Faculty of Technology, Policy and Management

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
Master of Science

Operational Scheduling in a Multi-Actor Environment using Multiagent Systems

A case study on Multiprocessor Task Problems in Liquid Bulk Terminals

MSc Dissertation of:
Matthijs Brouns

Chair:
Prof. Dr. Ir. C. Chorus

First Supervisor:
Dr. H. Aldewereld

Second Supervisor:
Dr. S. van Cranenburgh

Third Supervisor:
Dr. M. de Weerdt

External Supervisor:
M. Jansen MSc

September 2015

Preface

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science in *Systems Engineering, Policy Analysis and Management*. It contains work done from February to September 2015.

This project was conceived during my time working for Systems Navigator. Systems Navigator was working on a decision support system to aid the scheduling staff of liquid bulk terminals in creating schedules. During a meeting with my supervisor, Huib Aldewereld, he proposed to assess whether Multiagent systems could be successfully employed to create scheduling options and propose these to planners working at liquid bulk terminals.

Writing this thesis and doing the related research has been a hard, but satisfying process in which many of the basic aspects of Multiagent systems - cooperation and negotiation - were practised.

I would like to thank my supervisor, Dr. Huib Aldewereld, for the inspiration and guidance he has offered in the process. Thanks also to Prof. Caspar Chorus and Dr. Sander van Cranenburgh for their immense expertise on the subject of discrete choice modelling and the random regret minimization model especially. Furthermore, lots of thanks are due to Dr. Mathijs de Weerdt, who offered great insights in Multiagent systems, Mechanism Design and Algorithmics, and was willing to remain a supervisor in my graduation committee even though officially not being a part of my faculty.

Furthermore, I would like to thank Mart Jansen, without whom I would not have been able to finish this thesis. His knowledge regarding liquid bulk terminals and scheduling problems in general has proved essential in the emergence of this research. Finally, I also want to thank Systems Navigator and its employees for providing me with the means for doing this research.

Matthijs Brouns
Delft
September 2015

Summary

Liquid bulk is one of the largest industries of the modern world, and efficient scheduling is needed for liquid bulk terminals to remain competitive. A common solution for the planning efficiency problem is to apply planning algorithms, rather than human planners for creating schedules. However, replacing human planners by algorithms is often a difficult problem and its implementation is often obstructed both by management and the planning departments. In order to still achieve efficient schedules, attention has shifted from planning algorithms intended to replace the human planner, to scheduling support systems which aim to assist the planner in making efficient schedules. A core aspect of both automated scheduling systems as well as scheduling support systems is an optimization engine, to allow the human planner to quickly generate alternative schedule options.

In recent years, literature has emerged introducing the concept of Multiagent systems for planning and scheduling. Multiagent scheduling seems to be a suitable approach for scheduling in liquid bulk terminals since the agents provide a natural metaphor for the scheduling problem in a liquid bulk terminal. This is mainly because a terminal serves vessels which are owned by a set of customers which are largely uninterested in the vessels of other customers. Multiagent systems can, in such a scenario, offer flexibility with regards to optimality criteria which traditional scheduling systems cannot.

The main goal of this study was to determine whether Multiagent Systems (MAS) could be effectively applied for proposing schedule allocation options to assist schedulers in liquid bulk terminals.

The research started with an extensive background study, in which it was found that there are a number of stakeholders involved in the scheduling process, which have partially conflicting goals. The main relevant stakeholders are the terminal and its customers, both consisting of several independent departments or entities. The main goals of the terminal are achieving a high terminal utilization, having a low impact on surroundings, having low pressure on operations, and not paying demurrage costs. The main goals from the customer side are achieving short turnaround times, and maintaining product quality. By combining the stakeholder requirements with information regarding the processes, infrastructure, and products at liquid bulk terminals, it was found that the scheduling problem resembles a specific version of the job-shop problem: $JMPT|pmtn, prec, r_i, s_{ij}|multi$. Since this problem is strongly NP-hard it is generally infeasible to find an optimal solution as the size of the problem instance grows, and efficient algorithms are needed to find solutions.

After formally defining the scheduling problem as present in liquid bulk terminals, this thesis moved on to determine the applicability of several common Multiagent scheduling solutions based on four dimensions: whether the solution provides a good analogy to the scheduling problem, whether the solution could deal with the arrival and departure of vessels over time, whether the solution could incorporate shared infrastructure between routes and whether terminal preferences could be modelled. It was found that Combinatorial Auctions provided the best fit on the

scheduling problem. Due to existing contract structures between terminals and their customers it proved impossible to incorporate an additional payment structure in the scheduling problem, making it infeasible to design the auction to be truthful and strategy-proof. Nonetheless, this is not considered a problem as strategic behaviour can easily be verified.

For incorporating multi-attribute optimizations several design directions were explored including Pareto-optima, hard constraints and discrete choice models. It was found that only discrete choice models offered the flexibility needed in this case. Two types of discrete choice models were proposed and implemented: a Utility Maximization model, as well as a Regret Minimization model. Empirical studies have shown that in certain cases, a regret minimizing model provides a significantly better fit on the preference data, meaning that they are better able to capture the trade-offs made by humans when comparing alternatives. As the goal of this scheduling system is to assist the planner in making allocation choices, it can benefit greatly from these features. A major drawback of the regret model is that finding the regret for all alternatives is an $O(n^2)$ operation, rather than an $O(n)$ operation. For solution methods where this posed a bottleneck, a hybrid regret / utility model was proposed and implemented.

The auction structure was implemented using a minor modification on the Contract Net Protocol. The final Multiagent Systems consists of three agent types: a Route Finder agent, a Vessel agent, and a Planner agent. The Route Finder agent is responsible for finding routes for specific vessels and products on the terminal, as well as for making accurate estimations on the processing times on that route. This is implemented using a combination of Yen's algorithm and a modification of Dijkstra's shortest path algorithm. The Vessel agent uses te routes from the Route Finder agent to determine a valuation on the routes and submits these valuations as bids to the Planner agent. The Planner agent retrieves all bids and uses these to create final schedule proposals. This is done by solving the Winner Determination Problem (WDP) of the auction.

Several algorithms were implemented and tested for solving the WDP, including a Branch and Bound algorithm, informed search methods and two Simulated Annealing approaches. It was found that Simulated Annealing, implemented as a search among alternative complete solutions, was the most feasible solution strategy. Typically, it provided solutions which, on average, achieve a 50% higher utility model value than a First-Come First-Served Heuristic in three minutes. Furthermore, the Simulated Annealing process can be stopped at any moment and still provide a better alternative solution.

All algorithms were implemented using a utility-maximising as well as a regret-minimising strategy. It was found that the top ten results from these different models typically differed from each other with an edit distance of five. Further empirical studies would be needed to determine whether regret-based models provide better accepted solutions in practice than their utility-based counterparts.

Contents

Introduction 1

Liquid Bulk is one of the largest industries of the modern world. As of today, there are about 4,000 tank terminals operational with a combined storage capacity of over 800,000,000 m^3 (TankTerminals.com, 2014) and prognoses indicate the market is only expected to grow in the upcoming years (Figure 1.1). The majority of the tanker market consists of crude oil and petroleum products, which make up about a third of all marine trade (United Nations Conference on Trade and Development, 2010).

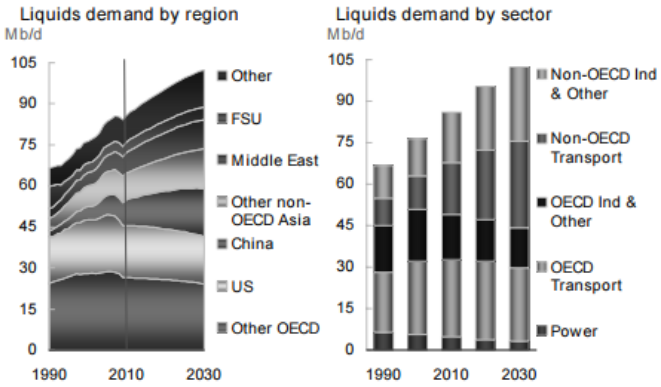


FIGURE 1.1: Expected growth in liquid bulk (Finley, 2012, p. 30)

In this industry, good planning and scheduling is vital for attaining good production results. However, as McKay, Buzacott, and Safayeni (1989) state: "Real world scheduling and planning of jobs often appears to be very unsophisticated, imprecise, prone to error and haphazard", and the liquid bulk industry is no exception to this statement. In many liquid bulk terminals, scheduling is done by using a myriad of tools and utilities such as Microsoft Excel and in-house developed terminal information systems (Diao, 2014). Since these tools are mostly general-purpose tools they offer little to no guidance in the actual scheduling tasks such as berth and infrastructure allocation, or the planning of maintenance slots. The tools that are specifically designed for scheduling liquid bulk are typically scoped to only include berths, thereby releasing land-side infrastructure constraints (e.g. Umang, Bierlaire, & Vacca, 2011; Douma, Schuur, & Schutten, 2011).

Systems Navigator identified several main sources of losses at liquid bulk terminals which originate from this capricious nature of real world scheduling and

planning jobs, which can be divided into information errors and planning inefficiencies (J. Smits, Personal communication, July 29^{th}, 2015).

The first category contains losses which originate from communication errors between for example the terminal and its customers, or the different departments within terminals. Several occasions at terminals were identified in which updates on vessel arrivals were not handled correctly, which resulted in vessels arriving which could not be processed. These types of errors typically result in fines leading up to €1,000,000. Another typical communication error which was identified caused high-quality product to be processed by the same infrastructure as low-quality product, thereby contaminating the high-quality product and reducing its value.

The second category of losses contains losses which are due to planning inefficiencies. Typically, the planning department consists of several people, but only a small part of this department will be able to schedule the use of the terminals' infrastructure at its fullest. This causes longer processing times and thereby extra costs for the terminal.

Scheduling Support in Liquid Bulk

The findings with regards to losses at liquid bulk terminals highlight the need for improving the predictability and consistency of scheduling at liquid bulk terminals. A typical solution strategy for the planning efficiency problems is to replace human planners by planning algorithms. However, replacing human planners by algorithms is often a difficult problem and its implementation is often obstructed by both management and the planning departments. This is mainly because in practice, planners spend much time on collaboration, communication and negotiation about various (soft) constraints and goals (van Wezel & Jorna, 1999, p. 4). This statement is supported by Verbraeck (1991) who states that only 3% of the planners time is actually used for creating a plan. Also, while planning algorithms might be able to solve the planning inefficiency losses, it will not be able to properly deal with the communication error losses. Because of this, we should consider planning as a multi-faceted problem and it is vital for all of these facets to be covered in order to design good planning support systems.

In order to still achieve good planning results, attention has shifted from automated schedule generation which attempt to replace the planners job, to using collaborative *Decision Support Systems* (DSS) or Scheduling Support Systems which attempt to help the planner in creating good schedules. In order to assist the schedulers of tank terminals, Systems Navigator is working on a Decision Support System for Tank Terminals. Systems Navigator sees market opportunities in supporting schedulers of liquid bulk terminals by, among others, suggesting and calculating infrastructure routing options. A scheduling support system differs from regular scheduling systems in that its goal is not to replace the human planner by automation, but rather assisting him in making good schedules.

A core aspect of both automated scheduling systems as well as scheduling support systems is a scheduling algorithm, to allow the human planner to quickly generate

alternative plans for comparison (Pinedo & Chao, 1998). The focus of this thesis will be on such a scheduling algorithm for application in a scheduling support system.

Multiagent Systems for Scheduling

Traditionally, scheduling algorithms are centralized and monolithic systems, which are optimized for a single application (Aldewereld, Dignum, & Hiel, 2012). More recently, literature has emerged introducing the concept of *Multiagent Systems* (MASs) for scheduling. MASs are applied in a number of different domains such as barge planning, manufacturing control, and micro grid scheduling (Aldewereld et al., 2012; Baker, 1998; Douma et al., 2011; Logenthiran, Srinivasan, Khambadkone, & Aung, 2010; Rabelo, Camarinha-Matos, & Afsarmanesh, 1999) often leading to great benefits in areas such as reusability of the algorithms, modularity and composability of its components and parallelizability (Jennings & Bussmann, 2003).

Multiagent scheduling seems to be a suitable approach for scheduling in liquid bulk terminals since the agents provide a natural metaphor for the processes on a terminal (Wooldridge, 2002); as a terminal serves vessels which are owned by a set of customers which are largely uninterested in the vessels of other customers and which might employ difference optimality criteria than other customers. In this scenario, MAS offer flexibility which traditional scheduling systems cannot (Agnetis, Billaut, Gawiejnowicz, Pacciarelli, & Soukhal, 2014).

Even though MAS have been applied for scheduling problems on numerous occasions, there seems to be a knowledge gap in the literature regarding the application of MAS on the scheduling problems in liquid bulk terminals.

Research Objective

Although MAS literature suggests many benefits over traditional scheduling systems, it is still unclear how a MAS should be applied in the domain of liquid bulk scheduling. The main goal of this research is to design a Multiagent Scheduling System to effectively propose schedule allocation options to assist schedulers at liquid bulk terminals. In this context, an effective system refers to a system which generally proposes highly acceptable scheduling options to the planners, since low acceptable solutions offer no benefits to the planner. This design goal corresponds to the following main research question:

- Can a *Multiagent System* (MAS) be effectively applied in a scheduling support system for liquid bulk terminals?

In order to achieve this goal, it is first necessary to get a clear overview of the different elements in liquid bulk scheduling and how they differ from regular scheduling problems. This is the focus of the first four research questions.

- Who are the main stakeholders involved in liquid bulk scheduling?
- What are the requirements for application of a optimization system in a scheduling support system?

- What are the main characteristics of scheduling problems in liquid bulk terminals?
- What are the main differences in applying MAS between liquid bulk scheduling problems and regular job-shop scheduling problems?

