the problem has been defined as an unconstrained optimisation problem. A relatively simple, linear penalty function was implemented as a cost term in the objective function. This penalty function is described in Equation 4.1. In this function MC_v denotes the number of missed calls (or delays) for vehicle type v, c_{mc} the estimated cost for a missed call, PF_v the number of occurrences that the parking is full and c_{pf} the cost of one occurrence of the parking being full.

$$c_{penalty} = \sum_{v=1}^{2} MC_v * c_{mc} + \sum_{v=1}^{2} PF_v * c_{pf} \tag{4.1}$$

Equation 4.2 shows the complete cost function $f(x)$ for individual x. A solution x consists of values for the amount of owned vehicles at each plant, the multiplication factor for external vehicles at each plant and the amount of parking capacity. The objective function aims to minimise $f(x)$. In this cost function, $r_{veh,v}$ is the total number of vehicle resources (trailers or tankers) of type v, where 1 equals trailers and 2 equals tankers. $c_{own,v}$ equals the cost of an owned trailer of type v. $S_{ext,v}$ equals the total number of shipments on external vehicles of type v and $c_{ext,v}$ is the cost of a single shipment on an external vehicle of type v. The parameter $r_{park,l}$ equals the amount of parking spots at site l and c_{park} equals the cost per parking spot. The total costs of defects are determined by the number of defects D and the estimated cost per defect c_d. Finally, the penalty cost as derived in Equation 4.1 is added to the cost function.

$$f(x) = \sum_{v=1}^{2}(r_{veh,v} * c_{own,v} + S_{ext,v} * c_{ext,v}) + \sum_{l=1}^{3} r_{park,l} * c_{park} + D * c_d + c_{penalty} \tag{4.2}$$

It is important to note that the cost function incorporates both inputs and outputs of the simulation model. The number of vehicles and parking places are direct inputs in the simulation

model whilst the number of defects, the number of shipments performed on external vehicles and the penalty costs are dependent on the outcomes of the simulation.

The cost of a defect was derived from the expected amount of defects per year (668) and an estimated yearly cost of 350,000 euro (Schrover, 2017). This gives a cost of circa 500 euro per defect. The cost of a trailer was approximated to be 8,200 euro per year (between 600 and 900 euro per month based on the type of trailer) (van Hulst, 2017). This results in a three month cost of 2,050 euro. The trailer cost of shipments performed on both trailers and tankers, were approximated to be 40 euros per shipment (van den Hoogenhoff, 2017). No data was readily available on the monthly cost of a tanker and as their availability gives less problems than trailers, an approximation was made that was slightly lower than those of trailers to reduce their priority in the optimisation process. The cost was approximated to be 1,500 euros per 3 months. It was difficult to define the cost of one delay as in many cases no real costs occurs. When costs occur they can be in the hundreds of euros when penalties apply. Worst case an entire shipment gets cancelled by the customer (as they no longer need the shipment) which would result in potential lost sales of thousands of euros. Fortunately, the latter only occurs extremely rarely. When estimating the cost of a single delay, taking into account the many cases were it does not cost anything, it was estimated, together with stakeholders, that a single delay would cost around 200 euros. On the contrary, having a full parking has not resulted in direct costs. It was however stressed that this problem was to be avoided as much as possible in the future so a fictional cost was added. The estimated cost of a full parking was defined to be 200 euros to have a similar importance as delays. In Table 4.1 the estimated values for the cost parameters are given.

Table 4.1: Cost parameters for the objective function

Parameter	Notation	Value
Cost of owned trailer	$c_{own,1}$	2050 euro
Cost of owned tanker	$c_{own,2}$	1500 euro
Cost of shipment on external trailer	$c_{ext,1}$	40 euro
Cost of shipment on external tanker	$c_{ext,2}$	40 euro
Cost of parking	c_{park}	1500 euro
Cost of defect	c_d	500 euro
Cost of delayed order	c_{do}	200 euro
Cost of parking full	c_{pf}	200 euro

4.2 Simulation optimisation methodology

In this section the used simulation optimisation methodology is described by explaining the methodologies used and the different concepts applied. The general simulation optimisation algorithm works similarly as the general simulation optimisation algorithm using a meta-model filter as described by April et al. (2003). It is an online simulation optimisation algorithm as it continuously uses the outputs of the simulation evaluations to improve the meta-model used in approximation of the fitness while running the algorithm. The main approach is outlined in Figure 4.2.

The gray box outlines the part of the algorithm that will continuously be performed until one of the stopping criteria (see Section 4.2.3) is reached.

The used approach in this research, extends the general simulation optimisation setup described in April et al. (2003) with an additional probability, called the ensure probability, that may prevent discarding solutions even though their approximated fitness is not good enough. It is important to keep some of these solutions to increase variety in the population and reduce the impact of approximation errors.

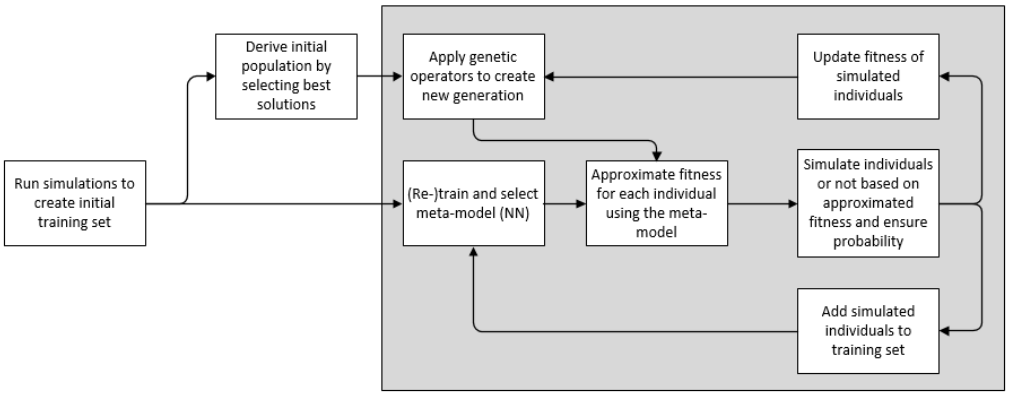

Figure 4.2: Simulation optimisation setup

4.2.1 Optimisation approach

In short the simulation optimisation algorithm works as follows: a random set of simulations is performed and the inputs and outputs are used to train a global meta-model on the solution space. Subsequently, the initial population of the genetic algorithm is created by selecting the random simulations with the best fitness. Cross-over (point-wise), elitism and mutation probabilities are applied.

A genetic algorithm is used as meta-heuristic optimiser that generates a population of individuals. Each individual represents a configuration of the simulation model, where each of the chromosomes of the individual is one of the decision variables in the simulation model (trailers Leuven, tankers Leuven, trailers Jupille, etcetera). In Appendix E an example is more specific example is given of an individual, together with a description of each of the chromosomes.

The fitness of each individual x is subsequently approximated using the trained global meta-model. This gives approximations $\hat{f}(x)$ for every individual x. These approximations are checked against the best actual simulated solution found. The actual simulated solution for an individual x, can be denoted as $f(x)$. As the objective function aims to minimise the value of $f(x)$, the approximated solutions $\hat{f}(x)$ are compared to the best solution so far $min(f(x))$. When the approximation of the solution is smaller than or equal to the best solution so far plus a pre-defined threshold value d, i.e., $\hat{f}(x) \leq min(f(x)) + d$, the individual is simulated to get the real output $f(x)$. Additionally, there is a probability of p_{ensure} that an individual will be simulated regardless of the estimation. This is to ensure global convergence of the algorithm and is further discussed in Section 4.5.4. Whenever a solution is simulated, its inputs and outputs are stored so that they can be used in future retraining of the meta-model.

4.2.2 Meta-modelling technique

As mentioned earlier in Chapter 2 a wide variety of techniques have been used and combined for simulation optimisation. In this research it was however decided to focus on one kind of approximator: feed forward neural networks (FNN).

FNN have been used multiple times in simulation optimisation due to its capabilities as a universal approximator of measurable numbers Hornik et al. (1989). The effectiveness of a FNN is dependent on the degree of training, number of hidden layers and most importantly on whether there is a definable relationship between inputs and outputs. Simulation optimisation problems often have less data points than traditional optimisation problems due to the costliness of evaluations and the dependency between inputs and outputs does not only depend on the optimisation problem but also on the degree of variability in the simulation outputs.

A downside of FNNs is that it is a black box approximator, giving no direct insight in the importance of individual inputs in defining the outputs. Another disadvantage is that (re-)training

the model may be time consuming, especially when complexity of the model increases.

The meta-model uses input values (x) of the simulation model (an individual in the genetic algorithm) to predict the value of the cost function $f(x)$ defined in Section 4.1. Before training the input values were normalised to allow easier training of the meta-model. The subsequent approximations of individual x give approximations $\hat{f}(x)$.

For FFNs it is crucial to use the right parametrisation in order to create effective approximation models. To achieve the best parametrisation, every time the model is trained, a grid search approach using K-Fold Cross Validation was used to determine the best parametrisation. Grid search tries all different combinations of parameters using the cross-validation set-up and will return the best performing network. Each of the neural networks were trained using the rectified linear unit (ReLU) activation function. This network in course is trained on the full training set (the 80% of the total data-set) and tested on the validation set that was holdout. Subsequently, the MAE, R^2 and MSE statistics are calculated and stored.

The parameters explored in the grid search are the number and size of hidden layers and the L2 regularisation rate (alpha). The explored parameter values are shown in Table 4.2.

Table 4.2: Grid search parameters FNN

Parameter	Values
Alpha (L2 regularisation)	[0.05, 0.01, 0.005, 0.0001, 0.00001]
Hidden layers	[(15), (25), (50), (50,5), (50,25,15), (25,15),]

In Section 4.4.1, it is tested whether a sufficiently good global meta-model of the simulation model can be trained using FNNs. Furthermore, the required training data set size will be determined.

4.2.3 Stopping criteria

There are multiple ways to define whether a genetic algorithm converges to a local optimum. In many cases stopping criteria have been used such as a maximum number of generations or a certain number of generations in which the best solution did not improve. There is not always a one solution fits all for the stopping criterion and as different models are compared it was decided to use a two rule stopping criterion. The optimisation stops when either 10 generations of the GA have been performed or when 300 simulations have been done. This criterion was chosen to add a bound to the total available time for optimisation as each generation may result in a number of simulation evaluations equal to the population size. Furthermore, each new generation meta-model retraining and selection occurs. Aside from that every simulation takes significant time as well. Alternatively, running time (using processor time) was considered to be used as a stopping criteria of the algorithm but due to the simulation being performed using a different program this gave some implementation issues.

4.3 Implementation of interfaces

In order to effectively apply simulation optimisation techniques the interfaces between the different algorithms need to be identified and implemented.

The three main components in the used simulation optimisation methodology consists of the optimisation model (genetic algorithm), the approximation model (meta-model) and the simulation model. Both internal and external interfaces are used to communicate between these components.

The simulation model consists for a part of a static `.cfg-file` in which all of the simulation logic is stored as text. This `.cfg-file` includes the files `arrivals.inc` and `transitions.inc`. These files include simulation specific information (arrival and transition rates) and are stored seperately to ensure integrity of the config file. The include files can be generated by using a

Python script, that reads the required data from the excel-file `MasterData.xlsx` and translates this into the expected arrival rates and transitions. This set-up allows, after generating the required files, to simulate manually using the JaamSim executable or `JaamSim.jar` files.

In Python a function called `RunCreator` was created to translate the genes of the optimisation model into input values for the JaamSim-model. The optimisation model calls the simulation model whenever it requires simulations of one of its individuals. For this it uses the generated input files. The simulation model returns outputs in text form. In order to use this an interpretation function was written `procOutputs` which aggregates the individual runs and calculates the mean and standard deviation of the outputs. The outputs generated by the simulation model are defined in the `.config-file` of the simulation model. The code creating the interfaces between the simulation model and optimisation algorithm is can be found in Appendix F.

In Figure 4.3 the component structure diagram is shown for the optimisation program. The component structure diagram is modelled according to the specifications of UML 2. In the component structure diagram the different interfaces between the components are shown.

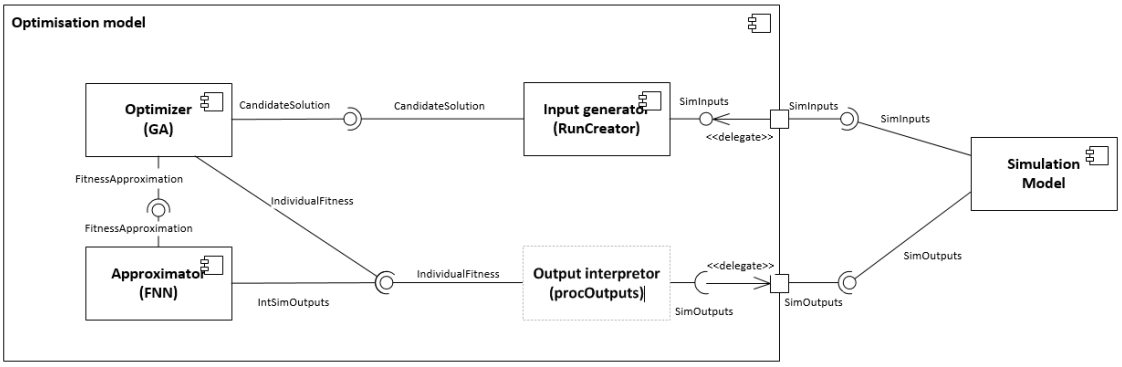

Figure 4.3: Component structure diagram for the simulation optimisation program

The simulation optimisation model was created in Python version 3.6. The commented source code of the simulation optimisation algorithm can be found in Appendix G. The implementation uses the Distributed Evolutionary Algorithms in Python (DEAP) library (Fortin et al., 2012) for the implementation of the genetic algorithm. Furthermore, the scikit-learn library (Pedregosa et al., 2011) is used in the creation and usage of the meta-models such as neural networks and decision tree regressors. Other libraries used are `Numpy` and `pandas`, which have mostly been used to ensure compliance of data types between the different interfaces and easy importing and exporting of solutions to comma-separated values (CSV) formatting. The interfacing of the different algorithms is explained in Section 4.3.

4.4 Evaluation of the optimisation methodology (NN and GA)

The goal of simulation optimisation is to find the simulation configuration that gives the optimal solution to a given problem in a timely manner. As mentioned in the theoretical background of this report, Bowden and Hall (1998) described the different facets that are important in creating good simulation optimisation models. The effectiveness of the simulation optimisation is directly dependent on factors such as problem formulation and the methods used. It is therefore important to consider both the correctness of the simulation methodology and the means for optimisation, such as the objective function and the optimisation algorithm. This does however not mean that the performance of the different models cannot be evaluated independently. The simulations model performance has been discussed in Chapter 3 and the model has been verified and validated. In order to allow easier evaluation of the optimisation algorithm used, they are regarded as two

individual parts. One part is the meta-model, which is used for the fitness approximations and the second part is the meta-heuristic search algorithm. The algorithms effectiveness is defined in its ability of finding the true global optimum, subject to the time it requires to do so.

This section is structured as follows: first the performance of the meta-model is determined and the required amount of initial training data determined. It is important to note, that as the meta-model is retrained after ever generation, that there is no single best meta-model, but instead a methodology to derive meta-models.

4.4.1 Evaluation of the meta-modelling technique

The performance of the simulation optimisation algorithm is directly dependent on the performance of the meta-model. As it uses this meta-model to define which individuals to simulate or not. To evaluate the performance of the meta-model, two criteria were focused on: the accuracy of the approximations and the required amount of training data. The accuracy is important as inaccurate approximations may lead to unnecessary rejections of better solutions or simulations of inferior solutions. On the other hand, the required amount of training data is important as this gives an immediate need of (random) costly simulations to be performed (to generate training data) before the optimisation even starts. In this case, the optimisation approach would be unsuitable when very large amounts of data would be required to train the initial global meta-model, in order to accurately approximate the fitness score of solutions.

The meta-model uses the simulation input parameters as the features to predict the fitness score of the solution (the target value). As the required amount of training data is to be determined a set-up was used with a variable amount of training data per experiment.

In order to test the meta-model performance, first a data set was generated which was used for training and testing the meta-model. This was done by running simulations with random input values (conforming to the feasible parameter value defined in Appendix B). The generated dataset consisted of 2000 random simulations which translates to approximately 5 hours of constantly running simulations (as one simulation takes approximately 22 seconds).

Next, in order to derive the required size of the initial training dataset, a set-up was created that takes increasingly larger parts (multiples of 50 samples) of the original generated data-set (of size 2000) and uses these datasets to select, train and evaluate the meta-model. This resulted in a setup with 50 cases, 100 cases, 150 cases, etcetera. In total this gave 40 different experiments. By using different data-set sizes, the impact of the amount of training data can be seen on the meta-models approximation abilities.