These questions will be the objective of Part I of this thesis. Chapter 3 starts by giving a general overview of liquid bulk terminals and its operations, as well as the stakeholders involved in the scheduling process. It then goes into detail on the application of a scheduling algorithm in scheduling support systems, as well as the systems environment in liquid bulk terminals. It ends by giving a full, formal definition of the scheduling problem found in liquid bulk terminals. Then, Chapter 4 will describe the findings of literature research regarding MASs in scheduling, and evaluates the applicability of the found research in the liquid bulk scheduling domain.

The second part of this thesis consists of an iterative design approach in which preliminary designs are generated based on the knowledge gained in the first part, which are then (partially) implemented and tested for feasibility. The main goal of this part is to answer the following research question:

- How can the differences in applying MAS between liquid bulk scheduling and regular job-shop scheduling problems be overcome in practice?

The first chapter of Part II will describe the basic functioning of the MAS using the Gaia Agent Design methodology. The subsequent chapters of Part II will go in depth on the design and build of the individual agent types present in the MAS.

Finally, in order to decide whether the application of MAS can be considered to be effective, the implementation needs to be assessed on a number of criteria. In order to answer this question a prototype implementation of a scheduling MAS will be made for application in liquid bulk terminals and compared to existing knowledge available from Systems Navigator about scheduling algorithms. The prototype implementation will be compared to schedules made by human planners. This evaluation phase is described in Part III of this thesis and answers the final research question:

- To what extent is a MAS-based scheduling system able to assist the human planner with regards to run-time and quality of results?

Research Methodology 2

The main goal of this research is to design a system for assisting schedulers at liquid bulk terminals by proposing scheduling options. To do this, a *Multiagent System* (MAS) will be designed, implemented, and tested. This MAS is to be a part of a larger decision support system for scheduling of liquid bulk terminals. This thesis will describe all the steps from conceptualization of the system up to the design and implementation.

This chapter proposes a structured research approach for designing this scheduling system. The approach is based on a combination of the five stage prescriptive model by Dym and Little (2008), and the "V-Model" for software development. The five stage model (Figure 2.1) prescribes five distinct stages in engineering design: the problem definition, conceptual design, preliminary design, testing, and detailed design. The V-Model combines a top down design approach with a bottom-up implementation approach. The top-down part is often referred to as the project definition phase in which requirements are set and a design is drafted. the bottom-up part is a testing phase in which the implementation is compared with the results from the project definition phase.

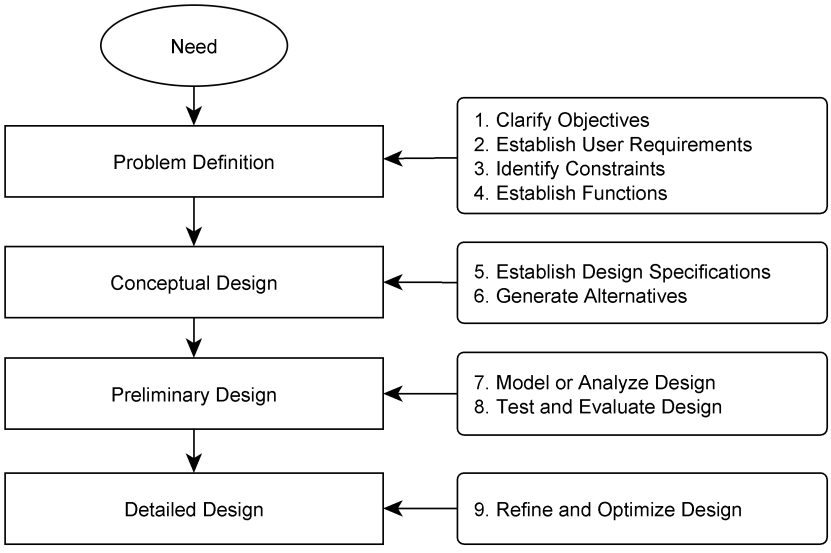


Figure 2.1: Five stage model for design (Dym & Little, 2008)

Figure 2.2 shows the research process for this thesis. In the first phase a top-down research will be done. This top-down approach will be used to identify relevant stakeholders and their requirements, the processes at liquid bulk terminals, the ecosystem in which the system is supposed to operate, and relevant literature in the field of scheduling and MASs. The findings from this first phase can be found in Part I of this thesis. The stakeholder analysis will be done using formal stakeholder analysis methods (M. Reed et al., 2009) and is based on expert knowledge within Systems Navigator, (semi-structured) interviews with involved parties, and literature research in the maritime sector. The operational processes will be studied from a scheduling research perspective, mostly based on research by Brucker (2007), Pinedo (2012), and Graham, Lawler, Lenstra, and Kan (1979). The systems environment will be studied based on earlier research by Diao (2014) and expert knowledge within Systems Navigator. Based on the identified operational processes several solution strategies for applying MASs in the liquid bulk domain will be explored based on literature reviews. Articles regarding the general solution methods are collected by searching through the TU Delft Library, Scopus and Web of Science using the keywords *MAS*, *Agent-based*, *Agents*, *Distributed Artificial Intelligence* (DAI), *Scheduling*, *Planning*, and *Job-shop Scheduling*. Extra attention was given to articles originating from journals such as *Artificial Intelligence*, *Engineering Applications of Artificial Intelligence*, *European Journal of Operational Research*, and *Journal of Scheduling*, as well as conference proceedings from the *Conference on MultiAgent Systems*, and *European Conference on Artificial Intelligence*. The works by Russel and Norvig (2009), Cramton, Shoham, and Steinberg (2006), Shoham and Leyton-Brown (2009), Wooldridge (2002), Weiss (1999) were used as a foundation upon which further detailed solution strategies were identified and researched.

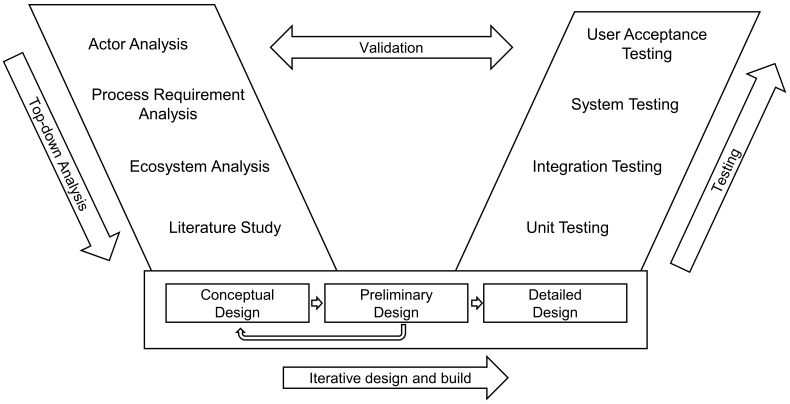


Figure 2.2: Research Methodology

The second phase consists of an iterative design and build process in which preliminary MAS designs are drafted, (partially) implemented and tested to verify whether they could be successfully implemented in the final design. The outcome of this phase is a final design consisting of all relevant components of the system, as well as a prototype implementation of this design. Designing the MAS will be done based on the Gaia methodology as introduced by Wooldridge, Jennings, and Kinny (2000). This methodology is developed specifically for designing MASs, both on the macro and micro level. Gaia is designed for MASs which have a global goal or quality measure, heterogeneous agents and a relatively small number of agent types (< 100). Gaia does not support dynamic organisation structures in which the inter-agent relationships change over time, or dynamic agent abilities, in which the abilities of agents change over time, but this is not expected to be problematic. The prototype will be implemented in the multi-paradigm programming language F#. This language was chosen because of its native support for actor programming, conciseness, and compatibility with the .NET framework and its extensive ecosystem. The findings from this phase can be found in Part II of this thesis.

The third and final phase consists of verifying the implementation, validating the design of the system with regards to the requirements as found in the first phase, and experimenting with the different proposed designs. For verification and validation this thesis will borrow heavily from relevant literature in the simulation field. This phase is documented in Part III of this thesis.

Part I

Background

Liquid Bulk Terminals 3

A liquid bulk terminal (sometimes called an oil terminal, storage terminal, or oil depot) is a facility for the storage of oil and/or petrochemical products, and is an important part of the oil and gas supply chain. In most cases, the terminal assumes custodianship but not ownership of the product. Liquid bulk terminals are often relatively simple facilities in which no processing of product is done, although in some cases additives are blended in with existing product. They mainly function as storage plants for intermediate or finished product and are therefore often situated close to refineries, or act as a transshipment point in which finished product is brought in on vessels, and distributed further to end customers such as petrol stations using trains, barges and trucks (Ligteringen & Velsink, 2012). Figure 3.1 shows a graphical overview of this petroleum supply chain and the role of liquid bulk terminals.

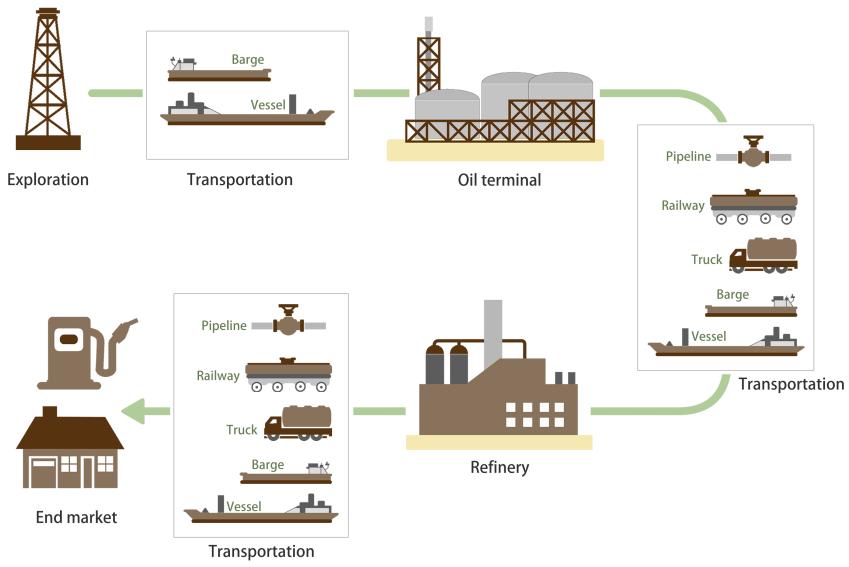


Figure 3.1: Petroleum supply chain (Diao, 2014, p. 9)

There are four main types of liquid bulk terminals (Verheul, 2010):

- **Strategic Terminals,** for strategic storage by organisations or governments;
- **Industrial terminals,** often close to refineries or chemical terminals to store raw materials and finished goods;
- **Import or export terminals,** mainly for making and breaking cargo;
- **Hub terminals,** as transshipment points.

Other common types of (maritime) terminals include container terminals and dry bulk terminals, and although there are some similarities there are also a number of important differences that need to be considered when designing a scheduling solution.

A container terminal is a transshipment point for cargo containers. In such a terminal the containers are loaded from and to vessels using gantry cranes. From the crane the containers are transported by truck and stacked for storage in designated areas. The main difference between container- and liquid bulk terminals, with regards to scheduling, is that container terminal operations are best characterized as discrete operations, in which each container can be scheduled separately, while liquid bulk terminals consist mostly of continuous processes. Due to the continuous processes at liquid bulk terminals, most operations require all infrastructure elements simultaneously, transforming the scheduling problem from a regular job-shop problem in container terminals to a job-shop problem with multi-processor tasks.

A dry bulk terminal is similar to a liquid bulk terminal in function but instead of storing product in tanks, the product is usually stored in large open fields. The product is loaded to and from the vessel using cranes with shovels and is transported to and from the fields either using conveyor systems or trucks. When the terminal mostly uses trucks, the scheduling process is once again a more discrete type of process which yields the same conclusions as with container terminals. However, if conveyor systems are used the processes are very similar to those at liquid bulk terminals making most of the conclusions from this thesis also valid for dry bulk terminals.

The remainder of this chapter will provide a comprehensive overview of the operations in and around liquid bulk terminals as well as the stakeholders involved in the various processes. The goal of this chapter is to provide deeper understanding of the functioning of liquid bulk terminals in order to support the decisions regarding the scheduling problem throughout the rest of this thesis. The first section of this chapter will examine the stakeholders involved in the operation of liquid bulk terminals as well as their main goals, interests and dependencies. The second section describes a set of typical jobs and tasks which are commonly found at liquid bulk terminals. It then goes on to describe the various infrastructure elements that make up these storage terminals. Finally, a full and formal definition of the scheduling problem will be given which is used as a basis for the rest of this thesis.

3.1 Stakeholders

Scheduling in liquid bulk terminals involves a number of different parties. The stakeholder analysis presented in this section is done using a three step approach by M. Reed et al. (2009). These three steps consist of identifying stakeholders, categorizing stakeholders and investigating the relations between the stakeholders. This thesis will use the following definition of a stakeholder by Freeman (1984) "Any group or individual who can affect or is affected by the achievement of the organizations objectives"

Nine different parties were identified as being a part of the scheduling process in liquid bulk terminals. This identification was done based on a combination of earlier work by Diao (2014), work by Ligteringen and Velsink (2012), as well as verbal communication with employees at a liquid bulk terminal and experts from Systems Navigator.

These parties are visualized using a stakeholder map in Figure 3.2, which shows the formal relations and dependencies between the involved stakeholders. The rest of this section will go in depth on the roles of the parties and their objectives.

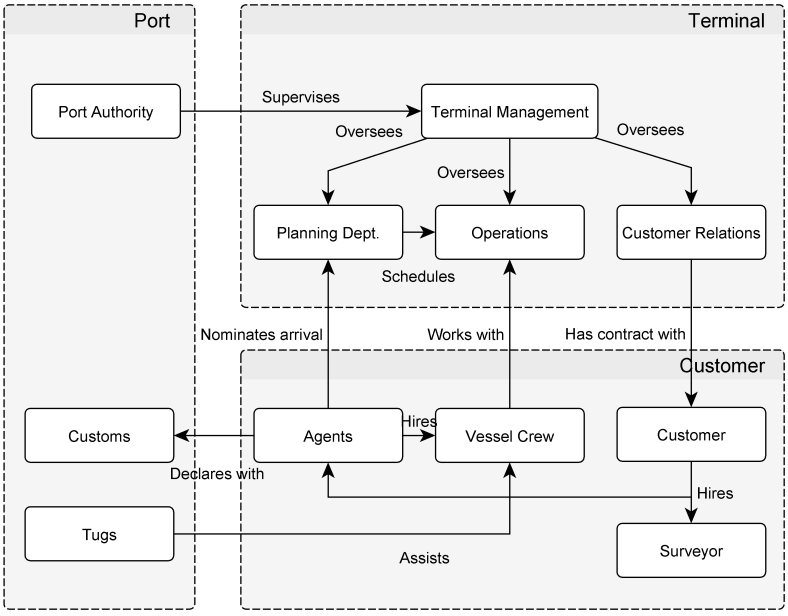


Figure 3.2: Stakeholder map of stakeholders involved in the Liquid Bulk Scheduling process

3.1.1 Terminal

The main stakeholder to be considered in scheduling liquid bulk is the terminal . The terminal offers a service to their customers which consists of receiving, storing, blending and dispatching liquid bulk. Customers rent storage capacity for fixed time periods and pay a predetermined amount for additional tasks such as receiving, blending and dispatching.

The main goal of the terminal is to maximize revenue. This can be achieved by keeping operational costs low, by increasing the utilization of the terminal, and by increasing the total storage capacity of the terminal. Increasing storage capacity can be done by expanding the number or size of available tanks at the terminal. However, this is costly both financially as well as in time and is out of scope for the remainder of this research. Keeping operational costs low can be achieved by keeping product changeovers low, as these tend to incur loss of product, and by keeping power costs low. Power costs in these terminals is not only calculated by multiplying the total power consumption with a price per kWh, but also by measuring the peak power usage. Therefore, it is beneficial for these terminals to spread out energy usage as much as possible.

Increasing terminal utilization can be achieved by optimally allocating routes on the terminal so that they only block few other possible routes, and fill in the idle gaps as best as possible.

A terminal consists of several departments. The relevant departments in this scheduling problem are Customer Relations, Terminal Management, Planning, and Operations.

Customer relations is responsible for the contact with customers. Their main interest is to improve the quality of service the terminal can deliver in order to make better agreements with potential clients and ensure continuity with existing clients.

Operations is responsible for the day to day operational tasks on the terminal. These include the berthing of vessels and the routing of product. Their main goal is to effectively use operations capacity in order to reduce peaks and thereby reduce the possibilities on errors.

The planning department is the department which allocates berths and routes to arriving vessels. They will be the end user of this planning solution. Their main task is to ensure that vessels are serviced in time so that no demurrage costs need to be paid.

Terminal management oversees the functioning of the entire terminal. It has several high-level objectives such as making sure that the terminal has a low impact on its surroundings regarding contamination and stench.

3.1.2 Customers

Apart from the terminal another main stakeholder are their customers. Typical customers of liquid bulk storage terminals are oil and gas companies such as Vitol and Shell. These companies use the services offered by storage terminals as a part of their supply chain in order to transport their product further to and from refineries and petrol stations using vessels, barges, road tankers, or pipelines.

The main goal of customers with regards to the scheduling at liquid bulk terminals is to have a low turnaround time when loading or discharging product. Lower turnaround times can be achieved by decreasing pumping time and / or waiting times. Pumping times can be decreased by using higher capacity infrastructure or utilizing more pumping capacity. Waiting times can be decreased by ensuring there are suitable routes available when the vessels arrive.

Another main goal of the customers of liquid bulk terminals is to maintain a high product quality. Since in many terminals the infrastructure elements are shared between customers and products, there is a chance of cross contamination between products, thereby reducing the quality. In the case of incompatible products being processed after each other in the same infrastructure, the infrastructure should be cleaned and prepared. This process is called infrastructure changeovers. Product quality is checked by the Surveyor.

Liquid bulk terminals typically serve several independent customers which can incentivize strategic behaviour from the customers side to achieve better allocations. There are several possibilities for customers to report job characteristics strategically. For example, reporting a lower cargo volume, or higher pumping capacity on the ship can result in a lower estimation of infrastructure utilization time. When the terminal uses this lower time to plan their routes, the vessel could receive a better allocation than they would get if they reported their characteristics truthfully (Figure 3.3). Although there is no structure in place to ensure truthful representation of pumping attributes by customers, most of these attributes are easily verified which lowers the incentive for strategic behaviour. It is not possible for the customer to strategically represent its arrival time as these are controlled by either *Automatic Identification System* (AIS) or the Port Authority.

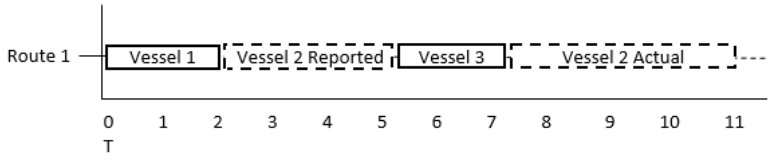


Figure 3.3: Strategic reporting of processing times

3.1.3 Surveyor

A Surveyor is part of an independent firm hired by the customer or terminal to check on their behalf whether the product quality is as expected. He typically inspects the product before loading or discharging is started. The surveyor is not a main stakeholder in the scheduling process, but it should be taken into account that these inspections take time.

3.1.4 Agents

The shipping agent is the organisation responsible for handling shipping and cargo, as well as representing the interests of their clients in their daily operations. Agents are hired by the terminals' customers for handling their shipments. Some of the general tasks of the agents include:

- Ensuring berth for the incoming ship;
- Arranging a pilot and tugs if necessary;
- Doing customs declarations;
- Contacting vessels and vessel crews.

The agents are the main point of contact for the planning department of liquid bulk terminals with regards to details concerning the vessel.

3.1.5 Vessel Crew

The vessel crew is responsible for all operations on board of vessels and barges. In some cases the vessels are owned by the customers themselves, in other cases the vessels are chartered by the customer (Huber, 2001). There are four main types of chartering agreements.

Voyage Charter
In a voyage charter the customer hires the vessel and crew for a specific voyage from an origin port to a destination port. The vessel is paid on a per-ton or lump-sum basis. Port costs, fuel costs and crew costs are paid by the owner of the vessel.

Contract of Affreightment
A contract of affreightment is similar to a voyage charter. In a contract of affreightment, the vessel is hired to transport a number of cargoes within a certain specified time period.

Time Charter
In a time charter the vessel is hired for a specific period of time. In this case, the charterer pays all the fuel costs, port costs as well as a daily hiring rate.

Bareboat Charter
A bareboat charter is a charter type in which the charterer hires only the boat, without crew or administration. The charterer assumes all responsibility for operating the boat. These charters are often for longer periods of time.

In liquid bulk operations the most common types of charters are time charters which means that the customer is incentivized to ensure short turnaround times at the terminal.

3.1.6 Port Authority

The port authority is responsible for the operation of the port as a whole. They govern, operate and develop the general port area. Their main task is to efficiently use the port area and ensure safe, fast and secure vessel handling in the port (Nijdam & Romochkina, 2012). The port authority is generally not directly involved in the scheduling process.

3.1.7 Customs

Customs is the agency responsible for collecting customs duties and controlling the flow of goods in and out of a country. When receiving or dispatching goods internationally, the customer or agent needs to ensure that the correct customs declarations are done before the operations are carried out. Usually, this does not pose as a bottleneck in the operations of liquid bulk terminals and can therefore be considered as out of scope for the remainder of this research.

3.2 Operations at Liquid Bulk Terminals

This section describes typical operations at liquid bulk terminals, as well as their defining characteristics. Four main sets of operations can be distinguished: tank-to-tank transfers, berth-to-berth transfers, tank-to-berth transfers, and berth-to-tank transfers. Tank-to-berth transfers are often referred to as vessel loading, while berth-to-tank transfers are often referred to as vessel discharging. Tank-to-tank transfers can also be referred to as internal transfers.

Apart from vessel loading and discharging tasks, there are also truck and rail loading tasks. However, these are less impactful to consider in scheduling liquid bulk as their operation times are shorter due to their smaller parcel sizes.

Oil vessels typically consist of a number of different compartments which can be loaded or discharged separately and can contain different products (Figure 3.4). A single vessel arrival at a terminal can therefore result into several operations, which can happen at multiple berths. The process of moving to different berths to process different parcels is called re-berthing. The order in which operations take place is mostly determined by attributes specific to the vessel and loaded product (Burns, 2015).

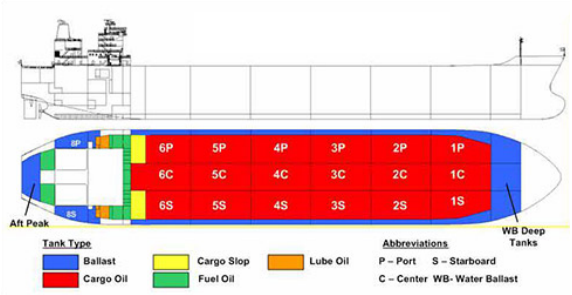


Figure 3.4: Diagram of Oil Tanker (Alaska Tanker Company, 2012)

All operations at liquid bulk terminals consist of several tasks. These tasks are cursory described in the following sections. A detailed description of the various steps is available in Appendix A.

Nomination
The first step in the process of a vessel arrival is the sending of a nomination form. This form represents a request for arrival and contains vessel and job specific information such as the vessels name, *Estimated Time of Arrival* (ETA), call sign, size, the product it is carrying and the quantity of product.

When a nomination is accepted by a terminal the vessel gets an allocated time slot and berth for its loading and/or discharging operations. When a nomination gets assigned to a route, it is commonly referred to as an order.

Arriving / Pilotage
The second step in vessel handling is the arrival or pilotage step. In this step, the vessel approaches the terminal and should be guided to a berth. After this step is complete, the vessel is securely attached to a berth and the final preparations for pumping can start. During this step, the target berth is claimed, as well as the waters surrounding the target berth.

Prepumping / Pre-Operations
Pre-pump processes take into account the time required for connection, line-up, analysis and ramp-up time. The prepumping step requires the berth as infrastructure as well as a team from the operations department.

Pumping
The pumping step includes all the steps in the actual transfer of product. During the pumping step, the entire route infrastructure as well as a part of the vapour return system is used.

Postpumping / Post-Operations
In the postpumping step the vessel is prepared for departure. During this time, a team from the operations department as well as the berth and its surrounding waters are used.

3.2.1 Operational Criteria

There are several criteria to which a job must conform to be processed in order to ensure rapid loading and discharging.

Waiting times
In most cases the terminal has contractual obligations to the customers not to exceed specific waiting times. Waiting times indicates the time period from the vessels' notice of readiness until it can proceed to berthing.

Idle Times at berth
Vessels and barges are allowed a maximum number of hours for idle times. These idle times include time for berthing, inspection, paperwork, and intermediate stoppages. These idle times vary mostly from 4 to 7 hours.

Product Viscosity, pressure and flow-rate
The combination of product viscosity, pump pressure and flow-rate together make up the pumping speed of product. The flow-rate determines the hours required to load or discharge a vessel/barge. The terminal and customers have contractual agreements so as to perpetuate the minimum flow-rates in correlation with minimum and maximum allowed pressures. These flow rates vary from 500 m^3/hour for small bunker-barges to $15,000$ m^3/hour for large vessels such as *Very Large Crude Carrier* (VLCC) or *Ultra Large Crude Carrier* (ULCC) supertankers. If the predetermined flow rate cannot be reached within the pressure boundaries, the product should be heated to decrease viscosity.

3.3 Liquid Bulk Products

The products that are handled at a typical oil terminal range from different types of crude oil, heavy fuel oils and a number of petroleum products. In many cases these products are divided into two categories: black and white product. Black oil is any type of oil that may be considered heavy or dirty. Petroleum products and distillates are considered 'clean' and light, and are classified as white product. These typically require infrastructure cleaning before they can be processed in case black product was loaded first. In many terminals, the land-side infrastructure is arranged such that black and white oil has separate pipe lines and loading arms.

Even though the distinction between black and white product covers most constraints that rise due to product types, there are still some exceptions that need to

be considered. For example in order to keep the quality of product high, low-sulfur products should not be processed after high-sulfur products without an infrastructure changeover (Leffler, 2008).

3.4 Terminal Infrastructure

Figure 3.5 shows a cross section of the infrastructure at a general liquid bulk terminal. Product arrives through vessels or other *Modes of Transport* (MOTs) such as trucks and trains. From the MOT, the product is loaded through a loading arm using pumps. The product then moves through a network of headers and lines in order to reach the tank of destination. A set of headers and pipes used for a transfer is often referred to as a route. The two main scheduling tasks in liquid bulk are the allocation of time slots and locations to the various MOTs and the allocation of routes.

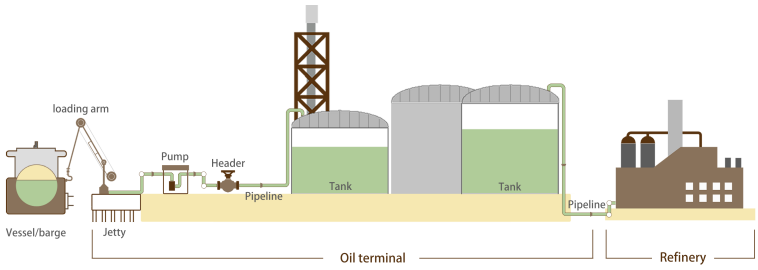


Figure 3.5: Cross section of liquid bulk terminal (Diao, 2014, p. 10)

After a job has completed, the used infrastructure is usually not cleared of product. This residual product remains in the lines, pumps and headers and should be considered when scheduling future routes. In order to facilitate different products, so called product changeovers are needed which are costly both in time as well as in lost product. Some terminals, in order to not deal with these changeovers, instead opt to use dedicated infrastructure elements per product group. Both options should be considered by the scheduling solution. A detailed description of the various infrastructure elements found at liquid bulk terminals is found in Appendix C.