For each of the experiments performed a holdout set of 20 % of the total dataset was used to validate the selected model. The other 80 % was used in the grid search setup (with parameters from Table 4.2 in Section 4.2.2) using K-Fold cross validation to select the optimal model. For each experiment the cross validation used 10 folds and the model was selected based on the R^2 statistics. An example neural network that could be selected subsequently has 14 input nodes (for the simulation inputs), two hidden layers with respectively 50 and 5 nodes with a learning rate (alpha) of 0.0001. The activation function used was the ReLU activation function.

The approximation ability was quantified using three different performance measures: the coefficient of determination (R^2), the mean absolute error (MAE) and root mean squared error (RSME). The coefficient of determination is used to determine the amount of variance in the predicted value that can be explained by the input variables. Next to this, it is important to know how much the predicted values differ from the real values, as wrongly predicted values could result in skipping optimal solutions, effectively making convergence to the optimal solution impossible. Both MAE and RMSE look at the differences between predicted values and real values and take into account both negative and positive differences. The main difference between MAE and RMSE is that RMSE uses the mean squared error calculation. Because of this, larger deviations are more penalised than smaller deviations. In the execution of the simulation optimisation algorithm a part of the deviation is acceptable due to the threshold. However larger deviations give a higher chance that good solutions are rejected (and bad solutions are accepted). Even though each of these performance measures was calculated and shown in the following figures, it is important to

remember that the meta-model selected for each of the training set sizes was selected based on its R^2 statistic in the grid search cross-validation setup described earlier.

With the defined set-up the required amount of training data was determined by plotting the meta-model accuracy statistics (R^2, MAE and MSE) for different total data-set sizes n. The plots for each of the performance measure are shown in Figure 4.4. On the x-axes, the different training set sizes are shown. On the y-axes, the respective performance measures are shown of the best selected meta-model (based on R^2). To clarify, each of the performance measures is shown with a different colors. R^2 is shown in blue, MAE is shown in green and MSE is shown in red.

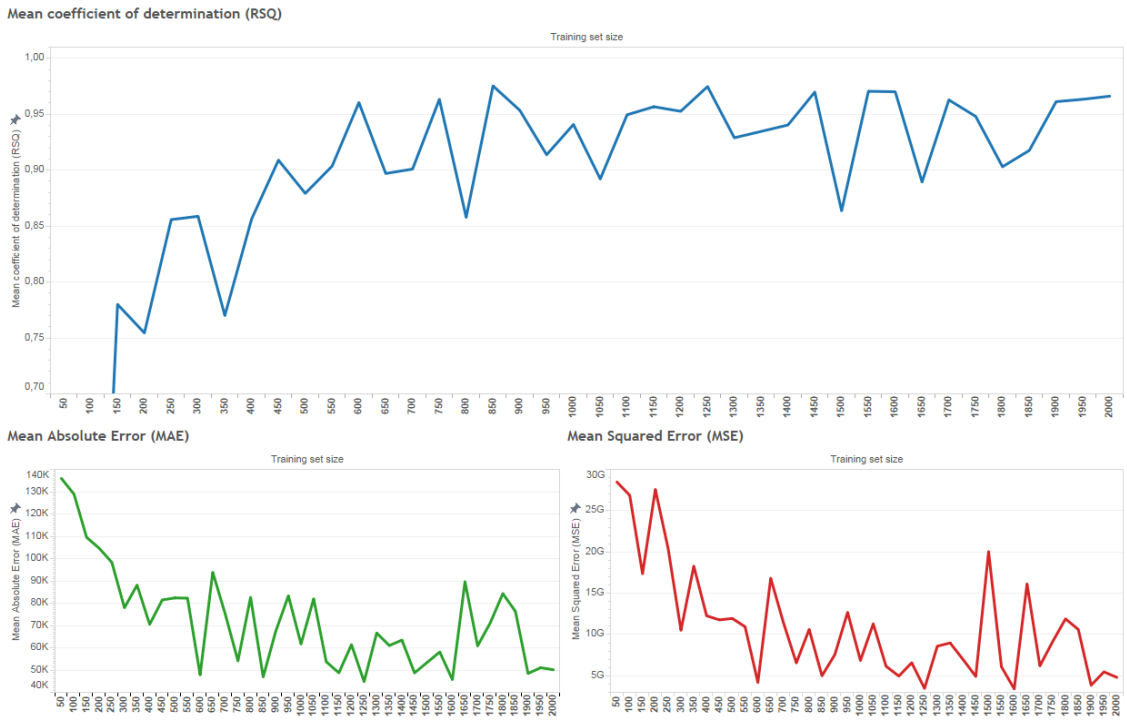

Figure 4.4: Meta-model performance under different training set sizes

Based on the graphs in Figure 4.4, several conclusions can be made. First of all, the meta-model seems to be able to get close approximations of the input-output relations of the simulation model. As with sufficient training data (more than 500 samples) most meta-models surpassed an R^2 of 0.9 and having a mean absolute error of less than $90,000$ when applied on the testing set. With average fitness scores ranging between $1,300,000$ and $2,000,000$ this would result in errors being less than 10% of the fitness values (and in some cases even less than 5%). Seeing as these are approximations used to determine whether a simulation is rejected or accepted, this accuracy is likely good enough. If however, the fitness was only determined by the approximations done by the meta-model, one could argue that a higher accuracy would be needed. It is interesting to note that having more training data, did not always result in better meta-model performance of the meta-model selected. In general a slight increase overall increase can be seen with more cases, but actual performances is quite variable as can be seen in Figure 4.4. Seeing as it is time-intensive to gather more test cases it is likely not worth it to simulate many more cases for the initial meta-model as the meta-model accuracy likely does not increase a lot more. The MAE could form a good basis for the threshold of the optimisation algorithm. It can be seen that when the training set size is sufficiently large (500 training cases or more) the mean absolute error is generally less than 100,000.

In Table 4.3 the aggregated performance statistics of the meta-models with training set sizes ranging between 500 and 2000 are shown.

Table 4.3: Aggregate meta-model performance statistics (training set sizes 500 to 2000)

	R^2	MAE	MSE
Mean	0.934	64,405.127	8.524E+09
St. Dev.	0.034	14,717.022	4.074E+09

4.4.2 Genetic algorithm performance

It was not possible to fully test the genetic algorithms performance individually, as had been done for the meta-model. As a genetic algorithm can very quickly go through very many different configurations (depending on its evaluation function) it would require many different cases in order to ensure that the algorithm performs adequately. Nevertheless, the performance of a genetic algorithm is based on its capability to find the optimal solution in the least time possible. Important measures to consider for this are the convergence to the global optimum property and the speed in doing so (defined as the required evaluation cost in doing so). These measurements are taken into account in the next section, where different parametrisations are compared to find an optimal configuration of the overall optimisation model.

4.5 Configuration of the optimisation model

Now that the objective function and optimisation methodology have been defined it is important to determine the best configuration of the optimisation model.

4.5.1 Experimental set-up

The impact of the different parameters on the optimisation performance is tested by using the same set of parameters for each experiment except for the explored parameter value. The base case is designed to use both the threshold and ensure probability mechanics. To improve the comparability of the parametrisation a similar starting state is ensured by using the same initial random training simulations for each experiment. Furthermore, the same selection methodology is used to determine the initial neural network (see Section 4.2.2).

Table 4.4 shows the base case scenario for the experiments performed.

Table 4.4: Base case scenario for experimental set-up

Parameter	Population size	Ensure Probability	Threshold	Mutation Probability	Cross-over Probability	Number of elites
Value	100	0.01	100,000	0.2	0.5	2

The results of the experiments are presented as a figure in which the simulation evaluations are shown with their predicted value (blue), their real simulated value (red) and the best value found so far in the optimisation process (green).

4.5.2 Impact of population size

When using genetic algorithms it is important to consider the population size used. A bigger population allows for more diversity in the gene pool but it also results in slower evolutions, which in the case of this algorithm, results in slower retraining of the simulation model and potentially wasted evaluations on less fit solutions than would be found in a next generation. In order to check the impact of the population size the following set-up was used:

1. Run the initial random simulations to gather training data for the global meta-model as described in Section 4.5.1.
2. Create an initial population of size k consisting of the k individuals with best fitness from the initial random simulations.
3. Find and train the initial meta-model using the methodology described in Section 4.2.2.
4. Apply genetic operators on the population to create a new population.
5. Check for each of these new individuals whether their approximated fitness is lower than the best score found so far plus a threshold value. If so, simulate the individual to get the simulation evaluation, otherwise let the ensure probability decide whether the solution will be simulated or not.
6. At the end of each generation add the simulated individuals with their fitness to the training set and repeat steps 3-6.
7. Continue until one of the stopping criteria has been reached.

Experiments were performed for a population size (k) of 100, 500 and 1000 and the results can be seen in Figures 4.5 to 4.7. In order to interpret these results it is important to know that the x-axis shows each of the simulation evaluations, with the real simulated fitness in red, the approximated (by meta-model) fitness in blue and the best fitness so far in green.

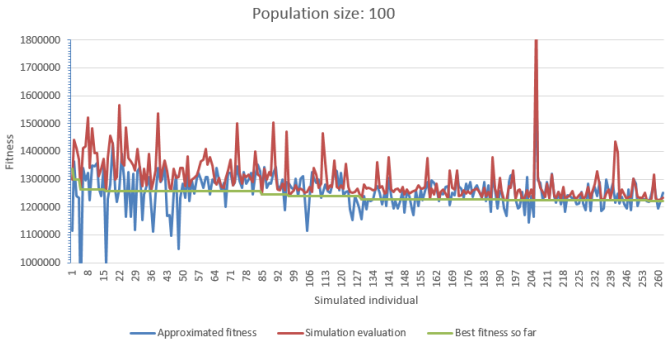

Figure 4.5: Predicted and real simulation values with population size 100

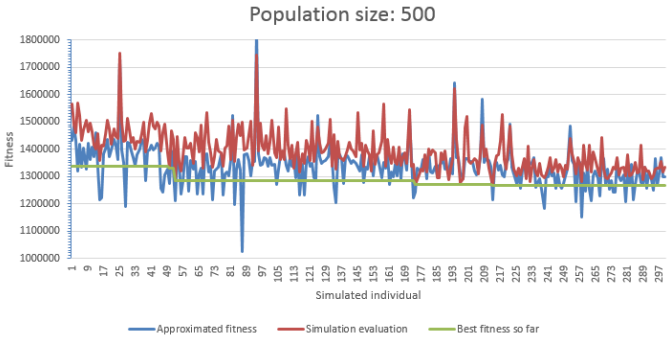

Figure 4.6: Predicted and real simulation values with population size 500

Looking at the situation with a population size of 100 (Figure 4.5) it can be seen that the approximation model quickly finds some better solutions and the first generation ends with 49 simulation evaluations performed. After retraining, the accuracy of the model improves greatly

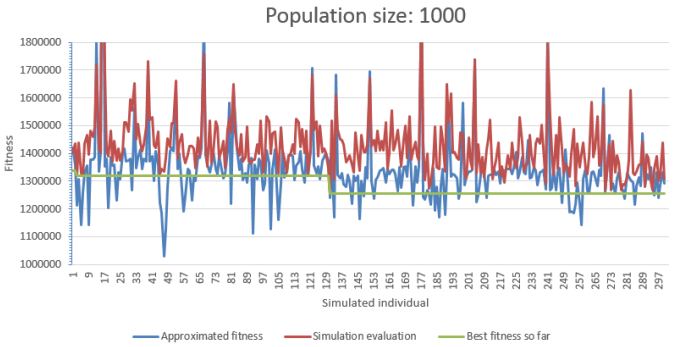

Figure 4.7: Predicted and real simulation values with population size 1000

Figure 4.8: Relation between population size, number of evaluations and best fitness score

after which it more consistently finds solutions that are better or bordering the best solution found by then. This shows the advantage of slightly smaller populations. Smaller populations are also less sensitive for approximation models with less accuracy and can even benefit from this. In the example the initial accuracy is lower but biased on lower predictions than found in the test set. This eventually finds a couple of much better solutions which subsequently used in the training of the next model improve accuracy and find lower solutions. This effect is seen less in the simulation with 500 (Figure 4.6) and 1000 (Figure 4.7) population size. It is important to note that while the original thought of a larger population allows a higher degree of diversity should hold, it is important to consider that diversity does however not always mean a higher population fitness. In this experiment the initial population used are the k individuals with the best fitness from the random experiments. Each experiment starts with the same individuals but due to applied genetic operations and selection, the first evaluated generation may very well be of lower fitness. A smaller population size may also be preferable in this methodology due to the focus on finding better solutions faster. In Figure 4.8 the total number of simulation evaluations compared to the best found fitness are shown for each of the setups. It was found that the setup with the population size of 100 showed the best performance $(1,221,940)$. The peak in Figure 4.5 at evaluation 204 shows that the ensure probability does result in simulation evaluations of solutions with bad fitness approximations. In this case the actual evaluation was also of low quality, hence the peak.

4.5.3 Impact of initial meta-model training on performance of the algorithm

The initial model used in the optimisation method is best described as a global meta-model of the simulation model. Even though exploration mechanics are in place to remove the impact of

the initial population, it is interesting to see the impact of the initial training dataset on the overall performance of the algorithm. In Figures 4.10 to 4.13, the respective performances are seen for situations with an initial training set of 500, 1000, 1500 and 2000 cases. To reduce the impact of random sampling each of the smaller datasets is taken as a subset of the larger training set. Therefore, the dataset for the experiment of 1000 training samples contains all the training samples of the experiment with initial training data set 500 and 500 additional samples. The 1500 initial dataset contains the training data of the 1000 training cases experiment and 500 additional ones, etcetera. An overview of the best solution found plotted against the number of evaluations is shown in Figure 4.9.

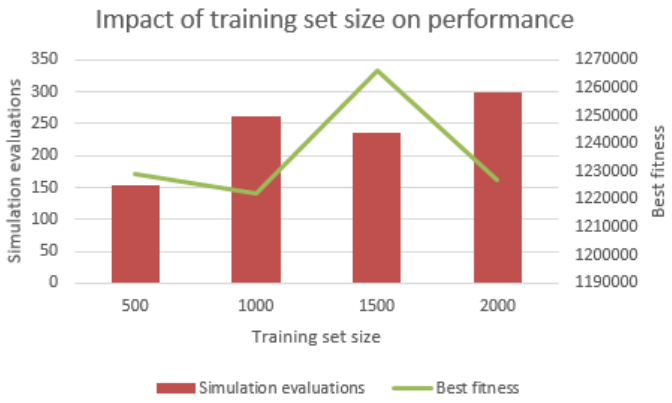

Figure 4.9: Relation between training set size, number of simulation evaluations and best fitness score

It can be seen that there is no clear definable relation between the training size used and the actual performance of the algorithm for the evaluated cases. Two hypotheses could explain this, either the meta-model does not help the optimisation process at all (for example due to its limited accuracy), or the meta-model does help but the effectiveness of the overall algorithm is not very dependent on the actual accuracy of the meta-model, as long as the meta-model is mostly accurate. This second hypothesis comes from what was already seen in Figure 4.4.1 in Section 4.4.1, that the meta-models accuracy nears an R^2 of approximately 90% with a training set of 600 cases and then does not significantly improve any more when more training cases are added.