3.5 System Environment

The scheduling system as described in this thesis will be a part of a larger scheduling support system which operates in liquid bulk terminals. Appendix D describes an exploration of the system environment consisting on the position of the scheduling support system in the terminal based on the ISA-95 standard, as well as the individual components making up the scheduling support system.

Because of the application in a decision support system, several extra requirements surfaced which are typically not found in regular scheduling algorithms. These are:

- **Process propagation effects:** Automatically adjust jobs when adjustments in its dependency chain are made;
- **(Internal) re-optimization:** Automatically propose scheduling options after user changes;
- **Freezing jobs:** Indicate whether certain jobs are not subject for future re-optimization and cannot be altered as a result of propagation effects.

Furthermore, the context at liquid bulk terminals offers several routes for further improvements such as retrieving data from AIS, Terminal Information Systems, or Harbour Information Systems and scheduling maintenance jobs. These further improvements are out of scope for the remainder of this thesis.

3.6 Formal Scheduling Definition

Given the stakeholders, infrastructure and operations as presented before, it is now possible to provide a formal definition of the scheduling problem in liquid bulk terminals.

The scheduling of resources in liquid bulk terminals is a specific version of the generic job shop problem which was first studied by Graham, Lawler, Lenstra, and Kan (1979). In these types of scheduling problems a set of jobs $J = \{j_1, j_2 \ldots, j_n\}$ needs to be scheduled on a set of machines $M = \{m_1, m_2 \ldots, m_n\}$. Every job j_i has a set of operations (i, j) on machines m_j. In order to formally define the scheduling problems at liquid bulk terminals the $\alpha|\beta|\gamma$-notation will be used as introduced by Graham, Lawler, et al. (1979). In this notation α indicates the characteristics of the machines which are required for processing the jobs. The β describes the job characteristics such as whether job splitting is allowed. Finally, γ indicates the optimization criterion of the scheduling problem. These three categories will be described for the case in liquid bulk scheduling in the upcoming sections.

3.6.1 Machine Characteristics

When examining the scheduling problems of liquid bulk terminals it becomes obvious that each task has to be processed by multiple processors at the same time, since the complete route needs to be available for the entire duration of the pumping. These types of tasks are often referred to as *Multiprocessor Tasks* (MPT) (Brucker, 2007).

There are three main types of MPT systems: a flow-shop MPT, an open-shop MPT, and a job-shop MPT problem. In a flow shop MPT problem, there are n jobs to process on m machines. All jobs need to be processed by all machines and all

jobs are processed by the machines in the same order. The order of the jobs on each machine can be different. In liquid bulk terminals the routes for each job can vary massively which causes the flow-shop problem to be an unsuitable representation of the scheduling situation.

The open-shop MPT problem once again consists of a set of n jobs and m machines. Each job has a (possibly zero length) processing time for each machine. Each machine can process only one job at a time and only one set of tasks can be processed simultaneously per job. The main difference with the flow-shop MPT system is that the order in which jobs are processed can vary freely. Due to this free ordering of tasks the open-shop MPT problem is not a suitable representation for scheduling in liquid bulk terminals.

Finally the job-shop MPT problem consists of a set of n jobs and m machines where each job consists of a chain of (strictly ordered) tasks which each needs to be processed on a subset of the m machines simultaneously. This type of problem closely resembles the situation in liquid bulk scheduling. Therefore, we set α= *Job-shop Problem with Multiprocessor Tasks* (JMPT). An extra m symbol is added after the MPT when the number of processors for each task is fixed. However, in liquid bulk scheduling this is not the case.

Apart from the various route infrastructure elements there are a number of processors needed for tasks which can be shared. For example, when gas is pushed through lines, this gas needs to be filtered using a vapour return system. Another shared re-source is the availability of electrical power. These aspects cannot be covered in the general job shop problem and are extensions on it.

3.6.2 Job Characteristics

Job characteristics are described by a set containing at most five elements: β_1 through β_6.

β_1 describes whether job splitting is allowed. If a job can be interrupted and resumed at a later stage, β_1 is set to $pmtn$. Else, it is not a part of the set. There are several moments in the operational processes at which jobs can be interrupted. Firstly, it is possible to interrupt jobs at the interfaces between tasks. For example, a vessel can wait a period of time after berthing is complete and before pumping is started. Furthermore, during the pumping, the pumping process can be interrupted and resumed at a later stage. It should be kept in mind though that when pumping is interrupted, the infrastructure will still be filled with product which may need to be pushed and cleaned. Therefore, β_1 is set to $pmtn$.

β_2 shows whether there are precedence relations between various jobs. This could mean that the loading of a vessel is dependent on another vessels arrival. There are several cases in which there are precedence relations between jobs. For example, when considering board-to-board transfers, both vessels will need to be berthed before the operations at either vessel can start. Furthermore, the loading of product on a vessel can depend on a certain blend operation to be finished beforehand. Therefore, β_2 is set to $prec$.

The third element of the set, β_3, indicates whether there are release dates specified for a job. Release dates can be considered as the earliest moment the job can be started. Since a job cannot be started before the vessel has arrived, $\beta_3 = r_i$.

β_4 is used to specify whether there are additional set-up times based on the order in which jobs are processed. When infrastructure such as a pipeline is used, it first needs to be cleaned of residue before another product can use that line. This can be indicated by setting $\beta_4 = s_{ij}$.

Since there are no hard deadlines, β_5 can be omitted. Finally, some scheduling problems require the batched processing of jobs. This means that certain jobs will need to be processed at the same time by a single machine. This is indicated by β_6. Since there is no requirement for batched processing in liquid bulk terminals, β_6 does not appear in the set.

3.6.3 Optimality criterion

As we have seen in the stakeholders section of this chapter, schedule optimality is, in the case of a liquid bulk terminal, often based upon multiple criteria. Important parameters are the total turnaround time of vessels, the total time at berth, and the energy consumption. Furthermore, the individual turnaround times are also important since there are often performance targets agreed with the individual customers. General job-shop problems do not consider multiple objective optimization, although several attempts are made in solving this problem such as the use of Pareto-based grouping algorithms (Gao et al., 2014; Geiger, 2006).

A Pareto optimal solution refers to a solution for which it is not possible to increase the value of an attribute while maintaining the value of all other attributes. A main disadvantage of only using Pareto optimal solutions is that the Pareto efficiency is a so called "minimal notion of efficiency" (Nicholas, 2012), meaning that it makes no statement about the overall desirability of a solution. For example, a solution which is much better on attribute A, but marginally worse on attribute B, is not a better solution from a Pareto point of view. Planners at a pilot terminal for Systems Navigator's scheduling support system specifically mentioned that vessels are often delayed a small amount of time to get a better overall solution (D. Merkestein, Personal communication, July 6^{th}, 2015). Due to this attribute of Pareto efficiency, it is not suitable for application in this case.

It has become clear that Pareto optimization does not offer the flexibility needed for application in liquid bulk scheduling. Traditionally, agent-based scheduling systems are better capable of weighing objectives than regular scheduling algorithms since the individual agents can safeguard their own interests based on their own optimality criteria (Agnetis et al., 2014). The application of *Multiagent Systems* (MASs) and the possibilities for different optimality criterion will be further discussed in the next chapter.

3.7 Scheduling Complexity

Not all computational problems are as easily solvable as others. Complexity theory offers a mathematical framework to study computational problems in order to classify them as being "easy" or "difficult". The most common classes in complexity theory are P, NP, NP-hard, and NP-Complete (Sipser, 2006). P includes problems which are solvable in polynomial time. The NP class includes problems where verifying the correctness of the solution is possible in polynomially bounded time. Within NP, the hardest problems are classified as NP-Complete. Finally, problems which are at least as hard as the hardest problems in NP, but not necessarily in NP are classified as NP-hard. For a full background on these classes the reader is directed to (Sipser, 2006).

Brucker and Krämer (1995) have proven that all $JMPT|n = k|C_{max}$ problems are strongly NP-hard[1], meaning it quickly becomes infeasible to search for optimal solutions as the problem size increases. When solving NP-hard problems there are three main constraints which can be released in order to find possible solutions (Kleinberg & Tardos, 2011):

1. Solve arbitrary instances of the problem;
2. Solve the problem in polynomial time;
3. Solve the problem to optimality.

These features will be covered in depth when discussing the solution strategy in Chapter 9.

3.8 Conclusion

This chapter has reviewed the three key aspects of the scheduling problem in liquid bulk terminals. First, a stakeholder analysis was performed to identify the main involved stakeholders and their requirements

From this stakeholder analysis it became clear that the two main parties involved in the scheduling problem at liquid bulk terminals are the terminal and its customers. For our purposes, the terminal can be considered as a single party with a set of shared objectives even though these might originate from different departments within the terminal. Furthermore, it should be taken into account that a terminal often serves a number of different customers, which are largely uninterested in the performance of other customers on that terminal. The surveyor, agents, vessel crew, port authority, and customs are considered out of scope. The goals from these two stakeholders can be summarized as follows:

[1] Strongly NP-hard is reserved for NP-hard problems for which there is no known polynomial reduction to a strongly NP-complete problem. A strongly NP-complete problem refers to a problem which remains NP-complete even when all numerical parameters are polynomially bounded (Garey & Johnson, 1978).

Terminal Goals:

1. High terminal utilization;
2. Low impact on surroundings;
3. Low peak power usage;
4. Low pressure on operations;
5. Not paying demurrage costs.

Customer goals

1. Short turnaround times;
2. Short dwell times;
3. Maintaining product quality.

Since the scheduling system described in this thesis is to be a part of a larger scheduling support system, several extra requirements surfaced. These are firstly that the system should be able to process propagation effects, meaning that the system should automatically adjust jobs in the dependency chain of a job if that job is changed. Furthermore, the system should be able to perform re-optimization after a user made changes to the schedule. Finally, the system should support marking jobs as "frozen", meaning that the re-optimization and propagation effects should not alter this specific job.

The second part of this chapter focused on identifying the operations and infrastructure at liquid bulk terminals in order to give a formal definition of the scheduling problem. Liquid bulk terminal scheduling closely resembles a specific version of the job-shop problem: $JMPT|pmtn, prec, r_i, s_{ij}|multi$. Since this problem is strongly NP-hard it is infeasible to find an optimal solution and heuristics will be required to find good solutions.

Having formally defined the scheduling problem in liquid bulk terminals, it becomes possible to assess the applicability of different MAS architectures and solution strategies on this problem. The results from this assessment are described in the upcoming chapter.

Multi Agent Systems for planning and scheduling 4

Having formally defined the scheduling problem in liquid bulk terminals, it is possible to move on to examine the applicability of *Multiagent Systems* (MASs) in this domain. Wooldridge (2002) describes, in his work, several main drivers for applying MASs for planning and scheduling which include cases when interfacing with legacy software, cases when the problem is inherently (geographically) distributed, and cases where the agents provide a natural metaphor for the system.

In case of liquid bulk terminals the main driver for applying MASs is that it provides a natural metaphor in which the relevant stakeholders can be represented by different agents which safeguard their own interests. This allows for more flexibility when considering optimization on multiple objectives (Agnetis et al., 2014).

Although MASs have been applied for planning and scheduling on numerous occasions, these applications are typically restricted to traditional Job-shop problems. As became clear in the previous chapter, the scheduling problem at liquid bulk is inherently a *Job-shop Problem with Multiprocessor Tasks* (JMPT) in which jobs require multiple processors simultaneously. Due to these differences the MASs described in the literature cannot be applied directly to this domain.

The purpose of this chapter is to review the literature surrounding MASs in order to determine what types of MASs will be suitable for scheduling JMPT problems as present in liquid bulk terminals. It starts by describing the architectural challenges found in designing MASs, both on the high level as on the individual agent designs. It then goes on to describe several methods in which MASs can be effectively applied for scheduling, as well as their applicability to the scheduling problem in liquid bulk scheduling.

4.1 Multi-Agent System Architectures

Defining MASs has posed to be a difficult problem and over time, and many mutually inconsistent definitions have been offered. Even the more simple question - What is a (single) agent? - has not yet been answered definitively (Shoham & Leyton-Brown, 2009). For this thesis, the following definition given by Wooldridge and Jennings (1995) will be used:

Definition 4.1.1 : Agent (Wooldridge & Jennings, 1995, p. 4)
"An agent *is a hardware or (more usually) software-based computer system that enjoys the following properties:*

- *Autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;*
- *Social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;*
- *Reactivity: agents perceive their environment and respond in a timely fashion to changes that occur in it;*
- *Pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative."*

Definition 4.1.2 : Multiagent System (Translated from (Ferber, 1995, p. 14) by (Madureira, Santos, Gomes, & Ferreira, 2007, p. 36))
"A Multi-Agent System is a system composed of a population of autonomous agents, which interact with each other to reach common objectives, while simultaneously each agent pursues individual objectives."

MASs can differ on three levels: the agents themselves, the interaction among the agents, and the environment in which they act (Fisher & Wooldridge, 1993). Important attributes on the agent level are the number of agents, whether they should be homogeneous or heterogeneous, whether their goals are contradicting or complementary, and how complex the agents are. On the interaction level the frequency of communication and the allowed communication paths are most important. The environment level focuses on the predictability of the system, and whether the agent environment is fixed or flexible (Weiss, 1999, p. 4).

When designing MASs we are faced with two main problems (Wooldridge, 2002):

1. How do we build agents that are able to successfully carry out the tasks we delegate to them?
2. How do we let agents interact with other agents in order to successfully carry out the tasks we delegate to them, especially in the case where agents cannot be assumed to share the same interests?

These two problems relate to two, overlapping, design questions. That of *agent design*, and of *agent organisation design*, a distinction which is often referred to as the *micro/macro* level.

The remainder of this chapter will explore the literature surrounding MASs for planning and scheduling in order to provide a clear solution direction for the rest of this thesis. First, several common micro- and macro level architectures will be described. It will go on by describing and classifying several applications of MASs for planning and scheduling and assess their applicability on the scheduling problem as defined in the previous chapter. Finally, a concluding paragraph will summarize the findings and propose a solution direction for the rest of this thesis.

4.1.1 Micro level architectures

The first main factor in the design of MASs is the architecture of the agents themselves. Russel and Norvig (2009, Chapter 2) define five different types of agents in their book which are, in order of increasing complexity: simple reflex agents, model-based reflex agents, goal-based agents, utility-based agents, and learning agents. The first two of these types are part of the group of reactive agents, which have at most a simple model of the world and a tight coupling between perception and action. The last three are so-called deliberative agents which have a much more detailed model of the world and make decisions based on logical reasoning. Although these five agent types are not exhaustive they fit most agent structures encountered in the literature and therefore provide a good basis for further research.

Simple reflex agents are the least complex form of agent. They have no state or memory and their actions are only defined by applying rules to the current state of the environment. This type of agents poses an interesting contradiction by the earlier definition from Jennings and Wooldridge in that these agents are not pro-active.

Model-based reflex agents have a form of state and a model of how they think the world works. Actions are triggered by applying rule-sets on the agents' model of the world.

Goal-based agents are an expansion on the model-based reflex agents in that they can think how actions will influence the world, based on their internal world model. When all possible actions are processed the action which achieves his goal will then be executed.

Utility-based agents are similar to goal-based agents but differ in the fact that their utility is not a binary function, but rather a continuous measure of utility. Utility-maximisation has been the de facto standard for choice models over the years. Recently however, the concept of regret minimisation has come up as an alternative for utility-maximisation (Chorus, 2010). This type of model can in several cases provide a better fit with human decision making than its utility maximising counterpart. Regret modelling will be further described in Chapter 5 but for a full explanation and tutorial the reader is directed to (Chorus, 2012c).

Typical implementations of utility-based agents include the Beliefs, Desires, Intentions (BDI) model which originates from folk psychology and is used originally

for describing a person's actions. The beliefs of the agent represent its environment, its desire is a change of the environment which is desired by the agent and intentions are the agents way of achieving these changes (Bratman, 1987).

Finally, learning agents have a feedback loop which alters their model of the environment based upon the results their actions have.

Most of the agent architectures for planning and scheduling encountered in the literature are either goal-based or utility-based agents (Kouider & Bouzouia, 2012; N. Liu, Abdelrahman, & Ramaswamy, 2005; Tripathi, 2014), although some examples of learning agents are also available (Aydin & Öztemel, 2000; Gabel & Riedmiller, 2008).

4.1.2 Macro level architecture

The macro level in MAS design refers to the question on how to let agents interact with each other in order to let them perform the tasks which are delegated to them. There is a large number of MAS organisations of which the most common are Hierarchical, Heterarchical, Blackboard, Dispatching and Pull (Baker, 1998). As with describing the agent architectures, once again this list is not exhaustive, but it provides a good fit with MASs encountered in the literature and is therefore considered as a good base for further analysis.

In a hierarchical architecture there are a number of master/slave relations between agents. The slave agents can only communicate with their master, who in turn communicates with his master until the highest level agent is reached. In the case of liquid bulk scheduling, we can imagine that a "route" agent is a master agent having multiple slave agents for each of the infrastructure components making up his route. Hierarchical MAS architectures are for example applied by Qin and Li (2012).

Heterarchical agents on the other hand have no distinct form of organization. All agents can communicate with each other and there is no concept of global information. Heterarchical architectures are mostly inspired by biological processes such as ant colonies which have an inherent capability of self-organisation and configuration (Duffie & Prabhu, 1994) and are applied by multiple authors for scheduling JMPT problems (Gabel & Riedmiller, 2008; N. Liu, Abdelrahman, & Ramaswamy, 2007; Zhou, Nee, & Lee, 2009).

An extension to the traditional Heterarchical architecture can be made through the application of agent organisations (Aldewereld et al., 2012). These organisations help in solving problems with the predictability of Agent Systems by clearly defining the communication structure and method between agents. The case by Aldewereld et al. is also a clear example of a Pull architecture where agents at the end of the chain signal earlier agents when for example inventory falls below a certain threshold.

Blackboard architectures are similar to Heterarchical architectures with the distinction that it allows the concept of global information through the use of a blackboard which agents can post information on. Examples of blackboard-based MAS are Boeing's shop-floor control system (Richter, 1993), as well as more theoretical work by Kouider and Bouzouia (2012).

Dispatching or scheduling architectures have a top-level broker agent which decides what job each agent will work on next. Dispatching architectures are more commonly found in MAS designed for control systems and are less suitable for application in liquid bulk scheduling.

4.2 MAS in liquid bulk scheduling

There are many different solution strategies available for planning using MASs, and selecting the right strategy is one of the main design choices in designing such a system. The research described in Chapter 3 has resulted in three main characteristics of the scheduling problem in liquid bulk terminals, with which a solution strategy should be able to deal with. The first of these characteristics is that the matching of vessels to routes vary over time since vessels arrive and depart within the scheduling horizon. The second characteristic is that routes cannot be considered as independent entities, as these consist of partially overlapping sets of infrastructure. The third main characteristic is that it is important to not only optimize on makespan, but terminal preferences regarding other attributes should also be represented in the solution.

Several solution strategies are proposed in the literature which can be divided into two main groups: cooperative approaches and competitive approaches. The rest of this chapter will describe several solution methods found in the literature regarding multiagent scheduling and will analyse their applicability on liquid bulk scheduling based on the criteria mentioned above.

4.2.1 Problem Decomposition

The first approach which will be described, is a cooperative approach in which the problem space is divided into subsets which require only local information for solving the scheduling problem. This type of solution strategy is for example employed by Kouider and Bouzouia (2012) and Wang and Liu (2006) who use problem decomposition for idle time minimization in a regular job-shop problem. Their method is based on representing the problem as a disjunctive graph[1], and finding cliques in this graph.

These cliques can then be represented by a heterarchical network of agents in which the agents only interact with their direct neighbours. The individual agents minimize their idle time by, for example, using a greedy heuristic. Kramer (1995) describes several modifications on the disjunctive graph model which make it possible for the model to represent multiprocessor task problems, making problem decomposition technically feasible for application in liquid bulk scheduling. However, in

[1] A disjunctive graph is one of the main tools in modelling a job-shop system. In these graphs the tasks that need to be performed are represented by vertices which can be connected by edges representing constraints between tasks. Edges are directed in case of direct order constraints between tasks, and undirected when there is no precedence order but both tasks require the same machine (Blażewicz, Pesch, & Sterna, 2000).

practice it is expected that the found cliques in JMPT problems will contain much more infrastructure elements as the dependencies between processors in JMPT problems are much greater than in regular job-shop problems.

A second disadvantage of problem decomposition is that, while it is referred to as an multiagent scheduling technique, it offers no benefits in the representation of stakeholder values in the scheduling problem. Agents in problem decomposition represent computational helpers for a divide-and-conquer scheduling strategy, rather than representing involved stakeholders.

In conclusion, problem decomposition seems like a promising solution strategy for scheduling systems as it allows for easily distributable and composable systems. However, it is currently unclear whether such a method can be effectively employed to JMPT as found in liquid bulk scheduling as the disjunctive graph representation of the problem will typically yield much larger cliques. Furthermore, the agents in problem decomposition represent computational helpers, rather than actual stakeholders in the scheduling process and therefore offer no benefits in the representation of stakeholder values in the scheduling process. For these reasons, problem decomposition is less useful in practice for application in liquid bulk scheduling.

4.2.2 Coalition Forming

A second cooperative approach is coalition forming (Shehory & Kraus, 1998). Coalitions are disjoint sets of agents which can together perform a task. Each agent that is added to a coalition increases the value of the coalition, but also its price. In coalition forming, the main goal is to find a coalition which can satisfy the task for the lowest coalition price. Several methods are available for forming coalitions such as Branch and Bound and Linear Programming algorithms (Bárta, Stepankova, & Pechoucek, 2002).

In liquid bulk scheduling a coalition forming approach could be implemented by representing each infrastructure element as an agent. In order to incorporate the time element each agent needs to represent his infrastructure at a certain timeslot. These agents can then keep increasing their coalition until a suitable route is found for handling a vessel arrival.

Bárta et al. (2002) identify clear benefits in coalition forming when applied in a dynamic agent environment where agents arrive and depart over time which at first sight makes it seem suitable for application in liquid bulk terminals. However, since the agents would represent the infrastructure (which is, apart from maintenance tasks, consistent) rather than the vessels (which actually arrive and depart over time) the usefulness of this property is diminished.

Coalition formation seems to be a promising approach when dealing with a largely connected compatibility graph between resources, since coalitions can be formed relatively freely in such an instance. However, in liquid bulk terminals, the compatibility graph is rather sparse since infrastructure elements are only connected to a limited number of other infrastructure elements which makes this approach less suitable. Furthermore, it is unclear whether coalition forming can properly deal with

terminal preferences regarding attributes which cover the entire schedule, rather than a single vessel arrival.

4.2.3 (Online) House Allocation Problem

The *Housing Allocation Problem* (HAP) is an example of a single-sided matching problem. Common applications of HAP include housing, school, and task allocation. A house allocation problem can be described as a tuple $(A, H, \succ)$ where A is a set of agents, H is a set of goods or houses with $|H| = |A|$, and each agent $a \in A$ has a strictly ordered preference over the houses. A feasible allocation is an assignment such that each agent receives exactly one house (Ergin, 2000). A common solution to the HAP is to use a serial dictatorship algorithm in which an order is specified over the agents, and in this order each agent receives his favourite house among the still available houses. This algorithm is easily proven to be strategy proof and provides a Pareto optimal solution since an agent can only improve his allocation by swapping with an agent which picked before him. However, this agent has already picked his most optimal house with a larger set of houses remaining which means he will be worse off when swapping.

Proof 4.2.1 : Strategy-proofness of Serial Dictatorship (after Ergin, 2000, p. 6)
Let A a be set of Agents
Let H a be set of Houses
Let each agent $a \in A$ have a strictly ordered preference over the houses
Let π be the priority order over the agents

Agent $\pi(1)$ will receive his favourite house if he is truthful, since all houses are still available. Therefore, he has no incentive for strategic behaviour.

Agent $\pi(n)$ will receive his favourite houses among the remaining houses if he is truthful, Therefore, he has no incentive for strategic behaviour.

The traditional HAP is a single-time matching problem in which the agents and houses do not vary over time which makes it unsuitable for application in liquid bulk scheduling. However, several extensions such as *Online Housing Allocation Problem* (OHAP) are available, in which agents can arrive and depart over time. The algorithm presented in (Jalilzadeh, Planken, & De Weerdt, 2010) gives a strategy-proof and Pareto optimal solution for the OHAP which works as follows: When an agent arrives, the best possible house is allocated to him based on his preference order. When an agent departs, check which reallocations can be performed to other agents. These allocations are done in order of the arrival time of the agents. Reallocations can be simple swaps but also chains of swaps are supported.

The second main characteristic of liquid bulk scheduling is that the houses (representing routes) are not singular entities, but rather sets of entities (infrastructure elements) which can be part of multiple groups. In practice this can be solved by adding all complete route options as houses and whenever a route gets allocated,

removing all routes which share an infrastructure element with it from the available route list.

A problem in the application of OHAP for liquid bulk scheduling is that it is a single-sided matching problem. This means that the agents (representing vessels) have a preference ordering over their allocations, but the terminal is unable to have any preference over what allocations are made. As already mentioned in Chapter 3, the terminal is an important stakeholder in the liquid bulk scheduling process and has clear preferences in their allocations. In practice it is unlikely that a scheduling system which is unable to represent the terminals allocation preferences will be accepted as a good solution.

A second disadvantage of applying OHAP is that it assumes that the agents are completely independent of each other in their allocation. However as we have seen, there are several cases such as for example board-to-board transfers which require both agents to be berthed simultaneously. Literature regarding OHAP has not provided us with possible solutions for this problem.

A final problem with applying OHAP in liquid bulk scheduling is that it only provides us with a Pareto optimal solution. As we have already seen in Section 3.6.3, the use of Pareto optima has several major disadvantages in that it is only a minimal notion of efficiency, which means it gives no indication of the overall desirability of the solution.

OHAP is an obvious example of an heterarchical agent architecture, since all agents are equal and there is no form of organisation or shared state between the agents. The agents are examples of the simple reflex agent described earlier.

It has become clear that while the OHAP provides some desirable properties and can theoretically be applied to liquid bulk scheduling, it still has several disadvantages. The first and most important disadvantage is that it is unable to deal with terminal preferences which will most likely not result in an acceptable schedule. Furthermore, since it only offers Pareto efficient results, there is no overall optimization on desirability of the solution.

4.2.4 Stable Matching Problem

The *Stable Matching Problem (or Stable Marriage Problem)* (SMP) is a two-sided matching problem which can be considered as a bipartite graph $G = (U, V, E)$ where U and V are sets of vertices, $|U| = |V|$, and E is a set of edges. Each element $u \in U$ has a preference order for the elements $v \in V$ and vice versa. The set E indicates possible matchings between U and V. An allocation A in G is a subset of E so that each vertex is only allocated once (Gusfield & Irving, 1989). The SMP was first introduced by Gale and Shapley (1962) and has been studied extensively due to its many real world applications. For application in liquid bulk scheduling this problem can be seen as U representing the available routes at a terminal, and V representing the vessels which are to be scheduled.