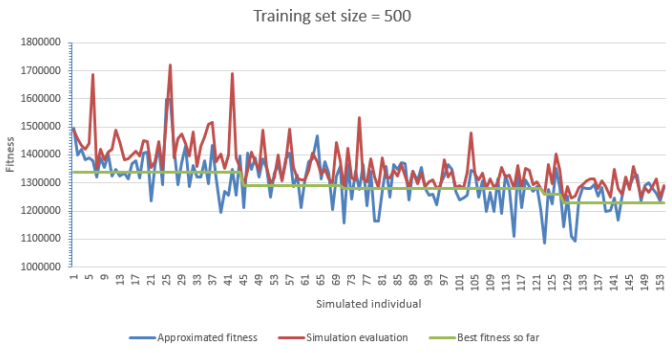

Figure 4.10: Predicted and real simulation values with training set size 500

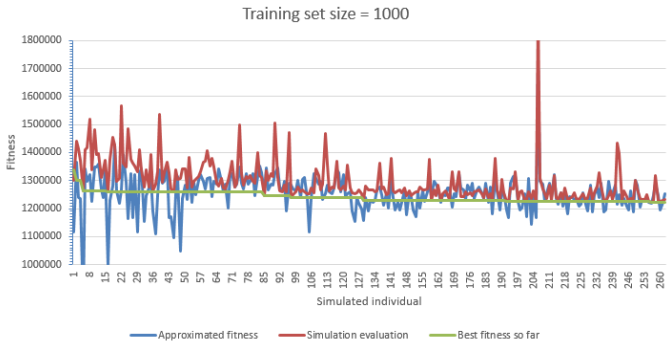

Figure 4.11: Predicted and real simulation values with training set size 1000

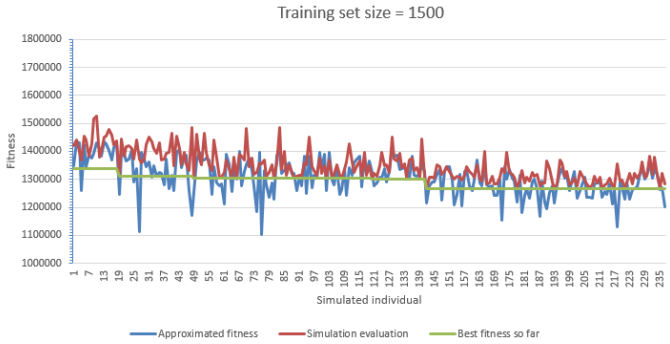

Figure 4.12: Predicted and real simulation values with training set size 1500

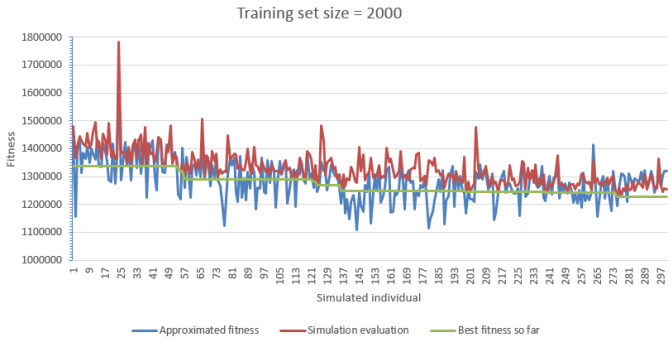

Figure 4.13: Predicted and real simulation values with training set size 2000

4.5.4 Impact of threshold and ensure parameter

In any optimisation study it is important to consider the concepts of exploration and exploitation. Exploration aims to diverge the solution space and escaping local optima. Exploitation aims to exploit possible optimal regions to find the true optima of these regions. In normal optimisation using genetic algorithms, exploration can be achieved using mutation and cross-over operations. However for the used simulation optimisation method these methodologies do not ensure that exploration occurs. The reasoning behind this is that evaluations are only performed on individuals that have a good approximation score. What is considered to be a good enough approximated score is dependable on the best score found so far and a threshold. Allowing the model to revisit and potentially visit every solution is crucial in ensuring global convergence as this requires the possibility of all solutions to be visited.

The threshold parameter is a mechanism to account for the difference between approximated

and real simulation output. As the threshold influences which solutions do or do not get simulated, it is crucial in achieving convergence. Ideally, the threshold should reduce the required amount of simulations, whilst minimally impacting the convergence to the global optimum. As the threshold is used in collaboration with the best fitness score found thus far in the optimisation process, a scenario is created in which solutions with too high approximation scores will never be visited (as long as their approximation is higher than the best score plus the threshold). In order to maintain the global convergence property, an ensure probability was added to ensure simulation of a solution, even when it is initially rejected. As mutation is used to randomly change chromosomes of an individual to other allowed values, there is a probability for each solution to be visited. As only real evaluated scores are stored for individuals, it is possible that a solution with a low (initial) approximation gets visited multiple times by the algorithm. Hence it will be approximated multiple times, which reduces the chance that the solution is a good solution with a bad approximation. Every re-visitation there is also the ensure probability which potentially forces the evaluation of the individual. This makes it that, assuming infinite run time, every potential solution will be visited eventually, which would ensure global convergence.

Even though it is shown that global convergence would hold, it only does so under the assumption of infinite time. However, in reality it is often the case that decision makers rather have good solutions quickly rather than true optimal solutions over a very long time. The proposed concepts of the threshold and ensure probability were tested using the experimental setup defined in Section 4.5.1.

Impact of threshold value

To measure the effect of the threshold on the performance of the algorithm, several experiments were executed with similar configuration, where only the threshold was changed. The threshold levels were determined based on the accuracy of the meta-model which has been defined in Section 4.4.1. The measure used was the mean absolute error (MAE) which ranged between 0 and 150,000. Experiments were performed for a threshold of 0 (meaning no threshold), 50,000 (slightly below MAE (77.6% of MAE)), 100,000 (above MAE (155.3% of MAE)) and 150,000 (much above MAE (232.9% of MAE)). In Figure 4.14 the relations are shown for the proposed thresholds, together with the best found fitness score and the number of simulation evaluations performed.

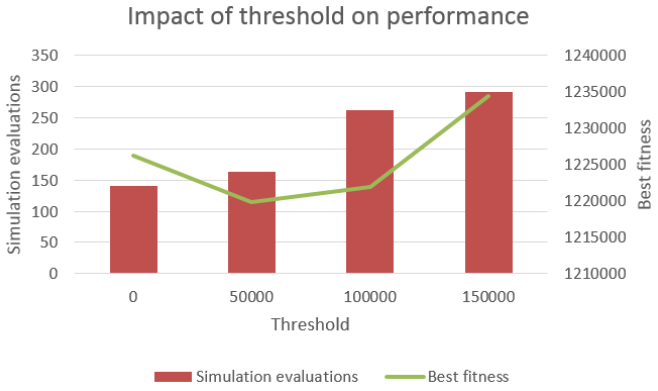

Figure 4.14: Relation between threshold, number of evaluations and best fitness score

When changing the threshold values it can be seen that a lower threshold value may be beneficial in finding better solutions. The best performance is achieved by the threshold of 50,000, but this is only marginally better than the performance of threshold 0 and 100,000. A bigger difference exists between the optimal solution and the best performance for the 150,000 scenario. Therefore, it can be concluded that a more restrictive threshold is better in finding good solutions in less time. Furthermore, no threshold results in even less evaluations but also a slightly worse performance which suggests that using the threshold may improve solution quality. Based

on the experiments it was decided that a threshold of 50,000 would be the most suitable for the optimisation model.

In Figures 4.15 to 4.17 the approximated fitness and real simulated fitness scores are shown for the evaluations performed by the optimisation model. These are plotted against the best found fitness so far.

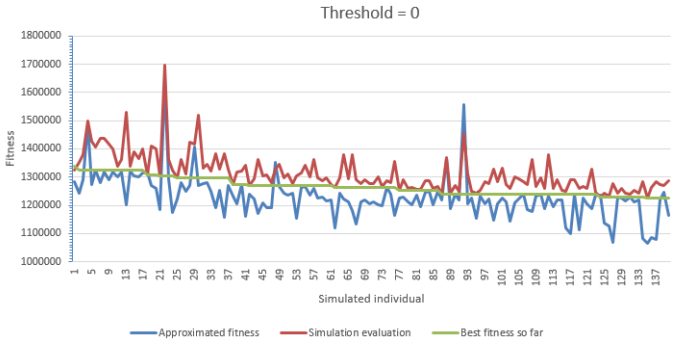

Figure 4.15: Predicted and real simulation values with threshold of 0

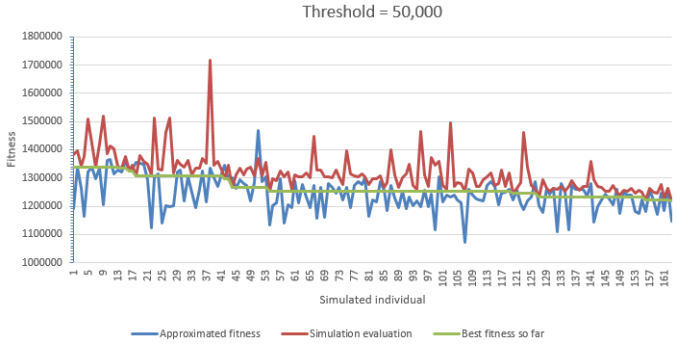

Figure 4.16: Predicted and real simulation values with threshold of 50000

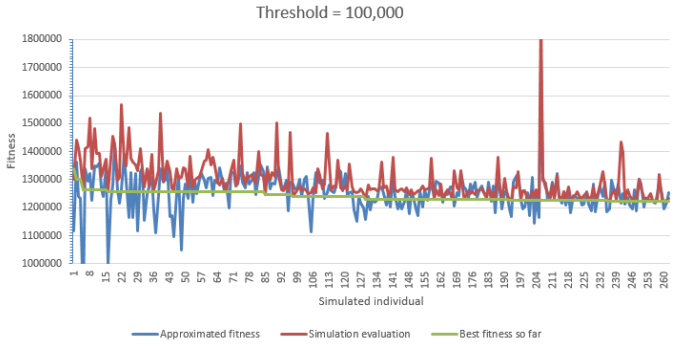

Figure 4.17: Predicted and real simulation values with threshold of 100000

Impact of ensure probability

To measure the impact of the ensure probability an experiment was made where different ensure probabilities were tested against each other. Each of the experiments used the same configuration

defined in Table 4.4 except for the value of the ensure probabilities. The probabilities tested were ensure probability 0% (no ensure probability), 1% (a small probability) and 10% (a high probability). In Figure 4.18 the results are shown for each of the probabilities.

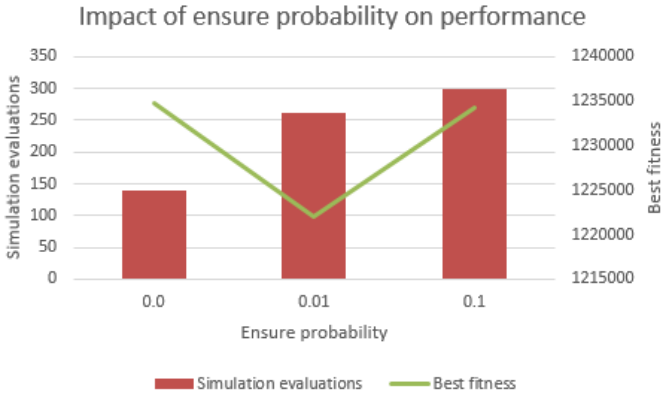

Figure 4.18: Relation between ensure probability, number of evaluations and best fitness score

It can be seen that a small ensure probability will in fact improve performance over only using a threshold based approach. It can also be seen that when the ensure probability is higher, the actual performance decreases. This is due to too many unnecessary simulation evaluations which heavily impact the limited simulation budget provided by the stopping criteria (300 evaluations).

In Figures 4.19 to 4.21 the individual experiments are shown for ensure probabilities 0%, 1% and 10%.

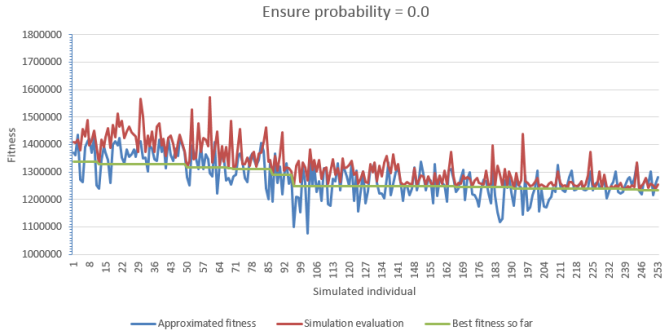

Figure 4.19: Predicted and real simulation values with an ensure probability of 0.0

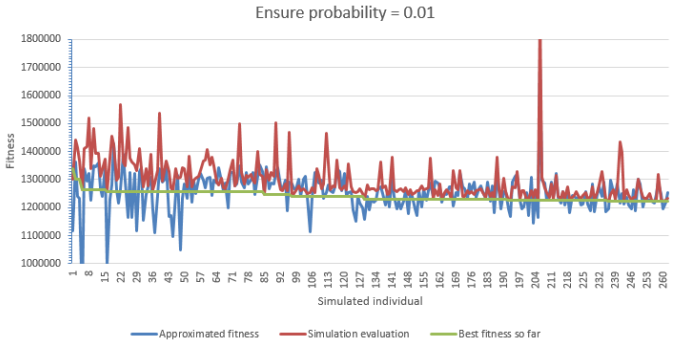

Figure 4.20: Predicted and real simulation values with an ensure probability of 0.01

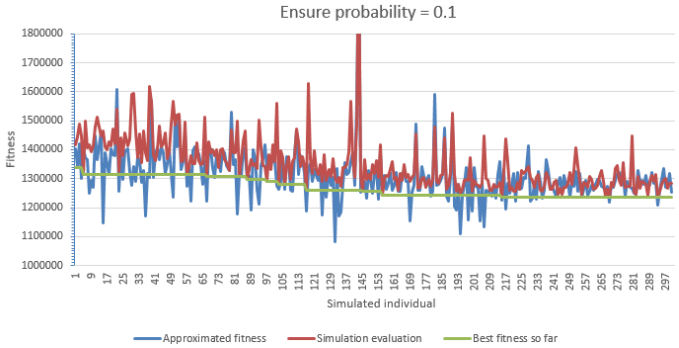

Figure 4.21: Predicted and real simulation values with an ensure probability of 0.1

4.5.5 Impact of mutation probability

Mutation probability impacts the optimisation methodology by allowing additional exploration of the solution space, outside of the existing gene pool. An experiment was made to find the isolated impact of the mutation probability on the overall optimisation quality. In order to do so, three different probabilities were tested: 0.1 (low), 0.2 (medium) and 0.3 (high). The summarised results can be found in Figure 4.22. In Figures 4.23 to 4.25 the mutation probabilities of 10%, 20% and 30% respectively are tested.

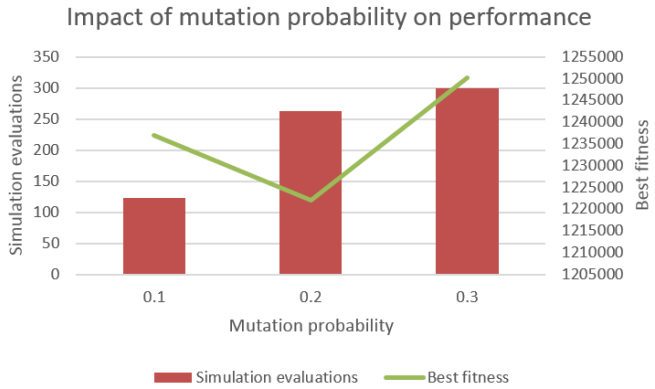

Figure 4.22: Relation between mutation probability, number of evaluations and best fitness score

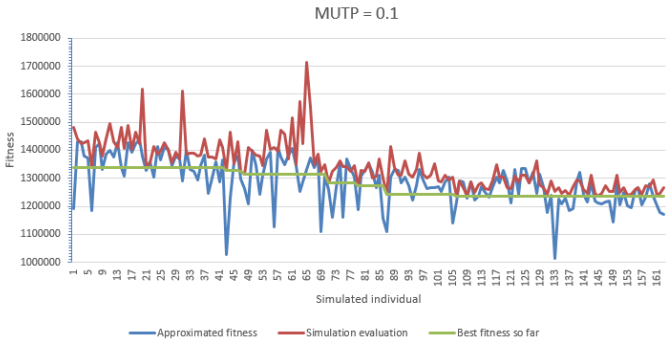

Figure 4.23: Predicted and real simulation values with a mutation probability of 10%

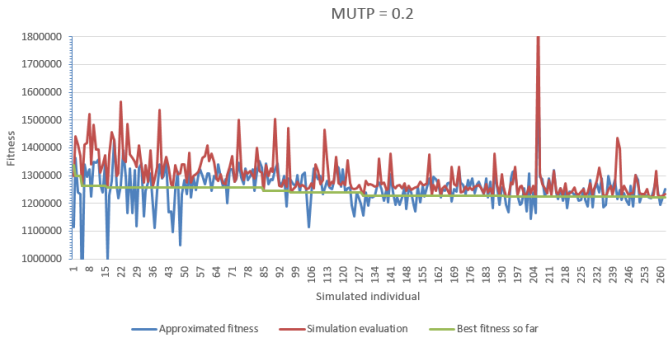

Figure 4.24: Predicted and real simulation values with a mutation probability of 20%

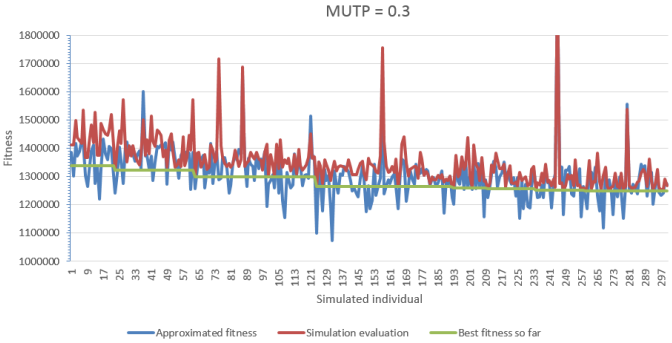

Figure 4.25: Predicted and real simulation values with a mutation probability of 30%

4.5.6 Comparison against single global meta-model approach

In Barton and Meckesheimer (2006), a specific approach is discussed which uses only the meta-model approximations as fitness evaluations in the meta-heuristic optimiser. In this approach, the algorithm starts with running a number of simulations and calculating their simulated fitness. This creates a sample of input-output pairs that can subsequently be used in creating a global meta-model. Unlike the original method used, the meta-model is not used to filter which solutions to reject but instead completely replaces the original fitness evaluation method. The optimisation is subsequently done only by using the approximated fitness.

This effectively eliminates the use of simulation evaluations while running the GA. The biggest problem with this methodology is that, as no evaluations are performed during the optimisation steps, it is unknown whether the population fitness is actually improving or becoming worse. It can only be said that the approximated fitness is improving or becoming worse. The accuracy of the approximation model quality is increasingly more important for this technique than the original methodology. As this methodology requires better accuracy, it is often better to use more training data than the methodology that has been used so far. In order to prevent intermediate simulation evaluations, a negative threshold was used of $-999,999,999$ (a very large negative number) to prevent any simulation evaluations being performed while running the genetic algorithm. As no simulation evaluations are performed, no-retraining of the meta-model was done either. This saves significant time compared to the original methodology proposed.

As the general algorithm is very similar to the original used algorithm, only a little extra code was added to the original algorithm to perform simulation evaluations of the final population generated. This allows to get the true (simulated) fitness of the individuals.

It was decided to test this methodology using 1000 and 2000 initial training cases, using the same set-up as has been used before. The best results of the methodologies were respectively $1,253,588$ for 1000 training cases and $1,224,344$ for 2000 training cases.

In the end only a limited amount of simulation evaluations are performed besides the original training simulation evaluations, respectively 64 simulation evaluations in the scenario with 1000 initial training cases, and 24 simulation evaluations in the scenario with 2000 initial cases. These simulations are performed to find the simulated fitness of the individuals. As the entire final population is evaluated, the difference in number of simulation evaluations can be described by the degree of convergence of the final population. Fewer simulation evaluations show that more of the final population consists of the same individuals (as every possible solution is only simulated once at most), showing a higher degree of convergence. The results of the single global meta-model methodology are relatively good, even though limited training data was used and the performance of the meta-model is not expected to be very good (as explained in Section 4.4.1).

The downside of this methodology is the effective time it takes to run simulations in order to train the global meta-model. For instance the scenario with 2000 cases would take roughly 11 hours to generate the training set. However, it subsequently only took a couple of minutes for optimisation.

4.5.7 Comparison of optimisation model against random simulation evaluations

In order to validate the effectiveness of the optimisation model a comparative experiment was done to test whether the optimisation model showed any improvement compared to random simulations were performed. To compare each of the experiments against random simulations, the best found fitness score after 1000, 1250, 1500, 1750 and 2000 simulation evaluations is shown in Table 4.5.

Table 4.5: Best fitness score after random simulation evaluations

Number of simulation evaluations	Best fitness score random simulations
1000	1,337,692
1250	1,337,692
1500	1,337,692
1750	1,337,692
2000	1,337,692

In the performed experiment the best fitness score did not improve after 800 evaluations and looking more closely to the simulation evaluations it was found that the best solution found by random simulation evaluations was found after 184 evaluations and it did not improve afterwards. On the contrary, the used optimisation model, regardless of its configuration, found multiple better performing individuals using the same amount or less evaluations than used in this experiment.

It can therefore be concluded that the optimisation model, does indeed help in optimising the configurations of the simulation model.

4.6 Selection and application of best optimisation model configuration

Based on the experiments performed in Section 4.5 a best found configuration was determined. This configuration can be seen in Table 4.6. Please note that the number of elites was not explicitly tested. All experiments run used a value of 2 for the number of elites. In addition the training set size for the initial meta-model was 1,000 cases.

In Figure 4.26 the predicted and real simulation values are shown for this set-up. The settings of the best found solution are 110 trailers and 28 tankers in Leuven with multiplication factors 1.061 and 1.108 for external trailers and tankers. 135 trailers and 2 tankers in Jupille, with multiplication factors 1.009 and 1.180 for trailers and tankers respectively. Furthermore, 36 trailers and 9 tankers

Table 4.6: Settings best configuration

Parameter	Population size	Ensure Probability	Threshold	Mutation Probability	Number of elites
Value	100	0.01	50,000	0.2	2*

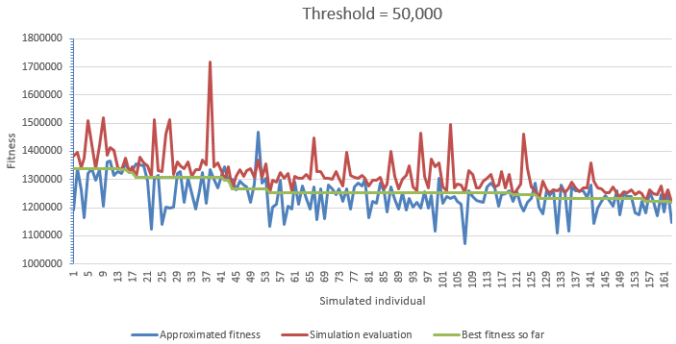

Figure 4.26: Predicted and real simulation values for best configuration

in Hoegaarden with multiplication factor 1.022 for external trailers. The parking in Leuven would be increased to 146, Jupille to 92 and Hoegaarden to 70. It is interesting to note the big increase in parking capacity at the different sites. The reason for this is likely the relatively cheap addition of parking places, compared to the high cost of parking problems. The score of this configuration equals $1,219,788$. The running time of this methodology can be estimated using the number of simulation evaluations done and the amount of retraining. The training set contains 1,000 simulated cases. Subsequently 161 simulation evaluations were performed in the optimisation cycle and 10 times neural network selection and training occurred. As one simulation takes circa 20 seconds and one model selection instance takes circa 2 minutes. This gives a total estimated running time of 407 minutes (or 6.8 hours). It is important to note that over 80% of this time is attributed to the generation of the initial training set. Subsequent simulation evaluations attribute to circa 15% and model training to circa 5% of the total time.

A sensitivity analysis was done on the best solution by running experiments of surrounding cases. This was done by generating new candidate solutions that were the same as the best found solution and adding either one or two in one of the parameters. This gave 55 surrounding cases. Experiments were run for each of these cases and their fitness scores were compared to the best solution. In Table 4.7 the statistics of the sensitivity analysis are shown.

Table 4.7: Summarized statistics of sensitivity analysis for best solution

Minimum fitness	Maximum fitness	Average fitness	Standard Deviation fitness
1,216,892	1,225,190	1,220,241	1,547.257

It can be seen in Table 4.7 that the minimum is lower than the solution found by the optimisation algorithm. This shows that the optimum found by the optimisation methodology was not the true optimum. In the new best solution it was found that adding one more tanker in Hoegaarden could improve performance. When adding another, however overall fitness decreased again. This could suggest a future improvement of the algorithm with an additional exploitation phase that uses local search to explore surrounding solutions. As the minimum and maximum fitness only differs slightly from the original fitness, as well as the low standard deviation in fitness score, it can be said that the solution is not very sensitive for changes.

Chapter 5

Conclusions

In this research, simulation and optimisation techniques have been combined to analyse and optimise the trailer fleet configuration at ABI in Belgium. This research project combined a simulation model, derived from real-world processes and data, with an optimisation algorithm that combined the popular genetic algorithm meta-heuristic with a feed-forward neural network filter that approximates fitness of solutions to determine whether or not to evaluate them using simulation.

By analysing the current situation of the trailer management at ABI it was found that there are many factors that influence trailer availability and trailer related costs. It was found that increasing the trailer fleet size cannot be done without taking into account capacities of parking as parking capacity problems were already being experienced at the three sites. Furthermore, it was found that the non-homogeneous pattern of processing shipments (and with this demand for trailers) was crucial to take into account when trying to improve the trailer management processes. Therefore, the non-homogeneous demands and variability of the different processes were involved in the model. It was found that there was a high degree of variability in the different processes which required many of the different processes to be modelled specifically for different kinds of flows and vehicles. In the end, processes were modelled site-specific, for owned versus external vehicles, for the different kind of vehicles (trailer and tanker) and from where to where they travel to. Table 5.1 shows the results of simulating the current situation with the simulation model.

Table 5.1: Outputs current situation

Statistic	Leuven	Jupille	Hoegaarden	Total
Average delayed orders	1,570.6	790.9	723.3	3084.8
St. dev. Delayed orders	847.1	764.0	487.9	1070.0
Total orders	15,509	12,958	5,257	33,724
Percentage of total (delays)	10.1%	6.1%	13.8%	9.1%
Average times parking full	272.6	412.0	537.1	1,221.7
St. dev. times parking full	126.1	111.1	182.1	186.5

While no specific data was available for these performance measures, it was decided by stakeholders that they seemed accurate providing face validity for the model.
The optimisation of this simulation model aimed to optimise a configuration of input parameters of the simulation model to minimise a cost function that used both the input parameters of the simulation and its outputs for the estimated cost of the configuration in the real world. The selected parameters to optimise were the amount of owned vehicles per type at each of the production locations, the amount of (additional) parking places at each of the sites and scaling parameters for the amount of shipments done on external vehicles for each of the sites at each of the location.

Before applying the optimisation model, the interfaces between the simulation and optimisation model were defined. It was found that the optimisation and simulation model needed a way to share the inputs and outputs of the simulation model. Therefore, interpreters were created. The original optimisation method was extended with a simple mechanic named the ensure probability. It was shown that this mechanic could improve the overall effectiveness of simulation optimisation methods using meta-model filters. Especially when only a small value is used to ensure evaluations. It was shown that the configuration of the algorithm can have a big impact on its effectiveness. The parameters tested (under a constrained simulation budget) were the population size, initial training set size, ensure probability, threshold value and mutation probability.

Applying the optimisation algorithm with its best configuration found a solution with fitness of $1,219,788$ euros compared to the current situation of $1,968,248$ euros which would lead to potential savings of $748,460$ euros. In the optimised configuration nearly no trailer unavailability problems exist anymore (221.4 delayed orders instead of 3084.8) and parking capacity problems have been solved as well (11.9 overloads compared to 1221.7 now). The optimal found solution reduces the amount of owned tankers and increases the amount of owned trailers. Furthermore, it increases the amount of both external trailers and tankers. The parking is increased at each of the locations to facilitate the extra resources deployed. The higher capacity significantly reduces the trailer availability problems and the larger parking nearly completely solves the parking problem.

It was therefore recommended that ABI increased their trailer fleet to better match it with their demand. Likewise, it was recommended to decrease the tanker fleet, as capacity outweighs the actual tanker need. Furthermore, it was recommended to identify flows currently performed on owned trailers that could be replaced with external shipments. These external shipments reduce the need for always having the trailers around and reduce the complexity of managing the trailers between sites. Last but not least, it was recommended to increase the parking capacity to prevent parking overflows from occurring and facilitating storage of the increased trailer fleet size.

To conclude, this research has shown that the implementation of a simulation optimisation algorithm may help in improving trailer management.

Chapter 6

Discussion

The used simulation optimisation methodology effectively combines three techniques: simulation, optimisation and regression. Each of these techniques have been researched to great extent and each of these techniques could likely be further optimised.

The simulation could likely be improved when variance in the simulation model is decreased. Ways to do this would be a lower abstraction level which would reduce the need for certain assumptions. Due to the low data quality and lack of information on specific parts of the process the simulation could also be improved when more and better data would be made available. For example, by combining global positioning system (GPS) data (currently not available) with the current WMS data, the actual position of trailers could be determined over time and more accurate process times could be derived. Furthermore, the simulation model could be improved by identifying specific flows and modelling them accordingly. The current simulation model has a high degree of variability as specific cases are all generalised into a couple of general cases. The reason for this is partly the scope of this project (as every additional flow modelled would require additional time to identify, analyse, design and implement) and the fact that it requires human intervention in the fitting of the distributions. The flows could be partly identified based on origin and next locations but would likely require some kind of clustering (location names contain mistakes and some location have multiple entries in the system) and classification (which shipments belong to which flows) in order to make this possible.

This research could be improved upon by taking a deeper dive on one or several of the used techniques. For the optimisation part, one could use parallel computing techniques to reduce the overall time required for optimisation as multiple simulations could be run alongside each other. The optimisation and regression techniques can also be improved using more elaborate hyper-parameter optimisation techniques.

Another improvement possibility that has not been widely explored in this research is the usage of feature selection to identify the important features for optimisation. Seeing however that each of the individual attributes are directly involved in the cost function it is likely unwise to ignore some of the features. On the contrary, seeing as there is definitely interaction between several of the inputs it could have been interesting to add interaction effects and subsequently use feature selection techniques to see if this would improve performance.

Even though this research focuses on optimisation of simulations it was not possible to say with 100% certainty that the global optimum was found. With over 10^{30} possible configurations it is simply impossible to find the true optimum by exhausting the search space, especially as evaluations take significant time. Furthermore, it was found that the solutions proposed by the optimisation algorithm might not be as easy to implement in the current processes of ABI. The current logistics process is focused on direct loading of trailers which increases the need for trailer availability. The proposed solutions of the optimisation algorithm tend to slightly reduce the size of the trailer fleet and instead increase the amount of shipments fulfilled on external trailers. The reasoning behind this is intuitive, as a fixed cost gets exchanged for a variable cost that will only occur when needed. Furthermore, external trailers do not use parking capacity which prevents

capacity problems. In reality, this would however only magnify the problem for direct loading as there is a higher dependency on the availability of transporters and less flexibility in the loading schedule. Based on these results it could be smart for ABI to re-evaluate their direct loading strategy and see in which cases direct loading is providing value and in which cases it should be avoided.

The simulation optimisation part of this research focused on finding a balance between owned and external trailers as well as the required parking size. However, there are many other parameters of the model that could have been explored as well. In theory any set of simulation parameters could have been explored using the simulation optimisation methodology. However, due to the initially defined focus on these parameters and the limitations of the simulation software (which required highly specific inputs in order to simulate cases) it was not feasible to explore other parameters to the same extend.

Future research could validate the impact of the genetic algorithm configuration on simulation optimisation effectiveness by testing their impact in the simulation optimisation of other simulation models. Furthermore, the used methodology is an online simulation optimisation method using the meta-model as a filter for potential bad solutions. The theoretical background however showed that there are many cases where the evaluation function is completely replaced by the meta-model. It would be interesting to analyse the cut-off point for when one of the methods would outperform the other. Related to this, further research could analyse the frequency of re-training the meta-model. In the current setup this was done every generation but the results indicate that this may not be necessary. It could improve the speed of the methodology significantly when less retraining is performed. Furthermore, this research used a two rule stopping criteria for all experiments. Further research could explore the optimisation effectiveness under stricter and less stricter stopping criteria to see the impact of the stopping criteria on this specific methodology.