The typical method for solving the SMP is using the Gale-Shapley algorithm which guarantees that every agent gets matched and all matches are stable. The

runtime of this algorithm is $O(n^2)$ where $n = |U| = |V|$. The algorithm works in a number of iterations in which each unmatched agent in the first set proposes to their preferred agent in the second set. The agents from the second set reply "maybe" to their most preferred agent, and "no" to all others. These agents are then matched with reservation. Due to the provisional nature of the matchings, the agents in the second set are allowed to "trade up" if they are to receive an offer from an agent which matches better. This process repeats itself until all agents are matched (Gale & Shapley, 1962). The Gale-Shapley algorithm exists in a man-optimal and a woman-optimal form. Consider for example a problem instance containing the preference orders as shown in Table 4.1. This problem instance can produce 10 different stable matches, as shown in Figure 4.1.

Table 4.1: An instance of the SMP (After Gusfield & Irving, 1989, p. 22, 23)

Men	Preferences			
1	1	2	3	4
2	2	1	4	3
3	3	4	1	2
4	4	3	2	1

Women	Preferences			
1	4	3	2	1
2	3	4	1	2
3	2	1	4	3
4	1	2	3	4

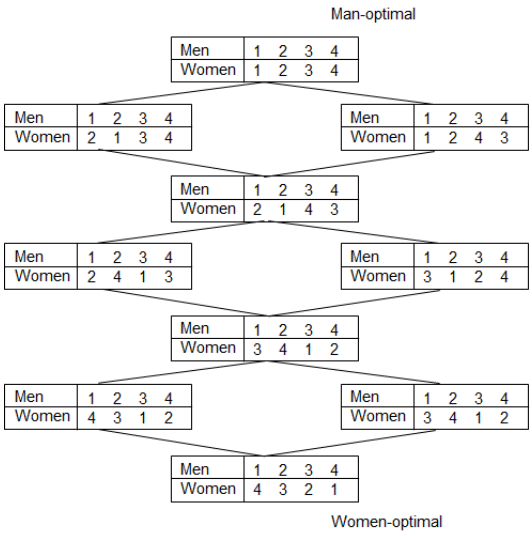


Figure 4.1: Lattice structure of stable matchings (After Gusfield & Irving, 1989, p. 22, 23)

The SMP is a clear example of a MAS with an heterarchical architecture, as there is no distinct form of organisation amongst the agents except for the division into two equal groups, and no global information is needed for the SMP to function. The agents are examples of the simple reflex agent described earlier.

The first obvious extension which is needed in order to apply SMP in liquid bulk scheduling is allowing uneven set sizes for U and V, as the number of routes is unequal to the number of to be scheduled vessels. This is a simple extension of the Gale-Shapley algorithm described above and can be implemented easily.

In the regular SMP each member of U can potentially be matched with each member of V. However, in liquid bulk scheduling there are obvious cases where this is not possible due to infrastructure restrictions like maximum allowed lengths at berths or incompatible products. Furthermore, a large portion of the routes will not be connected to the desired tank. A common extension on the SMP to deal with this is SMPI (Stable Matching Problem with Incomplete lists). This type of problem can be solved with minor adjustments to the algorithms used for the standard SMP.

A second problem arises when considering the time dimension in liquid bulk scheduling and the arrival and departure of vessel agents. Due to the arrival and departure of the agents over time the routes should not be allocated indefinitely, as this causes inefficient or infeasible schedules. This problem can be dealt with by discretizing time into timeslots of for example an hour and transforming the route agents into (route, timeslot) agents. Vessel agents can then be matched with all (route, timeslot) agents in the duration of their operations. Extra complexity is added due to the fact that the operation time of a vessel will largely depend on the used route infrastructure.

As with the HAP described in the previous paragraph, a problem surfaces due to the routes sharing infrastructure elements. This could theoretically be solved by matching each vessel to all (route, timeslot) agents which share an infrastructure element with the preferred route during the vessels operation duration.

While this extension seems similar to the *Hospitals / Residents* (H/R) problem in which each hospital can be matched with multiple residents, there are several main distinctions. First, in the H/R extension there is always a fixed quota on the number of residents a hospital can be matched with. Therefore, the H/R can be reduced into a standard SMP by replacing each hospital with quota q, by q copies of that hospital (Gale & Sotomayor, 1985). However, since the duration of operations for each vessel is not known beforehand, we cannot simply replace each vessel by a number of copies equal to its duration. More so, this problem is amplified with the sharing of infrastructure elements by the routes, since the number of overlaps for each route is impossible to determine beforehand. Systematic review of the literature surrounding stable matching has not led to a suitable solution for this problem.

A second problem arises when considering some of the terminals' preferences as these are not only preferences on single allocations, but rather attributes of the schedule as a whole. For example, the idle time at berth optimization does not only consider a single vessel allocation, but rather the vessel allocation as well as the

other vessels allocated at that berth. Stable matching does not provide a method for implementing such cross-agent preferences.

A third problem is similar to that discussed in the HAP section previously. Since the SMP provides us with a Pareto optimal solution, it gives no indication of the overall desirability of the solution.

Finally, as with the HAP, the SMP assumes independence between the allocations of vessels. However, with berth-to-berth transfers, both vessels need to be berthed in order for operations to be started.

In conclusion there are several interesting advantages in applying (extensions of) stable matching, but in practice it seems infeasible due to problems with agents arriving and departing over time, and with routes consisting of (partially) overlapping sets of infrastructure elements. Furthermore, the SMP offers no clear way to incorporate terminal preferences on attributes which span the entire planning, rather than only a specific interaction with a single vessel.

4.2.5 Combinatorial Auctions

The final solution strategy which will be discussed are *Combinatorial Auctions* (CAs), an instance of the group of competitive strategies. Combinatorial auctions are based on regular auction theory but with the distinction that instead of bidding on single items, bids are placed on sets of items. This has clear advantages when items are complementary in their value, such as in for example radio spectrum auctions, procurement, and airspace system resources (Cramton et al., 2006, Chapter 20 - 23). There are many examples of combinatorial auction systems such as the Vickrey auction and the ascending proxy auction which vary in simplicity, individual rationality, and truth-dominance (Wooldridge, 2002). Combinatorial auctions are successfully applied in a number of scheduling problems (Amir, Sharon, & Stern, 2015; Collins, Gini, & Mobasher, 2002; de Vries & Vohra, 2003; Hunsberger & Grosz, 2000; N. Liu et al., 2005)

A combinatorial auction can provide a good analogy for the problem in liquid bulk scheduling in the following way. There are two main types of agents: vessel agents, and a terminal/planner agent. Vessel agents are interested in receiving route allocations in certain timeslots. The vessel agents can evaluate potential route options and price these routes. These bids are then submitted to the terminal/planner agent which can use these bids to find a good allocation without overlapping infrastructure elements. In traditional CAs, the main goal of the winner determination is to find an allocation which maximizes revenue. However, the problem is easily extended to allow for multi-attribute weighing by the planner agent in determining the desirability of an allocation. By applying CAs the three main characteristics of the scheduling problem as introduced in Chapter 3 are covered. It is possible to add the time dimension of the problem by letting the vessels bid on (route, timeslot) tuples. Furthermore, the terminals' preferences can be incorporated since there is an agent representing the terminal making the final allocation. Finally, routes consisting

of overlapping infrastructure elements are naturally handled by bidding on sets of infrastructure, rather than single items.

Overall it seems that CAs can function as a good solution strategy for the scheduling problem in liquid bulk terminals. The following chapter will go in depth on the design of a CA mechanism for this problem.

4.3 Conclusion

Three main properties were identified as characteristics which mostly influenced the applicability of different solution strategies in the liquid bulk scheduling domain. The first of these characteristics is that the matching of vessels to routes vary over time since vessels arrive and depart within the scheduling horizon. The second characteristic is that routes cannot be considered as independent entities, as these consist of partially overlapping sets of infrastructure. The third main characteristic is that it is important to not only optimize on makespan, but also terminal preferences regarding other attributes should be represented in the solution.

This chapter evaluated several solution strategies such as coalition forming, stable matching, house allocation and combinatorial auctions based on these criteria. The results from this evaluation are summarized in Table 4.2. This table clearly shows that most of the evaluated solution strategies are unable to deal with one or more of the characteristics of liquid bulk scheduling. However, the application of *Combinatorial Auctions* (CAs) seems promising and will therefore be further expanded upon in the following chapter.

Table 4.2: Evaluation of solution strategies in liquid bulk scheduling

Strategy	Applicable	Time Dimension	Shared Infra.	Terminal Pref.
Problem Decomposition	✗	N.A.	N.A.	N.A.
Coalition Forming	✓	?	✓	✗
House Allocation Problem	✓	✓	✓	✗
Stable Matching Problem	✓	✗	✗	Partial
Combinatorial Auctions	✓	✓	✓	✓

Combinatorial Auctions 5

Auctions play a key role in the allocation of resources among self-interested agents in *Multiagent Systems* (MASs). As already discussed in the previous chapter, it is very common to solve agent agreement problems as a type of auction, which turn out to be a general solution for the allocation of resources among self-interested agents (Shoham & Leyton-Brown, 2009).

There are a number of types of auctions, including single-good, multi-unit, and *Combinatorial Auctions* (CAs). Single-good auctions are the most common and relatable type of auction. In this auction type, there is one copy of a single good available, and all bidders are interested in obtaining that good. Multi-unit auctions still have only one *kind* of good available, but there are now multiple copies of that good. Bidders can want either one or multiple of this type of good. In CAs there is a set of heterogeneous goods available to the market (de Vries & Vohra, 2003). Each participant in the auction is typically interested in acquiring a set of these goods, as the combination of goods is typically worth more than their individual components. CAs have been applied for multi-agent route allocations problems in previous studies (Collins et al., 2002; Amir et al., 2015; Hunsberger & Grosz, 2000) and seems to provide a good analogy to the route allocation problem as found in liquid bulk scheduling, as the infrastructure elements can be represented by goods, and the agents are interested in acquiring a bundle of infrastructure elements which make up a route.

This chapter will describe in detail the application of CAs in liquid bulk scheduling. In order to achieve this, this chapter is split into several parts. First, a general introduction to CAs will be given in which CAs will be defined and possible design dimensions in designing CA mechanisms will be specified. Sections 2 to 4 will then go in depth on these design dimensions for the specific application of CAs in liquid bulk scheduling. Finally, a concluding summary on the design decisions will be given.

5.1 Preliminaries

In the reviewed literature, no single exhaustive definition of auctions was given. For the purpose of this thesis, the definition as given in 5.1.1 will be used which is based on work by Shoham and Leyton-Brown (2009), Cramton et al. (2006), Sandholm (2002b).

Definition 5.1.1 : Auction
A mechanism for interaction between an auctioneer and one or more bidders – which allows the bidders to indicate their interests in one or more resources, and allows the auctioneer to determine an allocation of these resources and a set of payments by the bidders – of which the outcome is a deal between the auctioneer and one or more bidders.

In this thesis, the following notation will be used for discussing CAs. This notation is based on Lehmann, Muller, and Sandholm (2006). A CA is characterised by a set of bidders $N = \{1, ..., n\}$, and a set of items $M = \{1, ..., m\}$. A bundle S is a tuple of a set of items $I \subseteq M$ and a time period $t_{start}...t_{finish}$. A bid by bidder i on bundle S is denoted by $v_i(S)$. An allocation is described by $x_i(S) \in \{0, 1\}$, where $x_i(S) = 1$ when bundle S is allocated to bidder i. Finally, an allocation is feasible if every item is allocated not more than once for each timeslot and at most one subset is allocated to each bidder.

$$\sum_{i \in N} \sum_{S \subseteq M, S \ni j} x_i(S) \in \{0, 1\} \text{ for all } j \in M \tag{5.1}$$

$$\sum_{S \subseteq M} x_i(S) \in \{0, 1\} \text{ for all } i \in N \tag{5.2}$$

When exploring CA mechanisms, there are three main dimensions of auction design rules. These dimensions will be used throughout this chapter when discussing existing CA mechanisms as well as when discussing the differences with liquid bulk scheduling. These dimensions are:

1. **Bidding rules:** How are bids made;
2. **Clearing rules:** How is the final allocation determined based on the submitted bids;
3. **Information rules:** What information is available to who in which stages of the auction.

When designing CAs there are often several desirable attributes with regards to for example truthfulness, and overhead. Designing CAs to have these properties is commonly referred to as Mechanism Design.

Definition 5.1.2 : Mechanism Design (Wooldridge, 2002, p. 130)
"Mechanism design is the design of protocols for governing multiagent interactions, such that these protocols have certain desirable properties".

The rest of this chapter will go in depth on the application of combinatorial auctions for liquid bulk scheduling. It will do this by discussing the three main dimensions of auction design introduced above.

Several attributes of liquid bulk scheduling make the application of CAs different in this domain in comparison with traditional implementations. First of all, traditional auctions focus on allocating goods in exchange for some kind of payment. However, several issues arise with this concept which will be discussed in the upcoming section. The second main difference is that typical CAs do not allocate goods over time, which will be discussed in Section 5.3.

Finally, the auctioneer is not only interested in revenue maximization. Instead, other attributes such as number of vessels allocated and terminal utilization should also be taken into account. This difference will be discussed further in Section 5.4.

5.2 Truthfulness

A major topic in mechanism design is to design CAs which are strategy proof. This means that the mechanism should remove the incentive an agent might have to manipulate parameters in order to receive better allocations. Chapter 3 identified several parameters which can be influenced to receive better allocation such as reporting lower cargo volumes, reporting higher pumping capacity and reporting earlier ETA's.