Bibliography

AB InBev (2017). AB InBev - Bedrijf. http://www.ab-inbev.be/nl_BE/bedrijf.html. Accessed: 2017-03-01. 1

Abo-Hamad, W. and Arisha, A. (2011). Simulation-optimisation methods in supply chain applications: a review. *Irish Journal of Management*, 30(2):95. 12

April, J., Glover, F., Kelly, J. P., and Laguna, M. (2003). Simulation-based optimization: practical introduction to simulation optimization. In *Proceedings of the 35th conference on Winter simulation: driving innovation*, pages 71–78. Winter Simulation Conference. vii, 3, 7, 8, 11, 12, 37

Azadivar, F. (1999). Simulation optimization methodologies. In *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*, pages 93–100. ACM. 7

Barton, R. R. and Meckesheimer, M. (2006). Metamodel-based simulation optimization. *Handbooks in operations research and management science*, 13:535–574. 7, 8, 11, 52

Bastiaens, K. (2017). Personal Communication. First Line Manager Warehouse, Stella Artois Brewery, Leuven. 13, 14, 25

Bierlaire, M. (2015). Simulation and optimization: A short review. *Transportation Research Part C: Emerging Technologies*, 55:4 – 13. 9

Bowden, R. O. and Hall, J. D. (1998). Simulation optimization research and development. In *1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274)*, volume 2, pages 1693–1698 vol.2. vii, 6, 40

Caignau, K. (2017). Personal Communication. Transport & Asset Manager, Stella Artois Brewery, Leuven. 12, 13, 16, 22, 24, 25, 33

Carson, Y. and Maria, A. (1997). Simulation optimization: methods and applications. In *Proceedings of the 29th conference on Winter simulation*, pages 118–126. IEEE Computer Society. vii, 6, 7

Cleton, M. (2017). Personal Communication. BRM Supply Europe, AB InBev Global Headquarters, Leuven. 14

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11):1245–1287. 36

Cosemans, Y. (2017). Personal Communication. Plant Manager, Hoegaarden Brewery, Hoegaarden. 13

Crainic, T. G. and Laporte, G. (1997). Planning models for freight transportation. *European journal of operational research*, 97(3):409–438. 10

Daughety, A. F. and Turnquist, M. A. (1981). Budget constrained optimization of simulation models via estimation of their response surfaces. *Operations Research*, 29(3):485–500. 8

Davis, L. (1987). *Genetic algorithms and simulated annealing*. Morgan Kaufman Publishers, Inc.,Los Altos, CA. 8

de Ridder, C. (2017). Personal Communication. Project Manager Logistics, Global Headquarters, Leuven. 2, 25, 26

Dejax, P. J. and Crainic, T. G. (1987). Survey paper—a review of empty flows and fleet management models in freight transportation. *Transportation Science*, 21(4):227–248. 11

Delignette-Muller, M. L., Dutang, C., et al. (2015). fitdistrplus: An r package for fitting distributions. *Journal of Statistical Software*, 64(4):1–34. 26

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175. 40

Fu, M. C., Andradóttir, S., Carson, J. S., Glover, F., Harrell, C. R., Ho, Y.-C., Kelly, J. P., and Robinson, S. M. (2000). Integrating optimization and simulation: research and practice. In *Proceedings of the 32nd conference on Winter simulation*, pages 610–616. Society for Computer Simulation International. 9

Fu, M. C., Glover, F. W., and April, J. (2005). Simulation optimization: a review, new developments, and applications. In *Simulation Conference, 2005 Proceedings of the Winter*, pages 13–pp. IEEE. 1

Hong, L. J. and Nelson, B. L. (2009). A brief introduction to optimization via simulation. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 75–85. IEEE. 6

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366. 38

Hurrion, R. D. (2000). A sequential method for the development of visual interactive meta-simulation models using neural networks. *Journal of the Operational Research Society*, 51(6):712–719. 7, 8, 11

Ingalls, R. G. (1998). The value of simulation in modeling supply chains. In *Proceedings of the 30th conference on Winter simulation*, pages 1371–1376. IEEE Computer Society Press. 1, 5

JaamSim Development Team (2017). *JaamSim: Discrete-Event Simulation Software*. Version 2016-04. 25, 32

Jalali, H. and Nieuwenhuyse, I. V. (2015). Simulation optimization in inventory replenishment: a classification. *IIE Transactions*, 47(11):1217–1235. 7

Köchel, P., Kunze, S., and Nieländer, U. (2003). Optimal control of a distributed service system with moving resources: Application to the fleet sizing and allocation problem. *International Journal of Production Economics*, 81:443–459. 11

Laguna, M. and Marti, R. (2002). Neural network prediction in a system for optimizing simulations. *Iie Transactions*, 34(3):273–282. 7, 11

Langouche, L. (2017). Personal Communication. Consultant, Adneom, Jupille. 13

Law, A. M. (2005). How to build valid and credible simulation models. In *Proceedings of the 40th Conference on Winter Simulation*, pages 39–47. Winter Simulation Conference. vii, 5, 15, 16

Law, A. M., Kelton, W. D., and Kelton, W. D. (1991). *Simulation modeling and analysis*, volume 2. McGraw-Hill New York. 1, 5

Le Riche, R., Gaudin, J., and Besson, J. (2003). An object-oriented simulation–optimization interface. *Computers & structures*, 81(17):1689–1701. 10

Legato, P. and Mazza, R. M. (2001). Berth planning and resources optimisation at a container terminal via discrete event simulation. *European Journal of Operational Research*, 133(3):537 – 547. 11

Lonjaret, A. (2017). Personal Communication. Logistics Implementation Expert, AB InBev Business Service Centre, Prague. 13, 25

Magnier, L. and Haghighat, F. (2010). Multiobjective optimization of building design using trnsys simulations, genetic algorithm, and artificial neural network. *Building and Environment*, 45(3):739–746. 9

Parijs, C. (2017). Personal Communication. Logistics Reception, Stella Artois Brewery, Leuven. 14

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. 40

Pham, D. and Karaboga, D. (2012). *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*. Springer Science & Business Media. 8

Ratzer, A. V., Wells, L., Lassen, H. M., Laursen, M., Qvortrup, J. F., Stissing, M. S., Westergaard, M., Christensen, S., and Jensen, K. (2003). Cpn tools for editing, simulating, and analysing coloured petri nets. In *Proceedings of the 24th International Conference on Applications and Theory of Petri Nets*, ICATPN'03, pages 450–462, Berlin, Heidelberg. Springer-Verlag. 19, 21

Reynaert, D. (2017). Personal Communication. Regional Scheduling Manager, Stella Artois Brewery, Leuven. 12

Richelle, R. (2017). Personal Communication. BRM Supply Europe, AB InBev Global Headquarters, Leuven. 13, 14, 31

Robinson, S. (2005). Discrete-event simulation: From the pioneers to the present, what next? *The Journal of the Operational Research Society*, 56(6):619–629. 7, 8, 9

Rohrer, M. W. (2000). Seeing is believing: the importance of visualization in manufacturing simulation. In *Simulation Conference, 2000. Proceedings. Winter*, volume 2, pages 1211–1216. IEEE. 32

RStudio Team (2015). *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA. 26

Rudolph, G. (1994). Convergence analysis of canonical genetic algorithms. *IEEE transactions on neural networks*, 5(1):96–101. 10

Sargent, R. G. (2005). Verification and validation of simulation models. In *Proceedings of the 37th conference on Winter simulation*, pages 130–143. winter simulation conference. 5, 32

Schrover, E. (2017). Personal Communication. Logistics Manager, Stella Artois Brewery, Leuven. 12, 13, 18, 25, 31, 36, 37

Song, D.-P. (2005). Optimal threshold control of empty vehicle redistribution in two depot service systems. *IEEE Transactions on Automatic Control*, 50(1):87–90. 11

Spears, W. M. et al. (1992). Crossover or mutation. *Foundations of genetic algorithms*, 2:221–237. 9

Valckenborg, J. (2017). Personal Communication. Logistics Reception, Stella Artois Brewery, Leuven. 14

van den Hoogenhoff, R. (2017). Personal Communication. Transport Planning Expert, AB InBev Business Service Centre, Prague. 37

van Hulst, B. (2017). Personal Communication. Logistics Manager, Stella Artois Brewery, Leuven. 22, 37

Verhelst, F. (2017). Personal Communication. Supply Network Planner, AB InBev Global Headquarters, Leuven. 13

Walrand, J. (1988). *An introduction to queueing networks*. Prentice Hall. 23

Wang, L. (2005). A hybrid genetic algorithm–neural network strategy for simulation optimization. *Applied Mathematics and Computation*, 170(2):1329–1343. vii, 9

Xu, J., Huang, E., Chen, C.-H., and Lee, L. H. (2015). Simulation optimization: a review and exploration in the new era of cloud computing and big data. *Asia-Pacific Journal of Operational Research*, 32(03):1550019. 9

Xu, J., Zhang, S., Huang, E., Chen, C.-H., Lee, L. H., and Celik, N. (2014). Efficient multi-fidelity simulation optimization. In *Proceedings of the 2014 winter simulation conference*, pages 3940–3951. IEEE Press. 9

Appendix A

Distribution fitting

In this section, a step by step example is given of the used distribution fitting methodology as described in Section 3.4.4. Standardized outlier bounds have been defined as shown in Section 3.4.2. First, each dataset is split up into separate datasets based on the parameter being modelled. An example for the script splitting the datasets in combinations of plant, vehicle type and vehicle owned/not owned can be seen in Listing 1.

```
f <- file.choose()
vehdata <- read.csv2(f)
filesplit <- split(vehdata, list(vehdata$PLANT, vehdata$VEHICLE_TYPE,
vehdata$INBEV_TRAILER))

for(n in names(filesplit))
        write.table(filesplit[[n]],
                file = paste0(gsub("\\.", "_", n),".csv"),
                                row.names = FALSE, sep = ";", dec = ",")
```

Listing 1: Data splitting in R

After splitting the data each of the distributions is fitted using the `fitdistrplus` package in R. The code for the derivation of process times is shown in Listing 2. For the out times and wait before departure times the same setup is used but the variable name `"PROCESSING_TIME"` gets changed to `"OUT_TIME"` and `"WAIT_TIME_BEFORE_DEPARTURE"`. The process for the out times is slightly different in that the min and the max are changed to represent the different value ranges.

```
# import libraries
library("fitdistrplus")
library("gplots")

# Set working directory
setwd('working directory with files') # specify your own working directory

# Select the file to analyse
f <- file.choose()
data <- read.table(f, header=T, sep=";", dec=",", na.string=c("DOUBLE_SHIPMENT",
"NULL", "NO ABI TRAILER", "LAST_RECORD_OF_TRAILER"))

# Specify name of the output file, minimum value, maximum value and the interval
# of the histogram
name="JUPTankYes"
```

```
min = 0.01
max = 12
histinterval = 0.1

# Determine the number of breaks for the histogram
nrbreaks <- seq(min, max, histinterval)

# Filter data based on minimum and maximum
filtered <- data[!(data["PROCESSING_TIME"] <= min | data["PROCESSING_TIME"] >
max ), ]
varname <- filtered\$PROCESSING_TIME

# Calculate x bound of plot as the mean plus 3 times the standard deviation
xsd <- sd(varname, na.rm = TRUE)
xmean <- mean(varname, na.rm = TRUE)
xstdev <- xmean+3*xsd

# Calculate skewness and kurtosis and plot Cullen & Frey graph
pdf(paste0("CF_",name,".pdf"), width = 10, height = 7.5)
descdist(varname, boot = 100)
dev.off()

# Fit the lognormal, gamma and normal distribution using MLE
fln <- fitdist(varname, distr="lnorm")
fg <- fitdist(varname, distr="gamma")
fn <- fitdist(varname, "norm")

# Plot comparative diagrams for the fitted distributions
pdf(paste0("Fit_",name,".pdf"), width = 10, height = 7.5)
dhist <- hist(varname, breaks=nrbreaks, plot=FALSE)
highestDensity <- max(dhist\$density)
par(mfrow = c(2,2))
plot.legend <- c("Lognormal", "Gamma", "Normal")
denscomp(list(fln, fg, fn), xlim=c(min, max), ylim=c(0, highestDensity),
breaks=nrbreaks)
qqcomp(list(fln, fg, fn), addLegend = TRUE)
cdfcomp(list(fln, fg, fn), c(0, xstdev), c(0, 1.2),
legendtext = plot.legend)
ppcomp(list(fln, fg, fn),addLegend = TRUE)
dev.off()

# Write goodness of fit stats to text file
sink(paste0("Gof_",name,".txt"))
gofstat(list(fln, fg, fn), fitnames = c("lnorm", "gamma", "normal"))
cat("-------------------------------------------------\n")
summary(fln)
cat("-------------------------------------------------\n")
summary(fg)
cat("-------------------------------------------------\n")
summary(fn)
sink()
```

Listing 2: Distribution fitting code in R

To illustrate the outcome of the different scripts and the methodology, the outputs have been displayed and explained in the subsequent figures. For each figure it is explained what is seen and what the conclusions are. The chosen example is the distribution of wait before departure times for owned trailers in Leuven.

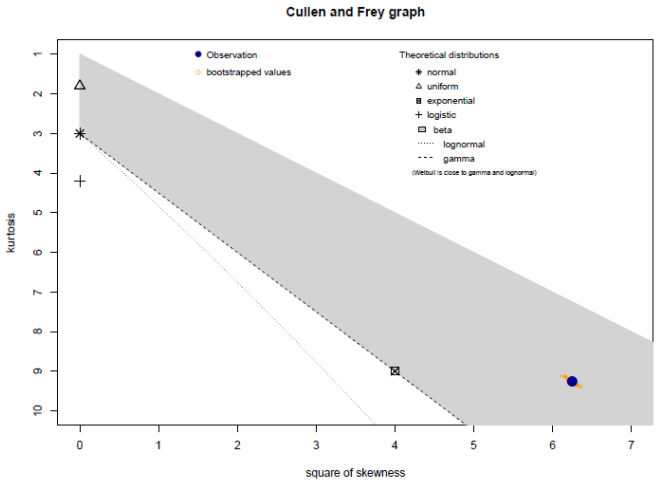

Figure A.1: Cullen & Frey graph example

In Figure A.1, the Cullen & Frey graph is shown. On the y-axis the kurtosis is shown and on the x-axis the square of skewness. Skewness is a measurement of asymmetry, where a high skewness means that the distribution is very asymmetric. Kurtosis is based on whether there are many outliers and heavy tails of the distribution compared to the normal distribution. It can be found that both values are very high which suggests that the empirical distribution is likely best fitted by an asymmetric, heavily tailed distribution such as a gamma or lognormal distribution.

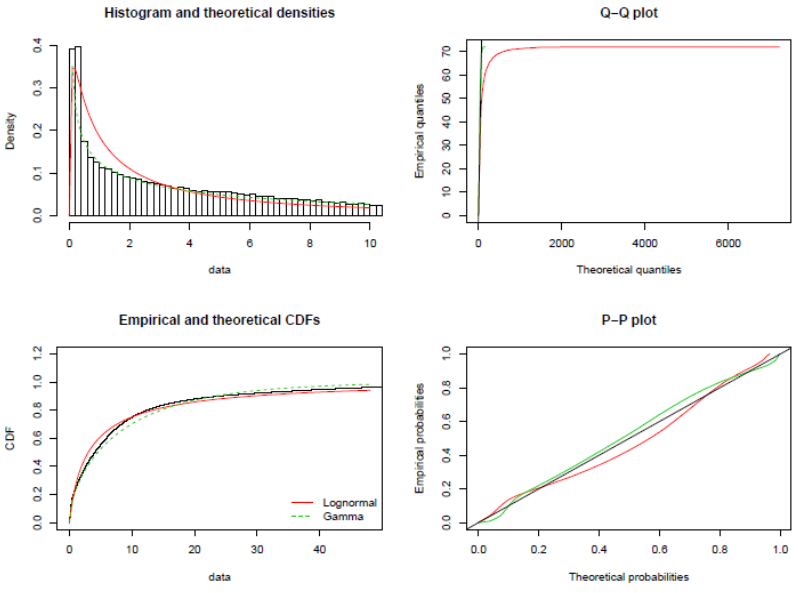

Figure A.2: Comparative distribution plots example

Figure A.2 shows the distributions fitted using MLE compared to the empirical distribution of values. The empirical distribution is shown as bars in the form of an histogram. Neither of

the theoretical densities of the fitted distributions fully captures the empirical distribution but it could be said that the Gamma distribution looks mostly similar to the empirical distribution. The Q-Q plot plots the quantiles of the candidate distributions against the quantiles of the empirical distribution. It can be seen that the candidate Gamma distribution's quantiles show a lot of similarity to the quantiles of the empirical distribution. Regarding the empirical and theoritical cumulative distribution functions it can be seen that both candidate distributions show a good degree of similarity to the empirical CDF. Finally the P-P plot is another way to plot the empirical CDF's against those of the candidate distributions. Eyeballing the results, it shows that the Gamma distribution is the most likely fit based on CDF. This concludes the visual analysis which would suggest the fitted Gamma distribution to be the best fit.

```
Goodness-of-fit statistics
                                      lnorm        gamma
Kolmogorov-Smirnov statistic     0.07036287    0.04714821
Cramer-von Mises statistic     160.64452235   79.89767094
Anderson-Darling statistic     873.99806075 582.61867229

Goodness-of-fit criteria
                                    lnorm    gamma
Aikake's Information Criterion 632781.8 625945.7
Bayesian Information Criterion 632800.8 625964.7
-------------------------------------------------
Fitting of the distribution ' lnorm ' by maximum likelihood
Parameters :
        estimate  Std. Error
meanlog 1.105665 0.005496685
sdlog   1.759042 0.003886738
Loglikelihood:  -316388.9   AIC:  632781.8   BIC:  632800.8
Correlation matrix:
              meanlog         sdlog
meanlog  1.000000e+00 -1.554448e-10
sdlog   -1.554448e-10  1.000000e+00

-------------------------------------------------
Fitting of the distribution ' gamma ' by maximum likelihood
Parameters :
        estimate  Std. Error
shape 0.57714416 0.002133667
rate  0.06507532 0.000359786
Loglikelihood:  -312970.8   AIC:  625945.7   BIC:  625964.7
Correlation matrix:
          shape      rate
shape 1.0000000 0.6684105
rate  0.6684105 1.0000000
```

Figure A.3: Goodness-of-Fit statistics example

Finally, the goodness-of-fit stats are shown in Figure A.3. The Kolmogorov-Smirnov test is one of the most commonly used non-parametric test to test whether two cumulative distributions belong to the same distribution. It can be seen that for both distributions the KS statistic is not particularly high. Especially the gamma distribution does not have a very high P-value which would suggest that the empirical distribution does not likely follow from either theoretical distribution. However, as the visual analyses showed the most likelihood with the Gamma distribution it is decided to model the distribution of the wait before departure times for owned trailers in Leuven using a Gamma distribution. The parameters of this distribution are also found in the Goodness-of-Fit statistics figure and are respectively 0.5771 for the shape parameter and 0.065 for the rate parameter, as was displayed in Table 3.9 in Section 3.4.7.

The out times were derived using a similar approach however one extra step was added in the beginning, namely the definition of the value ranges. This was done based on the original histogram of the data. The histogram of the out times for trailers from Leuven to Leuven is shown in Figure A.4.

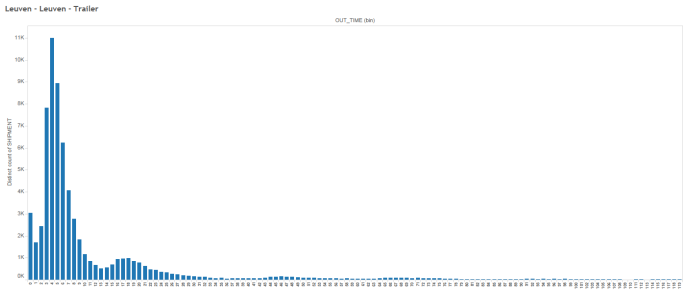

Figure A.4: Histogram of out times Leuven to Leuven - Trailers

It can be seen that the distribution is not likely caught in a theoretical distribution such as a gamma, normal or lognormal distribution. Instead the dataset was split in four parts, where each part contains the data of all out time values in a specific range. For the given example the splits where made from $[0, 2)$, $[2, 12)$, $[12, 72)$ and $[72, 120]$. The distribution fitting techniques were subsequently applied to these value ranges to determine the model distributions for the value ranges. Subsequently a discrete probability was used to determine the likelihood that a value was sampled from one of the value ranges. The theoretical distribution per value range, together with their probabilities are shown in Section 3.4.6.

Appendix B

Feasible decision variable ranges

In Tables B.1 to B.3, the upper and lower bounds for the decision variables are defined for the optimisation algorithm. These bounds are used to limit the potential solution space. The reasoning behind them is that

Table B.1: Decision variable ranges: Leuven

Decision Variable	Trailers LV	Tankers LV	Multiplier External Trailers LV	Multiplier External Tankers LV	Parking LV
Lower Bound	80	20	800	800	108
Upper Bound	150	60	1200	1200	148

Table B.2: Decision variable ranges: Jupille

Decision Variable	Trailers JUP	Tankers JUP	Multiplier External Trailers JUP	Multiplier External Tankers JUP	Parking JUP
Lower Bound	80	1	800	800	50
Upper Bound	150	20	1200	1200	94

Table B.3: Decision variable ranges: Hoegaarden

Decision Variable	Trailers HOE	Tankers HOE	Multiplier External Trailers HOE	Multiplier External Tankers HOE	Parking HOE
Lower Bound	20	1	800	800	30
Upper Bound	70	20	1200	1200	70

Appendix C

JaamSim model

In this section, the different parts of the JaamSim implementation are discussed. The implementation follows the process models as described in 3.3.1. The main difference between the process models and the JaamSim model is that specific process steps were added to the JaamSim model to ensure the logic required.

Trailers are created, their types assigned and then send to an arrival process. In this process, shown in Figure C.1, it is checked whether there is a surplus of trailers at the destination of the trailer.

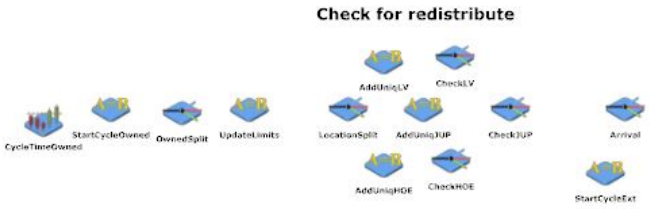

Figure C.1: Arrival process in JaamSim

If there is a surplus, the trailer is send to another site with the highest need for trailers. This need is defined as the number of open calls minus the current number of trailers at the site. Transferring is done in the redistribute process (Figure C.2) and involves a fixed delay of 1 hour. If there is no surplus the trailer arrives at its intended destination and claims a parking spot.

Figure C.3 shows one of the sites in JaamSim. These upper two queues are the queues with open orders and with available trailers.

When there is at least one trailer and one order of the same type available, they are coupled in the process shown in Figure C.4. Both orders get combined, the data of the order is transferred to the trailer and subsequently the order is destroyed. Based on the order data, the location and the vehicle type (owned or not), the processing time, wait time before departure and out time are derived.

The trailer is then moved to the dock seen in Figure C.3 of its type. Upon arrival at the dock, the parking spot is released and the dock is seized. If no dock is available the trailer instead is placed in the queue next to the dock until a dock is available. This process can be seen in Figure C.5.

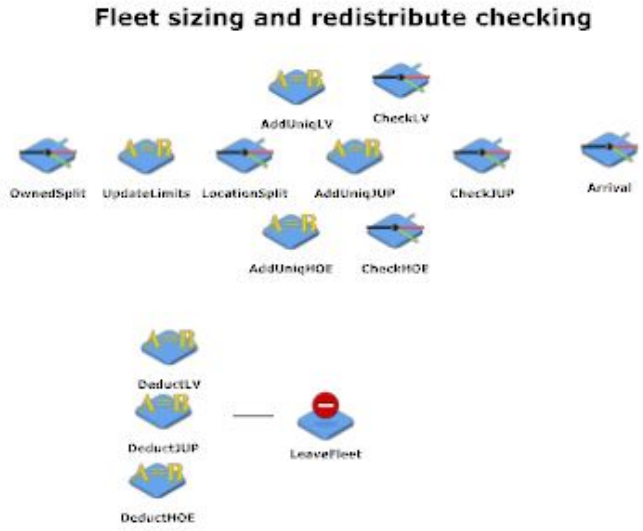

Figure C.2: Redistribution process in JaamSim

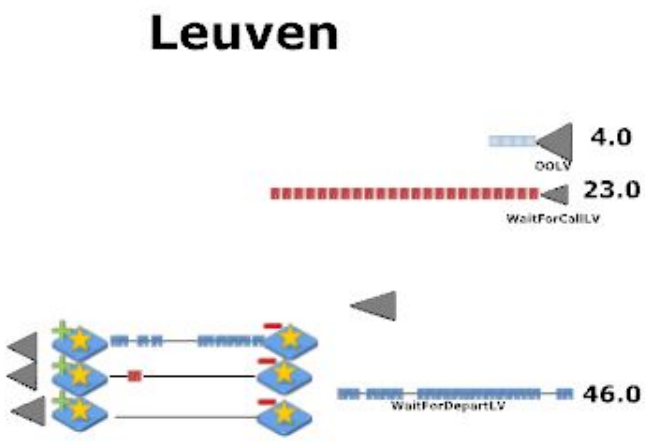

Figure C.3: Site view in JaamSim

Figure C.4: Order coupling in JaamSim

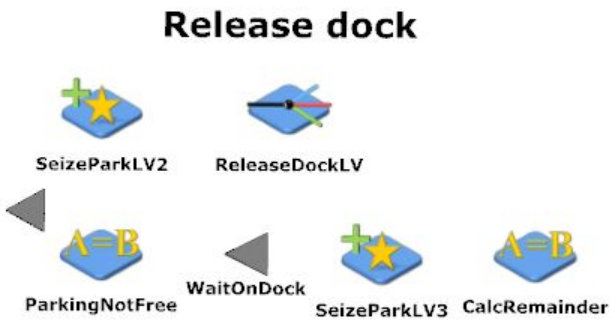

Figure C.5: Release of dock and claiming of parking spots after processing

Appendix D

Data

In Figure D.1 example WMS data is shown for three shipments. The columns in grey show the data that has been used, either directly or in one of the calculations. Data was available for 2014 until 2017 for each of the plants.

ARRIVAL	TRACTOR_IN	TRACTOR_LP_IN	TRACTOR_OWNER_IN	VEHICLE	LICENCE_PLATE	VEHICLE_TYPE	INBEV_TRAILER
15/12/2016 00:02:44	T484	CZP577	USE: 369645) BALL PACKAGIN	T9952	UDR445	Trailer	No
15/12/2016 00:14:32	T24031	1ADE568	DISTRI LOG	ABT2473	OP-30-KG	ATLS	Yes
15/12/2016 00:26:03	T74733	1LRB171	Dedicated Fleet ¿ Dummy	ABT2477	OP-34-KG	ATLS	Yes

SPLIT_LOAD_UNLOAD	UNLOAD_SHIPMENT	UNLOAD_DELIVERY	UNLOAD_SHUTTLE	CALL_DOCK_TO_UNLOAD	START_UNLOAD_CONTROL	END_UNLOAD_CONTROL	UNLOAD_CTRL_DURATION
N	14417145	1601939184	No	15/12/2016 00:44:32	15/12/2016 01:00:51	15/12/2016 01:05:10	259
N	30806285	1301672210	No	15/12/2016 01:06:26	15/12/2016 01:06:40	15/12/2016 01:08:01	81
N	30804387	1301669720	No	15/12/2016 02:43:11	15/12/2016 02:45:47	15/12/2016 02:45:51	4

START_UNLOAD	END_UNLOAD	FREE_DOCK_UNLOAD	UNLOAD_WH	UNLOAD_DOCK	UNLOAD_CUSTOMER_SUPPLIER_NUM	UNLOAD_CUSTOMER_SUPPLIER_NAME	THEO_REAL
15/12/2016 01:03:17	15/12/2016 01:20:18	15/12/2016 01:23:28	CD1	01	369645	Ardagh Metal Beverage Trading	Y
15/12/2016 01:08:01	15/12/2016 01:08:01	15/12/2016 02:16:11	CD1	71	PBE27	ANTWERPEN EXTERNAL WH	Y
15/12/2016 02:45:51	15/12/2016 02:45:51	15/12/2016 05:17:50	CD1	71	PBE03	INBEV JUPILLE S.A.	Y

UNLOAD_OCCUPATION	TRACTOR_OUT	TRACTOR_LP_OUT	TRACTOR_OWNER_OUT	CARRIER_NUM	CARRIER_NAME	SHIPMENT	PRIORITY
39	T484	CZP577	USE: 369645) BALL PACKAGING			14417145	0
70	T37404	1BPR788	DISTRI LOG	327259	DISTRI LOG	30806257	0
155	T61362	1JHK300	DISTRI LOG	327259	DISTRI LOG	30806256	0

LOAD_DELIVERY	CREATION_DATE	FORSEEN_DATE	NB_DELIVERY	NB_MIX_PALLET	START_MIX_PALLET	END_MIX_PALLET	ARR_WH_LOAD
1601939184	13/12/2016 12:02:21	15/12/2016 02:00:00	1	0			15/12/2016 00:43:03
1301672178	14/12/2016 12:46:44	15/12/2016 15:00:00	1	0			15/12/2016 00:17:49
1301672177	14/12/2016 12:47:20	15/12/2016 16:00:00	1	0			15/12/2016 00:27:48

START_LOAD	END_LOAD	LOAD_WH	LOAD_DOCK	CALL_DOCK_TO_LOAD	START_LOAD_CONTROL	END_LOAD_CONTROL	LOAD_CTRL_DURATION
15/12/2016 01:00:51	15/12/2016 01:23:13	CD1	01	15/12/2016 00:44:32	15/12/2016 01:23:23	15/12/2016 01:23:28	5
15/12/2016 01:08:01	15/12/2016 01:59:18	CD1	71	15/12/2016 01:06:26	15/12/2016 02:14:26	15/12/2016 02:16:11	105
15/12/2016 02:45:51	15/12/2016 05:12:46	CD1	71	15/12/2016 02:43:11	15/12/2016 05:17:08	15/12/2016 05:17:50	42

NB_LOAD_CONTROL	LOAD_CONTROL_OK	FREE_DOCK_LOAD	LOAD_OCCUPATION	SHIPMENT_TYPE	CONTAINER	CUSTOMER_OR_SUPPLIER	LOAD_CUSTOMER_SUPPLIER_NUM
1	Y	15/12/2016 01:23:28	39	Pickup		Supplier	369645
1	Y	15/12/2016 02:16:11	70	Delivery		Customer	PBE27
1	Y	15/12/2016 05:17:50	155	Delivery		Customer	PBE27

LOAD_CUSTOMER_SUPPLIER_NAME	DEPARTURE	CHECK_IN	FIRST_PLANING	NB_REPLENISHMENT
Ardagh Metal Beverage Trading	15/12/2016 01:40:26	15/12/2016 00:02:44	15/12/2016 02:00:00	0
ANTWERPEN EXTERNAL WH	15/12/2016 12:54:48	15/12/2016 12:18:47	15/12/2016 15:00:00	0
ANTWERPEN EXTERNAL WH	15/12/2016 13:45:51	15/12/2016 13:00:35	15/12/2016 16:00:00	0

Figure D.1: Example WMS data

The original data table was subsequently enriched by adding the `PLANT` field and the calculations as discussed in Section 3.4.1. The added data fields are shown in Figure D.2

PLANT	TYPE_TRANSPORT	TIME_ON_SITE	PROCESSING_TIME	AIT_TIME_BEFORE_DEPARTUR	DELTA_PLAN_DEPT	NEXT_PLANT	NEXT_ARRIVAL_TIME
LEUVEN	LoadUnload	8,697	1,8325	4,45	1,133333333	JUPILLE	7/01/2016 23:12
LEUVEN	LoadUnload	50,894	2,3674	32,36666667	0,533333333	JUPILLE	20/01/2016 11:42
HOEGAARDEN	LoadNoUnload	6,781	3,024	0,016666667	-0,783333333	LEUVEN	4/11/2015 13:49

OUT_TIME	NEXT_DESTINATION
10,7	INBEV DEPOT ZUUN (BELLE-VUE)
12,564	INBEV DEPOT ZUUN (BELLE-VUE)
2,457	INBEV LEUVEN CD1 - EF

Figure D.2: Added fields WMS Data

To determine these values two `VBA`-scripts were written. The first VBA script calculates processing times and waiting times, this script can be found in Listing 3. The second script calculates the out time and determines the next arrival time, next plant, the out time and the next destination. This second script can be found in Listing 4.

```
Sub CalculateFields()
    'Prevents screenupdates before calculating all fields
```

```
Application.EnableEvents = False
Application.ScreenUpdating = False

'This VBA file assumes correct data and missing fields and datevalues of 2921
'being replaced with "NULL"
Dim typeOfTransport As String 'Variable to store type of transport
Dim calcArray(1, 5) As Double
Dim i As Long
Dim output As Integer 'First field to use as output
Dim valuetaken As String
output = 63

Cells(1, output).Value = "TYPE_TRANSPORT"
Cells(1, output + 1).Value = "TIME_ON_SITE"
Cells(1, output + 2).Value = "PROCESSING_TIME"
Cells(1, output + 3).Value = "WAIT_TIME_BEFORE_PROCESS"
Cells(1, output + 4).Value = "WAIT_TIME_BEFORE_DEPARTURE"
Cells(1, output + 5).Value = "DELTA_PLAN_DEPT"
Cells(1, output + 6).Value = "DEPARTURE_DATE"

'For each of the shipment entries
For i = 2 To Rows.Count

    'Check if the row is not empty, if so then exit the for loop
    If Cells(i, 1).Value = "" Then
        Exit For
    End If

    'Determine the type of transport
    If Cells(i, 13).Value = "NULL" Then 'If there is no call to unload,
    'there is no unloading
        If Cells(i, 45).Value = "NULL" Then 'If there is no call to load,
        'there is also no loading
            typeOfTransport = "NoLoadNoUnload"
        Else
            typeOfTransport = "LoadNoUnload"
        End If
    ElseIf Cells(i, 45).Value = "NULL" Then 'When there is unloading, check if
    'there is loading
        typeOfTransport = "NoLoadUnload"
    Else
        typeOfTransport = "LoadUnload"
    End If

    'Determine time on site (hours) (departure minus arrival time)
    calcArray(1, 1) = (Cells(i, 58).Value - Cells(i, 1).Value) * 24

    'Determine processing and waiting times based on type of transport
    Select Case typeOfTransport
        Case "LoadUnload"
            If Cells(i, 9) = "N" Then 'Check if there was no split in loading
            'and unloading (N)
                calcArray(1, 2) = (Cells(i, 51).Value - Cells(i, 13).Value)
                * 24 'processing time
```

```
                calcArray(1, 3) = (Cells(i, 13).Value - Cells(i, 1).Value)
                * 24 'wait to be processed time
            Else
                calcArray(1, 2) = (Cells(i, 51).Value - Cells(i, 45).Value)
                * 24 + (Cells(i, 19).Value - Cells(i, 13).Value)
                * 24
                calcArray(1, 3) = (Cells(i, 13).Value - Cells(i, 1).Value)
                * 24 + (Cells(i, 45).Value - Cells(i, 19).Value)
                * 24
            End If
            calcArray(1, 4) = (Cells(i, 58).Value - Cells(i, 51).Value)
            * 24 'wait for departure time
        Case "LoadNoUnload"
            calcArray(1, 2) = (Cells(i, 51).Value - Cells(i, 45).Value)
            * 24 'processing time
            calcArray(1, 3) = (Cells(i, 45).Value - Cells(i, 1).Value)
            * 24 'wait to be processed time
            calcArray(1, 4) = (Cells(i, 58).Value - Cells(i, 51).Value)
            * 24 'wait for departure time
        Case "NoLoadUnload"
            calcArray(1, 2) = (Cells(i, 19).Value - Cells(i, 13).Value)
            * 24 'processing time
            calcArray(1, 3) = (Cells(i, 13).Value - Cells(i, 1).Value)
            * 24 'wait to be processed time
            calcArray(1, 4) = (Cells(i, 58).Value - Cells(i, 19).Value)
            * 24 'wait for departure time
        Case "NoLoadNoUnload"
            calcArray(1, 2) = 0 'processing time
            calcArray(1, 3) = 0 'wait to be processed time
            calcArray(1, 4) = calcArray(1, 1) 'wait for departure time
        End Select

    'Determine difference between departure and planned departure time
    'Based on how the planned time was registered in the system (forseen date
    'or first planning) determine the 'difference
    If Cells(i, 35).Value = "NULL" Then
        If Cells(i, 60).Value = "NULL" Then
            calcArray(1, 5) = 99999999999#
        Else: calcArray(1, 5) = (Cells(i, 58).Value - Cells(i, 60).Value) * 24
        End If
    Else: calcArray(1, 5) = (Cells(i, 58).Value - Cells(i, 35).Value) * 24
    End If

    'Write results to sheet
    Cells(i, output).Value = typeOfTransport
    Cells(i, output + 1).Value = calcArray(1, 1)
    Cells(i, output + 2).Value = calcArray(1, 2)
    Cells(i, output + 3).Value = calcArray(1, 3)
    Cells(i, output + 4).Value = calcArray(1, 4)
    Cells(i, output + 5).Value = calcArray(1, 5)
    Cells(i, output + 6).Value = Format(Cells(i, 58).Value, "d/mm/yyyy h:mm")
Next i

'Updates all fields to include calculated fields
```

```
    Range("A1").Select
    Columns.AutoFit

    Application.EnableEvents = True
    Application.ScreenUpdating = True

    'Below the indexes are displayed
    '1 = Arrival
    '9 = Split Load Unload
    '58 = Departure
    '13 = Call Dock To Unload
    '19 = Free Dock Unload
    '45 = Call Dock To Load
    '51 = Free Dock Load
    '35 = Forseen Date
    '60 = First Planing
End Sub
```