Typically, mechanisms use payment structures to guarantee truthfulness. For example, in one of the most well known CA mechanisms, the *Vickrey–Clarke–Groves* (VCG) auction, the highest bidder wins the item but instead of paying his bid value, he pays the second highest bid on that item. However, Schummer and Vohra (2007, p. 233) note that "there are many important environments where money cannot be used as a medium of compensation". For example due to ethical or legal considerations in for example organ donations (Ashlagi, Fischer, Kash, & Procaccia, 2010).

Liquid bulk scheduling is also a domain in which payments cannot be easily implemented, as existing contracts between terminals and customers do not allow for extra payments. It is therefore interesting to ask whether it is possible to design truthful auction mechanisms without payments.

Fotakis, Krysta, and Ventre (2013) and Krysta and Ventre (2010) study the feasibility of designing strategy-proof CA mechanisms where payments are not possible. Removing payments from these assignment problems adds a number of difficulties, since the properties of monotonicity and weak monotonicity[1] (Archer & Tardos, 2001; Lavi et al., 2003) no longer suffice for guaranteeing truthfulness (Dughmi & Ghosh, 2010). Furthermore, traditional truthful auction mechanisms such as the VCG mechanism require payments to guarantee truthfulness. Unfortunately, none

[1] "[Under weak monotonicity], if player i caused the outcome of f to change from a to b by changing his valuation from v_i to u_i, then it must be that i's value for b has increased at least as i's value for a."(Lavi, Mu'alem, & Nisan, 2003, p. 13)

of the findings from the literature seemed transferable to the case as described in this thesis.

As a result of this, the auction as described in this thesis loses the economic and game-theoretic properties regarding its allocation but is still useful as an agent coordination system. This is illustrated by Koenig, Keskinocak, and Tovey (2010, p. 1) as follows: "Such issues do not arise in auction-based coordination systems because the agents simply execute their program. Thus, while some insights from economics can be exploited for building auction-based coordination systems, many of them do not apply. Some researchers therefore prefer to use the term "auction-inspired control algorithms" to highlight these differences".

Losing these properties is not considered a problem since it is relatively easy to verify strategic behaviour as described in Section 3.1. Due to these properties, there is no necessity for considering existing CA mechanisms, as these are typically designed around payments to ensure truthfulness.

5.3 Bidding rules in liquid bulk scheduling

This section concerns the problem of representing bids of the agents in CAs. Bidding languages are often not formally specified in regular auctions. However, due to the often large space of possible bids in combinatorial auctions, they may not be ignored (Nisan, 2006).

Bidding languages for combinatorial auctions were always implemented implicitly but were first studied explicitly in Nisan (2000). The author defined several characteristics of bidding languages which will be discussed in this section.

5.3.1 Bid value determination

In many cases, a bid of an agent on an bundle will be equal to the agents' valuation of that bundle. However, in several iterative auction mechanisms this is not the case. Instead, agents bid the asking price of the bundle which will be raised iteratively until agents are no longer willing to bid on that bundle (Parkes, 1999b, 1999a). Iterative auctions are typically applied when an initial elicitation of all bid valuations is infeasible due to the size. It is expected that the bid size of the agents in the case of liquid bulk scheduling will not pose issues which cannot be solved easily by carefully designing the bidding language. Therefore, iterative auction mechanisms will not be further considered for implementation. A detailed description on how the agents formulate their bids is given in Chapter 8.

5.3.2 Size limitations

It should be noted that in combinatorial auctions it is most of the time impossible to allow bids on all possible valuations. Since the size of the space of all valuations v on all possible subsets of m items is exponential in 2^m, it becomes apparent that not all valuations should be represented, but only interesting valuations.

These size limitations are further amplified by the fact that scheduling problems have a time-dimension, which further increases the complexity. The first problem which arises is that adding a continuous time dimension to the problem makes the solution space infinitely large. This can be addressed by treating time as a set of discrete time steps $T = \{0, 1, 2, ...\}$.

The second problem has to do with route preferences. If agents would only report their single route preferences at the start of the auction, the situation as shown in Figure 5.1 can arise. In this simple example there are two routes, and two vessels. Table 5.1a shows the bids the agents submitted where a lower value is better.

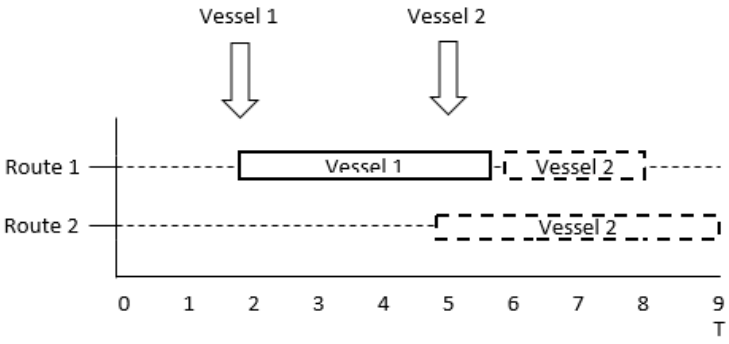


Figure 5.1: Example allocation

Considering these bids, the best possible allocation is to provide route 1 at $t = 2$ to vessel 1, and route 2 at $t = 5$ to vessel 2. However, as Figure 5.1 shows it is clearly better for vessel 2 to wait until vessel 1 is finished and then use route 1.

Two main solution strategies can be employed to deal with this issue. First, an iterative scheduling mechanism can be used which only allocates one route to one vessel during each iteration. After each iteration the vessels can then provide new bids based on the earlier made allocation. The main advantage of using this method is that it will always provide accurate results. However, due to increased overhead in communication and route recalculation this method is likely to be slow.

The second solution strategy is to let the agents not only bid on infrastructure at the time of their arrival, but to also let them bid on infrastructure several hours later. This has two advantages. First, the computational power needed to compute allocations is less than in the first case, since the finding of feasible routes only needs to be done once. Furthermore, this strategy also allows agents bidding on routes some time before their actual *Estimated Time of Arrival* (ETA). In this case, the vessel can be asked to sail faster in order for him to receive a better allocation. An example set of bids is shown in Table 5.1b

Table 5.1: Bid values on routes

(a) without slack

Agent	t	Route	v
1	2	1	3
	2	2	6
2	5	1	2
	5	2	4

(b) with slack

Agent	t	Route	v
1	2	1	3
	3	1	4
	4	1	4
	2	2	6
	3	2	7
	4	2	8
2	5	1	2
	6	1	3
	7	1	4
	5	2	4
	6	2	6
	7	2	7

5.3.3 Bidding languages

The leading paradigm for bidding in combinatorial auctions is to make sets of bids for bundles of items, where the bid can either be exclusive or non-exclusive. This paradigm is based on three basic types of bids which will be discussed in this section (Nisan, 2000).

1. **Atomic bids:** In an atomic biding language, the bidders submit a bid pair (S, p) where S is a bundle of items and p the price he is willing to pay for that bundle.
2. **OR bids:** OR bids are an extension on atomic bids where each bidder not submits a single bid pair, but a set of pairs. Agents are willing to receive any number of the bundles they submitted a bid on.
3. **XOR bids:** XOR bids are similar to OR bids but with the difference that in the case of XOR bids, the agents are only willing to receive a single bundle of items.
4. **AND or ALL bids:** AND or ALL bids indicate that the bidder wants to receive any of two or more distinct subsets. AND and ALL bids are somewhat redundant as they can also be expressed by making the cross product of the subsets, but can still be useful as they reduce the size needed for bid elicitation.

Using the three basic bidding types described above it is possible to make a number of powerful combinations to further enhance bidding expressibility such as XOR-of-ORs bids and OR-of-XORs bids. At first impression, it seems that agents

in liquid bulk scheduling can be considered to be single-minded, i.e. only interested in obtaining a single bundle. Therefore, it seems that the earlier described XOR bid is powerful enough. However, when introducing multiple parcels per vessel it becomes clear that a single agent will need multiple routes, and is therefore not single-minded. Instead, the agent wants one of possible allocations for each of his parcels. This corresponds with a AND-of-XOR or ALL-of-XOR bidding language (Zinkevich, Blum, & Sandholm, 2003). For further details on combinations of these basic types the reader is directed to Nisan, Roughgarden, Tardos, and Vazirani (2010).

5.4 Clearing rules in liquid bulk scheduling

Definition 5.1.1 shows that traditionally, auctions are focused on finding an optimal allocation of goods with regards to cost or revenue. However, in many scenarios, other attributes than only cost play a role in assessing the quality of allocation (Boutilier, Sandholm, & Shields, 2004). For example in procurement, the auctioneer can be interested in the diversity of suppliers awarded business, or the size of the suppliers. The same holds for scheduling in liquid bulk terminals, as the terminal is not only interested in the turn-around time of the vessels, but also the number of other possible routes are blocked, and the utilization of the terminals' infrastructure. This section will describe and evaluate several methods which can be used in practice for multi-objective combinatorial auctions.

5.4.1 Hard constraints

In practice, multi-objective auctions are often dealt with by imposing hard constraints on the non-price attributes, such as setting a maximum number of suppliers in a procurement auction (Boutilier, Patrascu, Poupart, & Schuurmans, 2006). However, this approach is often considered undesirable, as a bid-taker is usually willing to deviate slightly from the set boundaries on one attribute, if that gives large benefits on another attribute.

Another option for dealing with these multi-attribute bid-taker preferences is to accept percentage bids (Loertscher & Marx, 2014). This type of bidding scheme is typically implemented by letting the preferred bidder win as long as his bid is within a specified percentage of the normal winner. However, this is still a form of setting hard-constraints which suffers from the same problems as posed earlier.

5.4.2 Pareto Optima

Another common approach for multi-objective optimization is the use of Pareto optima as described in Section 3.6.3. However, as already seen in that section, the use of Pareto optima has several major disadvantages such as that it is only a minimal notion of efficiency, which means it gives no notion of the overall desirability of the solution.

5.4.3 Discrete Choice Modelling

When considering multi-attribute bid-taker preferences in auctions, we can further generalize these auction mechanisms by reducing the problem to a *Discrete Choice Model* (DCM). These choice models attempt to describe, explain, and predict choices between two or more discrete alternatives, such as transport mode choices (Ben-Akiva & Steven, 1985). A DCM specifies the probability that, given a set of observed variables, an unobserved error term, and a set of alternatives, a particular alternative is chosen. The intended purpose of applying DCM in this thesis is to accurately represent the planners preferences with regards to proposed schedule options based on a set of observable variables, rather than predicting the average choices a large group of planners would make. Therefore, the DCM discussed in the remainder of this thesis will be implemented as deterministic methods for ranking alternatives, without considering the random unobserved error.

A common used method for capturing trade-offs between price and non-price attributes is applying a utility function of the form:

$$U_x = \sum_{i=1}^{k} w_i f_i(x) - c(x) \tag{5.3}$$

In this funxtion, X represents the set of feasible allocations $\{x_1, ..., x_n\}$, $U(x)$ is the utility of an allocation, F the set of *features* or attributes, $\boldsymbol{w} =< w_1, .., w_k >$ a weight vector, $\{f_1, ..., f_k\}$ $w_i f_i$ the weighted utility per attribute and $c(x)$ the cost of an allocation.

In contrast with the common *(Random) Utility Maximization* ((R)UM) model, literature has emerged introducing its counterpart: *(Random) Regret Minimization* ((R)RM) models (Chorus, Arentze, & Timmermans, 2008). The main distinguishing feature of this model is that while the linear utility maximization model assumes that comparisons between alternatives are made only at the level of the alternatives' total utility, the regret minimization model assumes that comparisons between alternatives are made at the level of the individual attribute; i.e. a utility maximizer focuses on the performance of an alternative in isolation, where a regret minimizer compares alternatives with every other alternative in all aspects (Chorus, 2012c) (Figure 5.3). Consider, for example, a case as proposed in (Chorus, 2012a, p. 80) with three choice options A, B, and C where A is a low quality alternative and B is a high quality alternative. Using a (R)UM model, the probability for choosing A over B will be a fixed ratio independent of the quality of C. However, using a (R)RM model, the low quality alternative A will become less interesting as the quality of alternative C increases (Figure 5.2). The regret of an alternative can be calculated using a function of the form:

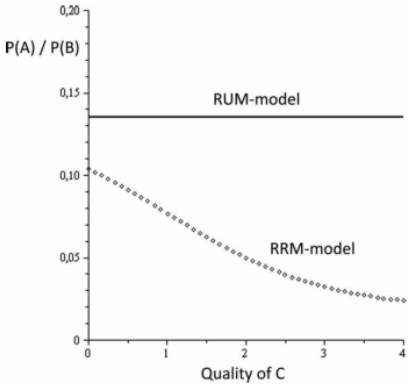


FIGURE 5.2: Probability of choosing alternative A over B using (R)UM and (R)RM (Chorus, 2012a, p. 80)

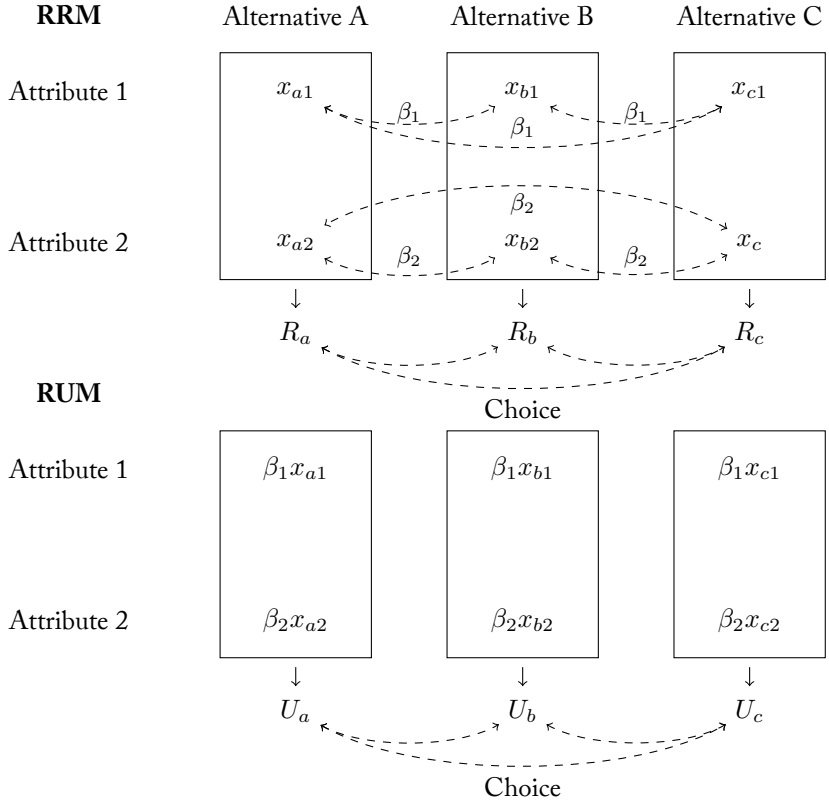


FIGURE 5.3: Regret Minimization (top) vs. Utility maximization (bottom) (after Chorus, 2012c, p. 18, 20)

$$R_i = \sum_{j \in A | j \neq i} \sum_{m \in M} \ln(1 + e^{(\beta_m (x_{jm} - x_{im}))}) \tag{5.4}$$

Where R_i is the regret of the alternative which is observed, A the set of alternatives, M the set of (relevant) alternative attributes and β_m the weight for attribute $m \in M$. For a full explanation of the random regret minimization model the reader is directed to (Chorus, 2012c).