Listing 3: Script to calculate extra data fields for WMS in VBA

```
'This method determines the out times for owned vehicles by checking there next occurance
'This method requires data fields to be added such as the plant (see indexes below)
Sub DetermineOutTimes()
    'Prevent screenupdating to improve performance
    Application.ScreenUpdating = False

    'Initialize variables
    Dim i As Long 'iterator for shipments

    'Temporary variables for data storage of next shipment
    Dim temp_arr As Double
    Dim temp_dept As Double
    Dim temp_vehicle As String
    Dim temp_loc As String

    Dim array_size As Long 'determines size of all arrays
    array_size = 1 'Initial array size
    Dim trail_name_array() As String 'array used to store names of vehicles
    Dim trail_arr_array() As Double 'array used to store departure times of vehicles
    Dim trail_loc_array() As String 'array used to store locations of vehicles
    Dim found_index As Long
    Dim j As Long 'iterator for trailer arrays
    ReDim trail_name_array(array_size)
    ReDim trail_arr_array(array_size)
    ReDim trail_loc_array(array_size)
    Dim output As Long
    Dim temp_dest As String
    output = 73 'Column location to start writing data

    'Place headers
    i = 1
    Cells(1, output).Value = "NEXT_PLANT"
    Cells(1, output + 1).Value = "NEXT_ARRIVAL_TIME"
    Cells(1, output + 2).Value = "OUT_TIME"
```

```
    Cells(i, output + 3).Value = "NEXT_DESTINATION"

    'Loop through all values from the end to the beginning
    For i = Rows.Count To 2 Step -1

    'If the row is empty or the vehicle is not owned (No) do nothing
    If Cells(i, 1).Value = "" Or Cells(i, 8).Value = "No" Then
        'Do nothing
    Else
        'Check for double shipments
        If Cells(i, 1).Value = temp_arr And Cells(i, 5) = temp_vehicle Then
            'Mark as double shipment
                Cells(i, output).Value = "DOUBLE_SHIPMENT"
                Cells(i, output + 1).Value = "DOUBLE_SHIPMENT"
                Cells(i, output + 2).Value = "DOUBLE_SHIPMENT"
                Cells(i, output + 3).Value = "DOUBLE_SHIPMENT"
        Else
            'Temporarily store arrival time and vehicle
            temp_arr = Cells(i, 1).Value
            temp_vehicle = Cells(i, 5).Value
            temp_dept = Cells(i, 58).Value
            temp_dest = Cells(i, 57).Value
            temp_loc = Cells(i, 62).Value

           'Search for the trailer in trailer array using IsInArray function
            found_index = IsInArray(temp_vehicle, trail_name_array)
            If found_index = -1 Then '-1 equals not found so add vehicle to array
                'increase array size, resize arrays and add trailer as new trailer
                array_size = array_size + 1
                ReDim Preserve trail_name_array(array_size)
                ReDim Preserve trail_arr_array(array_size)
                ReDim Preserve trail_loc_array(array_size)
                Cells(i, output).Value = "LAST_RECORD_OF_TRAILER"
                Cells(i, output + 1).Value = "LAST_RECORD_OF_TRAILER"
                Cells(i, output + 2).Value = "LAST_RECORD_OF_TRAILER"
                Cells(i, output + 3).Value = "LAST_RECORD_OF_TRAILER"
                found_index = array_size
            Else 'Vehicle has been found so retrieve data from next shipment
                Cells(i, output).Value = trail_loc_array(found_index)
                Cells(i, output + 1).Value = trail_arr_array(found_index)
                Cells(i, output + 2).Value = (trail_arr_array(found_index) - temp_dept) * 24
                Cells(i, output + 3).Value = temp_dest
            End If
            trail_name_array(found_index) = temp_vehicle
            trail_arr_array(found_index) = temp_arr
            trail_loc_array(found_index) = temp_loc
        End If
    End If
    Next i

    Application.ScreenUpdating = True

    'Below the indexes are displayed
```

```
    '1 = Arrival
    '5 = Vehicle
    '58 = Departure
    '13 = Call Dock To Unload
    '19 = Free Dock Unload
    '45 = Call Dock To Load
    '51 = Free Dock Load
    '35 = Forseen Date
    '60 = First Planning
    '62 = Plant
    '57 = Load customer name (i.e. Destination)
End Sub

'Function that checks whether a value exists in a given array, and if so, it returns its index
Function IsInArray(stringToBeFound As String, arr As Variant) As Long
    Dim index As Long
    'Default return value if not in array equals -1
    IsInArray = -1

    'Checks entire array for the string and returns index if its found
    For index = LBound(arr) To UBound(arr)
        If StrComp(stringToBeFound, arr(index), vbTextCompare) = 0 Then
            IsInArray = index
            Exit For
        End If
    Next index
End Function
```

Listing 4: Script to calculate next destination and out data fields in VBA

Appendix E

Example of individual

In Table E.1 an example individual is given. An individual is a configuration in which each of the defined decision variables in Section 3.2.4 has an allowed value as denoted in the feasible parameter ranges in Appendix B. The notation used in Table E.1 uses the notations $r_{veh,i,v}$, $m_{i,v}$ and $r_{park,i}$ which are the respective vehicles of type v at plant i, the multiplication factor of external vehicles of type v at plant i and the number of parking places at plant i. Where i is either 1 (Leuven), 2 (Jupille) or 3 (Hoegaarden) and v is either 1 (trailer) or 2 (tanker).

Table E.1: Example individual with description of chromosomes

Decision variable (chromosome)	Description	Example value
$r_{veh,1,1}$	Number of trailers in Leuven	100
$r_{veh,1,2}$	Number of tankers in Leuven	34
$m_{1,1}$	External trailers multiplier Leuven	920
$m_{1,2}$	External tankers multiplier Leuven	1040
$r_{veh,2,1}$	Number of trailers in Jupille	90
$r_{veh,2,2}$	Number of tankers in Jupille	15
$m_{2,1}$	External trailers multiplier Jupille	997
$m_{2,2}$	External tankers multiplier Jupille	907
$r_{veh,3,1}$	Number of trailers in Hoegaarden	32
$r_{veh,3,2}$	Number of tankers in Hoegaarden	10
$m_{3,1}$	External trailers multiplier Hoegaarden	1060
$m_{3,2}$	External tankers multiplier Hoegaarden	1100
$r_{park,1}$	Number of parking places Leuven	108
$r_{park,2}$	Number of parking places Jupille	50
$r_{park,3}$	Number of parking places Hoegaarden	30

Appendix F

Interface code

In this appendix the Python code for the interfaces of the optimisation model is shown. The code can be found in Listing 5.

```
random.seed(64)

# Input variable bounds
# (TRLVown, TALVown, EXTTRLV, EXTTALV, TRJUPown, TAJUPown, EXTTRJUP, EXTTAJUP,
# TRHOEown, TAHOEown,
# EXTTRHOE, EXTTAHOE, parkLV, parkJUP, parkHoe )
lbounds = [80, 20, 800, 800, 80, 1, 800, 800, 20, 1, 800, 108, 54, 30] # Lower
# bounds of input variables
ubounds = [150, 60, 1200, 1200, 150, 20, 1200, 1200, 70, 20, 1200, 128, 74, 50]
# Upper bounds of input variables

# Cost parameters
params = [2050, 1500, 40, 40, 500, 2000, 200, 200] # Cost per owned trailer, cost
# per owned tanker,
# cost per external trailer ride, external tanker ride, defects, additional parking
# spots, cost missed calls, cost  of parking limit reached

# The main simulation function that runs a simulation using the inputs of the
# individual, furthermore it requires
# a reference to the right jar-file, config-file and parameters required for
# evaluation
# This function also calculates the scaled external arrivals
def simulate(individual, jar, config, params, count, ext):
    siminp =''
    # override the multiplier of external arrivals based on the individuals
    # chromosomes
    ext.set_value(0,'multiplier',float(individual[2])/1000)
    ext.set_value(1,'multiplier',float(individual[3])/1000)
    ext.set_value(2,'multiplier',float(individual[6])/1000)
    ext.set_value(3,'multiplier',float(individual[7])/1000)
    ext.set_value(4,'multiplier',float(individual[10])/1000)

    # Define input variables using the individuals chromosomes
    siminp += runCreator(individual)
    # Create inputs for external arrivals based on chromosome parameters
    for y in range(ext.shape[0]):
```

```
            siminp += calcArrivals(ext.values[y])

        # Run the simulation with new parameters and retrieve the output
        print('Start simulation ' + str(count))
        getsimresults =  runsimulation(siminp, jar, config)

        # Process simulation outputs to make them usable in Python
        procout = procOutputs(getsimresults)

        # Evaluate the fitness of the individiual using the simevaluate function
        individual.fitness.values = toolbox.simevaluate(individual, procout, params)

        # Print outputs and return values
        print('simulation ' +str(count) + ' with inputs: '  + str(individual) +
        ' evaluated. Outputs: ' + str(procout[0]) +
        ' Score: ' + str(individual.fitness.values[0]))
        return [procout[0], procout[1], individual.fitness.values]

# Run the simulation as an external process and return its textform output
def runsimulation(inputs, jar_file, config_file):
        print(config_file)
        p = Popen(['java', '-jar', jar_file, config_file,'-s', '-h'],stdin=PIPE,
        stdout=PIPE, stderr=STDOUT, universal_newlines=True)
        out, err = p.communicate(input=inputs)
        return out

# This function calculates the non-homogenous poisson process parameters based on
# quarterly arrivals and ratio's
# It uses an input array
def calcArrivals(inputs):
        # sout is used as string variable which will be passed to the simulation model
        sout = "\n "+str(inputs[0]) + ' Value { {0 h 0} '
        v_temp = 0 # initial cumulative amount of shipments
        count = 0
        wd = 0 # weekday variable where values 1..5 equal weekdays and 6 and
        # 7 equal weekend
        dl = [inputs[2], inputs[3], inputs[1]] # daytime
        m = inputs[7] # scaling multiplier

        # For each timestep calculate the expected cumulative number of shipments
        for d in range(1,92):
            wd = wd + 1

            # if weekday:
            if wd < 6:
                w_temp = inputs[5]
                wd_temp = 5
            else:
                w_temp = inputs[4]
                wd_temp = 2
            # reset weekday counter on day 7
            if wd == 7:
                wd = 0
```

```
        # for each
        for j in range(1,4):
            count = count + 1
            s = inputs[6]
            # Calculate the expected cumulative arrivals at time t
            v_temp = v_temp + m*(s * dl[j-1] * w_temp)/(92*(wd_temp/7))
            # Add to the output file
            sout+='{ '+str(count*8)+' h '+ str(v_temp) +' } '
    sout+='}'
    #print(sout)
    return sout

# Creates input string that can be piped to the simulation software with the
# different values
def runCreator( inputs ):
    lines="\n"
    #lines+="Include 'Sim model - Trailer Management - V1-0.inc'\n"
    lines+="TrailersLV Value { " +str(inputs[0])+" }\n"
    lines+="TankersLV Value { " +str(inputs[1])+" }\n"
    lines+="TrailersJUP Value { " +str(inputs[4])+" }\n"
    lines+="TankersJUP Value { " +str(inputs[5])+" }\n"
    lines+="TrailersHOE Value { " +str(inputs[8])+" }\n"
    lines+="TankersHOE Value { " +str(inputs[9])+" }\n"
    lines+="ParkLV Capacity { " + str(inputs[11])+ " }\n"
    lines+="ParkJUP Capacity { " + str(inputs[12])+ " }\n"
    lines+="ParkHOE Capacity { " + str(inputs[13])+ " }\n"
    lines=lines[:-1]
    return lines;

# takes the inputs, outputs and cost parameters to determine the total cost function
# parameters should be passed as a list [cost of: owned trailer, owned tanker,
# external trailer ride, external tanker ride, defects]
def costs(inputs, outputs, params, mc_cost, pl_cost):
    tr_lv = inputs[0]
    ta_lv = inputs[1]
    tr_jup = inputs[4]
    tr_hoe = inputs[8]
    ta_jup = inputs[5]
    ta_hoe = inputs[9]
    # costs related to owned/rented trailers
    tr_costs = (tr_lv+tr_jup+tr_hoe)*params[0]
    # costs related to owned/rented tankers
    ta_costs = (ta_lv+ta_jup+ta_hoe)*params[1]
    # costs related to external shipments
    ext_costs = outputs[7]*params[2]+outputs[8]*params[3]
    # costs related to defects
    d_costs = outputs[6]*params[4]
    # costs related to extra parking
    ep_costs = (inputs[11]+inputs[12]+inputs[13])*params[5]

    # determine the total amount of delayed calls and parking overloads
    mcalls = outputs[0]+outputs[1]+outputs[2]
    plimit = outputs[3]+outputs[4]+outputs[5]
```

```
    # determine the costs
    mc_pen = mcalls*mc_cost
    pl_pen = plimit*pl_cost
    print('Trailers: ' + str(tr_costs) +', Tankers: ' + str(ta_csts) +
    ', Defects: ' + str(d_costs) + ', Extra parking: ' + str(ep_costs) + ',
    MC penalty: ' + str(mc_pen) + ' and PL penalty: '+ str(pl_pen))
    total = tr_costs+ta_costs+d_costs+ext_costs+pl_pen+mc_pen # total costs
    return total

def simevaluate(individual, simoutputs, params):
    return costs(individual, simoutputs, params),

def createArrivalsFile(file, output_name):
    arrivalsinp = ''
    demands = pd.read_excel(file, sheetname="arrivals", converters =
    {'multiplier': float})
    externals = pd.read_excel(file, sheetname="externals", converters =
    {'multiplier': float})

    # For every entry in the excel sheet create the JaamSim text equivalent for
    # order arrivals
    for k in range(demands.shape[0]):
        arrivalsinp += calcArrivals(demands.values[k])

    # For every entry in the excel sheet create the JaamSim text equivalent for
    # vehicle arrivals
    for a in range(externals.shape[0]):
        arrivalsinp += calcArrivals(externals.values[a])

    fed= open(output_name,"w+")
    fed.write(arrivalsinp)

def createTransitionsFile(file, output_name):
    transitions = pd.read_excel(file, sheetname="transitions", converters =
    {'LV': float, 'JUP': float, 'HOE': float})
    discrts = "Define DiscreteDistribution { "
    trans = "\n"
    transcnt = 500 # initial value for seeds for transition variables
    transoutput = ""

    # For every entry in the excel file create a JaamSim entry in text form
    for tr in range(transitions.shape[0]):
        discrts += transitions.values[tr][0]+" "
        oglv = transitions.values[tr][1]
        ogjup = transitions.values[tr][2]
        oghoe = transitions.values[tr][3]
        trans+= transitions.values[tr][0]+ " UnitType { DimensionlessUnit }\n"
        trans+= transitions.values[tr][0]+ " RandomSeed { "+str(transcnt)+" }\n"
        trans+= transitions.values[tr][0]+ " ValueList { 1 2 3 }\n"
        trans+= transitions.values[tr][0]+ " ProbabilityList { "+str(oglv)+" "+
        str(ogjup)+" "+str(oghoe)+" }\n\n"
        transcnt += 1
    discrts += "}\n"
    transoutput += discrts + trans
```

```
    f = open(output_name + ".inc","w+")
    f.write(transoutput)

# Function to create the out times file, inputs are the filename of the
# masterdata file (as string) and a string for the output name
def createOutTimeFile(file, output_name):

    times = pd.read_excel(file, sheetname="OutTimes", converters =
    {'Variable1': float, 'Variable2': float})
    probs = pd.read_excel(file, sheetname="OutProbabilities", converters =
    {'Value1': float, 'Value2': float, 'Value3': float ,'Value4': float,
    'Value5': float})

    seedcnt = 300
    simot = "\n"
    probot = "\n"
    gam = "Define GammaDistribution { "
    lnorm = "Define LogNormalDistribution { "
    norm = "Define NormalDistribution { "
    discr = "Define DiscreteDistribution { "

    for dist in range(times.shape[0]):
        # Out time format for lognormal distribution
        if times.values[dist][4]=="Lognormal":
            simot += times.values[dist][0] +" UnitType { TimeUnit }\n"
            simot += times.values[dist][0] +" RandomSeed { "+str(seedcnt)+" }\n"
            simot += times.values[dist][0] +" MinValue { 0 h }\n"
            simot += times.values[dist][0] +" Scale { 1 h }\n"
            simot += times.values[dist][0] +" NormalMean { "+
            str(times.values[dist][2])+" }\n"
            simot += times.values[dist][0] +" NormalStandardDeviation
            { "+str(times.values[dist][3])+" }\n"
            simot += "\n"
            lnorm += times.values[dist][0]+" "

        # Out time format for gamma distribution
        elif times.values[dist][4]=="Gamma":
            simot += times.values[dist][0] +" UnitType { TimeUnit }\n"
            simot += times.values[dist][0] +" RandomSeed { "+str(seedcnt)+" }\n"
            simot += times.values[dist][0] +" MinValue { 0 h }\n"
            simot += times.values[dist][0] +" Mean { "+str(times.values[dist][6])+
            " h }\n"
            simot += times.values[dist][0] +" Shape { "+str(times.values[dist][3])+
            " }\n"
            simot += "\n"
            gam += times.values[dist][0]+" "

        # Out time format for normal distribution
        elif times.values[dist][4]=="N":
            simot += times.values[dist][0] +" UnitType { TimeUnit }\n"
            simot += times.values[dist][0] +" RandomSeed { "+str(seedcnt)+" }\n"
            simot += times.values[dist][0] +" MinValue { 0 h }\n"
            simot += times.values[dist][0] +" Mean { "
            +str(times.values[dist][2])+" h}\n"
```

```
            simot += times.values[dist][0] +" StandardDeviation {
            "+str(times.values[dist][3])+" h }\n"
            simot += "\n"
            norm += times.values[dist][0]+" "
        seedcnt += 1

    # Format for the respective discrete probability functions for each of the
    # distributions
    for pb in range(probs.shape[0]):
        probot += probs.values[pb][0] +" UnitType { DimensionlessUnit } \n"
        probot += probs.values[pb][0] +" RandomSeed { "+str(seedcnt)+" } \n"
        probot += probs.values[pb][0] +" ValueList { 1 2 3 4 5 } \n"
        probot += probs.values[pb][0] +" ProbabilityList { "
        +str(probs.values[pb][1])+
        " "+str(probs.values[pb][2])+" "+str(probs.values[pb][3])+" "
        +str(probs.values[pb][4])+" "+str(probs.values[pb][5])+" }\n\n"
        discr += probs.values[pb][0]+" "
        seedcnt += 1
    gam += " }\n"
    lnorm += " }\n"
    norm += " }\n"
    discr += " }\n"

    # Combine all seperate JaamSim text in one joined text file
    outstring = gam + lnorm + norm + discr + simot + probot

    f = open(output_name,"w+")
    f.write(outstring)
```

Listing 5: Coded interfaces between the simulation model and the optimization model in Python

```
def transChecker(transitions, demands, externals):
    dlv_tr = demands.values[0][6]
    dlv_ta = demands.values[1][6]
    djup_tr = demands.values[3][6]
    djup_ta = demands.values[4][6]
    dhoe_tr = demands.values[6][6]
    dhoe_ta = demands.values[7][6]

    elv_tr = externals.values[0][6]
    elv_ta = externals.values[1][6]
    ejup_tr = externals.values[2][6]
    ejup_ta = externals.values[3][6]
    ehoe_tr = externals.values[4][6]

    lv_tr_own = dlv_tr - elv_tr
    lv_ta_own = dlv_ta - elv_ta

    jup_tr_own = djup_tr - ejup_tr
    jup_ta_own = djup_ta - ejup_ta

    hoe_tr_own = dhoe_tr - ehoe_tr
    hoe_ta_own = dhoe_ta
```

```
lvtr = (lv_tr_own) - (lv_tr_own)*transitions[0]-(jup_tr_own)*transitions[3]-
(hoe_tr_own)*transitions[6]
juptr = (jup_tr_own) - (lv_tr_own)*transitions[1]-(jup_tr_own)*transitions[4]-
(hoe_tr_own)*transitions[7]
hoetr = (hoe_tr_own) - (lv_tr_own)*transitions[2]-(jup_tr_own)*transitions[5]-
(hoe_tr_own)*transitions[8]
lvta = (lv_ta_own) - (lv_ta_own)*transitions[9]-(jup_ta_own)*transitions[12]-
(hoe_ta_own)*transitions[15]
jupta = (jup_ta_own) - (lv_ta_own)*transitions[10]-(jup_ta_own)*transitions[13]-
(hoe_ta_own)*transitions[16]
hoeta = (hoe_ta_own) - (lv_ta_own)*transitions[11]-(jup_ta_own)*transitions[14]-
(hoe_ta_own)*transitions[17]
return lvtr, juptr, hoetr, lvta, jupta, hoeta
```

Listing 6: Code to check the balance of transitions between sites

Appendix G

Optimisation algorithm

In this section the underlying code of the optimisation algorithm is shown and discussed. This includes running initial simulations, the genetic algorithm setup and the finding and training of neural networks. The full code can be found in Listing 7.

```
# Create an initial population to use in the optimisation algorithm
def initialSamples(sampleSize)
    pop = toolbox.population(n=sampleSize) # Create initial population
    popout = []
    popstd = []
    popsol = []
    popres = []
    dectrees = []

    # Run simulations for the original population
    for i in range(len(pop)):
        unique=True
        count = i+1
        # Check whether an identical simulation has
        # been run before
        try:
            loc=popsol.index(list(pop[i]))
        except ValueError:
            "Do Nothing"
        # If not unique then
        else:
            ind.fitness.values = popres[loc]
            print('Found identical value')
            unique=False
        # Else run the simulation
        if unique:
            simresults = simulate(pop[i], jar, config,
                params, count, externals)
        popsol.append(list(pop[i]))
        popout.append(simresults[0])
        popstd.append(simresults[1])
        popres.append(simresults[2])
        pop[i].fitness.values = simresults[2]

    # Save the outputs to temporary variables (allows
```

```
    # easier reusage)
    s_popsol = copy.deepcopy(popsol)
    s_popres = copy.deepcopy(popres)
    s_popout = copy.deepcopy(popout)
    s_dectrees = copy.deepcopy(dectrees)
    s_pop = copy.deepcopy(pop)

    # Optionally save the inputs and outputs to file
    dfsol = pd.DataFrame(popsol)
    dfout = pd.DataFrame(popout)
    dfres = pd.DataFrame(popres)
    dfstd = pd.DataFrame(popstd)
    dfsol.to_csv('popsol'+str(sampleSize)+'init.csv',
    index=False, header=False, sep=';', decimal=',')
    dfstd.to_csv('popstd'+str(sampleSize)+'init.csv',
    index=False, header=False, sep=';', decimal=',')
    dfout.to_csv('popout'+str(sampleSize)+'init.csv',
    index=False, header=False, sep=';', decimal=',')
    dfres.to_csv('popresults'+str(sampleSize)+'init.csv',
    index=False, header=False, sep=';', decimal=',')

from sklearn.grid_search import GridSearchCV

# Perform parameter tuning on a neural nework
def tune_params(model, paramgrid, inputs, outputs):
    print("Gridsearch for optimal model ....")
    x_tr, x_tst, y_tr, y_tst = train_test_split(inputs,
            outputs, test_size = 0.2, random_state=0)
    gs = GridSearchCV(model, param_grid=paramgrid, cv=10 )
    gs.fit(x_tr, y_tr)
    print("Best params: " + str(gs.best_params_))
    y_true, y_pred = y_tst, gs.predict(x_tst)
    return(gs)

print("Starting experiment ...........")
n = 1000 # train set size
m = 100 # population size
print('Train size '+ str(n))
print('Population size: '+ str(m) )
regressor_seed = 12

paramranges = {'alpha' : [0.01, 0.001, 0.0001, 0.00001],
               'hidden_layer_sizes' : [(25,), (50,),
                               (50,5), (25,15),]}


# Setup to easily re-use the saved initial runs
t_popsol=copy.deepcopy(s_popsol)
split_popsol = t_popsol[:n]
t_popres=copy.deepcopy(s_popres)
n_popsol = [normInputs(sol, lbounds, ubounds) for sol
        in split_popsol]
np_popsol = np.array(n_popsol)
reslist = [value[0] for value in t_popres]
```

```
np_reslist = np.array(reslist[:n])
t_popout=copy.deepcopy(s_popsol[:n])

# Train the meta model

MUTPB = 0.2
ENSUREPB = 0.01
elt = 2

# Add the used individuals, outputs and results
# to the lists
pop = []
for i in range(len(t_popsol[:n])):
    ind = creator.Individual(t_popsol[i])
    ind.fitness.values=t_popres[i]
    pop.append(ind)

popsol = copy.deepcopy(t_popsol[:n])
popres = list(np_reslist)
popout = t_popout[:n]

neural = MLPRegressor(solver='lbfgs', alpha=1e-05,
        activation='relu', hidden_layer_sizes=(5,),
                random_state=regressor_seed)
dectr = tree.DecisionTreeRegressor(random_state=regressor_seed)
pophistory = []

popsol=copy.deepcopy(s_popsol[:n])
popres=copy.deepcopy(s_popres[:n])
popout=copy.deepcopy(s_popout[:n])

# Add the used individuals, outputs and results to the lists

pop = []
for i in range(len(popsol)):
    ind = creator.Individual(popsol[i])
    ind.fitness.values=popres[i]
    pop.append(ind)

# Train the first meta model
neurals=[]
dectrees=[]

neural, dectr = trainModels(popsol, popres, neural,
        dectr, paramranges)
neurals.append(neural)
dectrees.append(dectr)
NRTRAIN=1
count = n

unique=True # Set unique flag to true
# set skip and total skip counters to 0
skips, totalskips = 0, 0
bests = []
```

```
poppred = []
poptrpred = []
bscore_so_far = 999999999 # High initial value
threshold = 100000

pop = toolbox.selbest(pop, k=m) # Create the population

print("Start genetic algorithm")
simcount = 0
g = 0
# After the inititial populaton new generations will be derived
while g < 30 and simcount < 300:
    print('--Generation '+str(g+1)+'--' )

    # Select the individuals with best fitness using
    # the selection method
    best = toolbox.selbest(pop) # selects best individual
    # list with best individuals per generation
    bests = bests + best n
    if best[0].fitness.values[0] < bscore_so_far:
        bscore_so_far = best[0].fitness.values[0]
    print('The best score so far: '+str(bscore_so_far))

    # select the elite
    elite = toolbox.selbest(pop, k=elt)
    # select the invididuals that will be used
    # for getting the next generation
    selected = toolbox.select(pop,len(pop))
    offspring = [toolbox.clone(ind) for ind in selected]

    # Apply genetic operations - mutation and cross-over
    offspring = genOperations(offspring, CXPB, MUTPB)

    # For all individuals without fitness value
    invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
    if len(invalid_ind) > 0:

        # Create normalized inputs from individuals
        n_invalid_ind = [normInputs(inv, lbounds, ubounds) for
                inv in invalid_ind]

        # Use these inputs to predict scores
        neuralfit = neurals[NRTRAIN-1].predict(n_invalid_ind)
        treefit = dectrees[NRTRAIN-1].predict(invalid_ind)

        # Now for each individual and its score
        for ind, fit, trfit in zip(invalid_ind, neuralfit, treefit):

            # if approximated fitness is in the acceptable range,
            # run the simulation and use the true fitness
            epb = random.random()
            if (fit < bscore_so_far + threshold) or (epb < ENSUREPB):
                print('There have been '+ str(skips)+ ' skips.')
                try:
```

```
                    loc=popsol.index(list(ind))
                except ValueError:
                    "Do Nothing"
                else:
                    ind.fitness.values = (popres[loc][0], )
                    print('Found identical value')
                    unique=False
                if unique:
                    count += 1
                    sim_start_time = time.process_time()
                    simresults = simulate(ind, jar, config, params, count, externals)
                    print('NN prediction: '+ str(fit) + ' Ensure probability: ' +
                        str(epb))
                    total_sim_time += time.process_time() - sim_start_time
                    popout.append(simresults[0])
                    popstd.append(simresults[1])
                    popres.append(simresults[2])
                    popsol.append(list(ind))
                    poppred.append(fit)
                    simcount += 1
                    poptrpred.append(trfit)
                    ind.fitness.values = simresults[2]
                    if simresults[2][0] < bscore_so_far:
                        bscore_so_far = simresults[2][0]
                unique=True
                #print('Predicted value: '+ str(fit))
                skips = 0
            else:
                # Else use the approximated fitness as its "true" fitness
                ind.fitness.values = fit,
                skips+=1

    # Train new model after the generation
    start_train = time.process_time()
    NRTRAIN+=1
        neural = MLPRegressor(solver='lbfgs', alpha=1e-05, activation='relu',
                hidden_layer_sizes=(5,), random_state=regressor_seed)
    dectr = tree.DecisionTreeRegressor(random_state=regressor_seed)
    neural, dectr = trainModels(popsol, popres, neural, dectr, paramranges)
    neurals.append(neural)
    dectrees.append(dectr)
        total_tr_time += time.process_time() - start_train
        print('Model number '+str(NRTRAIN)+' is trained')

    # Define the new population using the elite individuals and
    # the population_size - elite_size best individuals
    nextpop = toolbox.selbest(offspring+elite, k=len(pop))
    pop[:] = [toolbox.clone(ind) for ind in nextpop]
    pophistory.append(["Next Generation"])
    pophistory.append(pop)
    g += 1

dfsol = pd.DataFrame(popsol)
bestlist = [list(value) for value in bests]
```

```
dfbest = pd.DataFrame(bestlist)
outlist = [list(value) for value in popout]
dfout = pd.DataFrame(outlist)
reslist =  [value[0] for value in popres]
dfres = pd.DataFrame(reslist)
dfpred = pd.DataFrame(poppred)

print("Writing outputs to files ....")
dfbest.to_csv('popbest-gmmodel-pop'+str(m)+'-thresh'+str(threshold)+
    '-elt'+str(elt)+'-mp'+str(MUTPB)+'-trainsize'+str(n)+'.csv', index=False,
    header=False, sep=';', decimal=',')
dfsol.to_csv('popsol-gmmodel-pop'+str(m)+'-thresh'+str(threshold)+
    '-elt'+str(elt)+'-mp'+str(MUTPB)+'-trainsize'+str(n)+'.csv', index=False,
    header=False, sep=';', decimal=',')
dfpred.to_csv('poppred-gmmodel-pop'+str(m)+'-thresh'+str(threshold)+
    '-elt'+str(elt)+'-mp'+str(MUTPB)+'-trainsize'+str(n)+'.csv', index=False,
    header=False, sep=';', decimal=',')
dfout.to_csv('popout-gmmodel-pop'+str(m)+'-thresh'+str(threshold)+
    '-elt'+str(elt)+'-mp'+str(MUTPB)+'-trainsize'+str(n)+'.csv', index=False,
    header=False, sep=';', decimal=',')
dfres.to_csv('popres-gmmodel-pop'+str(m)+'-thresh'+str(threshold)+
    '-elt'+str(elt)+'-mp'+str(MUTPB)+'-trainsize'+str(n)+'.csv', index=False,
    header=False, sep=';', decimal=',')
```

Listing 7: Code for the optimisation algorithm in Python