Many empirical studies (Chorus, 2012b; Chorus, Annema, Mouter, & van Wee, 2011; Chorus & de Jong, 2011) have shown that in certain cases, the (R)RM provides a significantly better fit on either the stated or revealed preference in the sample data, meaning that they are better able to capture the trade-offs made by humans when comparing alternatives. Since the goal of this scheduling algorithm is to assist the planner in making allocation choices, it is important that the solution acceptance of the algorithm is high. Therefore, it is beneficial to let the ranking of alternatives be similar to the planners preference ranking.

One major drawback with applying (R)RM for determining the winner in combinatorial auctions is that the possible solution space is massive, which makes comparing all alternatives among each other on each attribute infeasible (Guevara, Chorus, & Ben-Akiva, 2013). Chapter 9 proposes a hybrid solution between (R)UM and (R)RM which first reduces the possible solution space using (R)UM and then does the final ranking using (R)RM.

So far, literature regarding (R)RM has not been able to clearly identify for which types of problems the (R)RM is expected to perform better than the (R)UM. Therefore, both versions will be implemented and tested.

There are still a number of issues with using fixed weight utility functions for eliciting bid-taker preferences. Unsurprisingly, bid-takers are usually uncertain about their feature weights, or are unwilling to communicate these (Keeney, Raiffa, & Rajala, 1979). Also, we have seen in Chapter 3 that the allocation preferences of a terminal are based on multiple stakeholders within the organisation which can further hamper constructing precise trade-off weights. This issue can be partly overcome by introducing a feedback loop in the system which adjusts future trade-off weights based on the actual choices made by the planner. Such a feedback system is relatively simple to implement but requires many decision points from the planner for it to have learned enough to have a noticeable impact which is infeasible within the time frame of this thesis. Therefore, it is considered out of scope.

5.5 Conclusion

The main goal of this chapter was to assess the applicability of *Combinatorial Auctions* (CAs) in liquid bulk scheduling. CAs have been a useful tool for the resource allocation among self-interested agents and this is also the case for the scenario presented in this thesis, as the agents are interested in obtaining route allocations, which can be represented as bundles of items at timeslots in the auction mechanism.

There are several differences between traditional CAs and the application of CAs in this thesis. First of all, there is no possibility for implementing payments in liquid bulk scheduling due to the existing contracts between customers and terminals. This means that the designed CA is not a traditional auction, but an auction-based coordination system. Due to this difference, the auction loses its game-theoretic properties which makes it unsuitable for ensuring truthfulness of the agents. However, as Chapter 3 already described there are only few possibilities for strategic behaviour by the agents, all of which are easily verifiable.

With regards to the bidding rules for the CA in liquid bulk scheduling the agents are required to submit all bids up front, and are required to bid on discretized time slots with possible slack.

For incorporating terminal preferences in the allocation an interesting approach is to apply discrete choice models such as *(Random) Utility Maximization* ((R)UM) and *(Random) Regret Minimization* ((R)RM) for choosing the final allocation.

Part II

Algorithm Design and Implementation

Algorithm Architecture 6

The previous part of this thesis discussed the background related to implementing a scheduling system for application in a scheduling support system in the liquid bulk domain. This background research regarding the scheduling situation, the environment, and possible solution strategies has shown that *Combinatorial Auctions* (CAs) are a feasible, and well-fitting solution strategy. The implementation of such a CA system in liquid bulk scheduling will be further discussed in this part of the thesis. This first chapter will describe the overall requirements and functioning of the *Multiagent System* (MAS), while the upcoming chapters will go in depth on the design of the specific agents in it.

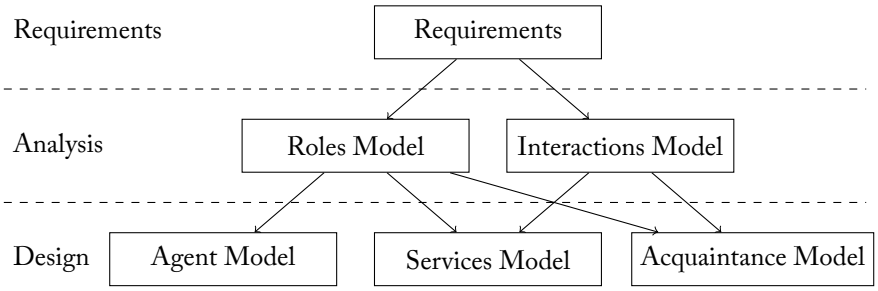


Figure 6.1: Gaia design steps (after Wooldridge, Jennings, & Kinny, 2000, p. 288)

This chapter follows the structure of the Gaia methodology for MAS design which encourages developers to think of building MASs as a process of organizational design (Wooldridge et al., 2000). It starts by summarizing and reviewing the requirements found in the previous part of this thesis. From there, it goes on to the analysis phase, in which the structure of the system is designed, consisting of several concepts such as roles, responsibilities, protocols and activities. The third part of Gaia is the design phase in which the individual agents are further expanded.

Typically, the aim of a design phase is to transform the abstract models from the analysis phase into the required level of detail for implementation. Gaia deviates from this traditional definition in the sense that the goal of the design phase in Gaia is to transform the abstract models into the required level of detail for traditional software design methods to be applied. The design phase consists of three models: the Services Model, the Acquaintance Model, and the Agent Model. The goal of the Services model is to identify the main functions of each agent. In *Object-Oriented* (OO) terminology this will be most closely related to an objects' methods. For each service, the inputs, outputs, pre-conditions, and post-conditions need to be specified. The Acquaintance design model is the simplest of the three models and describes the

communication links between agent types. The Agent models describe what agent types will be used in the system and which roles these will fulfil. Agent models will be discussed for each agent separately in the upcoming chapters. This approach is visualized in Figure 6.1.

6.1 Requirements

Based on the findings in Part I of this thesis, several main requirements for the scheduling system are found, which will be discussed in this section.

The main requirement is that the system should be able to deal with the specific characteristics of the liquid bulk scheduling problem, which are the presence of a time dimension in which vessels can arrive and depart, the concept of routes which can share infrastructure elements among them, and the presence of terminal preferences, both on individual allocations as on the allocation as a whole.

Since the system will be applied in a scheduling support system it should be able to propose multiple scheduling alternatives, preferably notably different and making different tradeoffs. Furthermore, it will often be the case that there are already several vessels planned which the system should account for when new vessels are planned. Due to the direct interaction between the planner and the system, the system should be able to propose scheduling solutions relatively fast (under 5 minutes).

Since multiple planners can use the scheduling support system simultaneously the scheduling system should be able to deal with multiple simultaneous scheduling tasks.

In order to increase re-usability the system should be easily adaptable to different terminals with regards to preferences, infrastructure and scheduling constraints.

Apart from the extended use case in which multiple vessels are allocated at once, a nice feature would be to plan single vessels efficiently.

A final desirable feature would be supporting re-optimization, in which earlier planned vessels are kept in place as much as possible but moved if needed. This feature is considered out of scope for this thesis.

6.2 Roles Models

The Roles model is used to identify the key roles present in the MAS. A role can be seen as an abstract representation of functionality such as “seller” or “buyer”.

A role is characterized by four attributes: responsibilities, permissions, activities and protocols. Responsibilities define the key function the role needs to perform and is thereby the most defining characteristic of a role. Responsibilities are defined by liveness expressions which define possible execution trajectories for the role. The ω symbol defines that an expression is repeated infinitely. In order to fulfil the responsibilities for a certain role, a role requires permissions. Permissions include for example reading information, interacting with the environment, or generating information. The activities of a role are a set of private actions in the sense that they

cannot interact with other agents. Protocols indicate the manner in which a role interacts with other roles. For example, a seller role can have a CA protocol.

6.2.1 Contract Net Protocol

The main goal of the MAS is to solve a *Job-shop Problem with Multiprocessor Tasks* (JMPT) problem tailored to the liquid bulk domain. We have seen in the previous part of this thesis that CAs provide a good fit on the scheduling problem and are a promising candidate for implementation. A common method for implementing auctions in MASs is the *Contract Net* (CNET) protocol. CNET is a high-level protocol which is based on how organizations put contracts to tender and was first proposed by Smith (1977) and expanded in (Smith, 1980a, 1980b). CNET is an example of a reverse (or procurement) auction in which the auctioneer proposes a task to be performed, and agents can bid on performing the task. Typically, the lowest bid in such a scenario wins the auction.

CNET starts with a *manager* role, who first announces a *task*. The manager typically does a *global broadcast* to all other agents in the system, although if the manager possesses knowledge on the capabilities of the other agents, it can issue a *limited broadcast* to the agents he knows can carry out the task. Agents receiving the message will evaluate the tasks and determine whether they can perform them. Based on this information, the agents submit *bids*. The manager typically receives multiple bids and selects the most appropriate agents to perform the task. This selection is communicated through an *award* message. Agents selected for performing a task are called *contractors*. After completion of the task, the contractor *reports* to the manager (after Smith, 1980a, p. 62, 63).

6.2.2 Contract Net in Liquid Bulk Scheduling

Several adjustments need to be made on the CNET protocol for it to be suitable for application in this scenario. First of all, CNET describes a reverse auction, in which the manager has a task he wants to be performed and contractors have resources to perform that task. In liquid bulk scheduling this is the other way around as typically the vessels have tasks they wish to be performed while the terminal has the required resources for performing the tasks. Therefore, instead of the manager broadcasting a task he wishes to be performed, he broadcasts a new planning round in which vessels can participate.

The role of contractor in liquid bulk scheduling is filled in by a Vessel role. The vessel has a task to perform (such as loading or discharging the vessel) and requires terminal infrastructure to perform that task. When a new planning round is started, the vessel role can subscribe to this planning round by bidding on a set of infrastructure elements which can together perform its task. In order to know which set of infrastructure is required to perform the tasks, the Vessel role requests a set of route options from a Route Finder role.

The Route Finder role is responsible for receiving route option requests and determining a set of feasible routes using these requests and specific knowledge on the terminal infrastructure.

When the bids from the vessel roles are submitted to the manager role it is possible to generate allocation options. The Planner role retrieves the bids and uses these in combination with its knowledge on the terminals preferences to draft several route options. These route options are then presented to the PlannerInterface role, which is responsible for interacting with the human planner. The human planner can select his preferred option which the PlannerInterface will communicate with the Planner role. The Planner then communicates this to the Manager, who in turn sends all acknowledged contracts to the Vessels. This process is also visualized as a sequence diagram in Figure 6.2.

In Step 1, a scheduling job is initiated by the planner. A Planner Agent will then be spawned, which in turn spawns the required Vessel agents (2). This agent hierarchy is in place to allow for easier error handling and fault tolerance in case one of the Vessel agent fails by applying Agent Supervision strategies. Once the Vessel agents are initialized, they are asked by the Planner agent to prepare their bids on routes (3). In order for the Vessel agents to make bids, they ask the Route Finder agent to find them appropriate routes (4). When the route finding is complete, the routes are returned to the Vessel agents, which in turn evaluate them and submit these evaluations as bids to the Planner agent (5). The Planner agent then uses these bids to determine a feasible and desirable allocation. Once this process is completed, the Planner agent presents a set of these allocations to the human planner for his final decision (6). Finally, the human planner chooses one of the proposed allocations and confirms it to the Planner agent, which in turn propagates it to the scheduling support system.

Apart from the full use-case, a simple use-case in which the human planner only requires route options for a single vessel is also supported. An extensive overview of the roles defined for this MAS is available in Appendix F.

6.3 Agent model

An agent in Gaia is best represented by a set of agent roles. While it is possible to have one-to-one mappings from the agent to agent roles, this needs not to be the case. In several cases it is beneficial to combine several closely related roles into the same agent type. No strict guidelines for mapping agent roles to agents are available and this step is mostly left to the discretion of the designer.

Given the role models as described in the previous paragraph it makes sense to combine the manager role, the Planner role, and the PlannerInterface role into the same agent as these are closely related in functionality and require the same information.

Keeping the Route Finder and the Vessel agent separate in different agent types is beneficial for several reasons. First of all, since vessels arrive and depart over

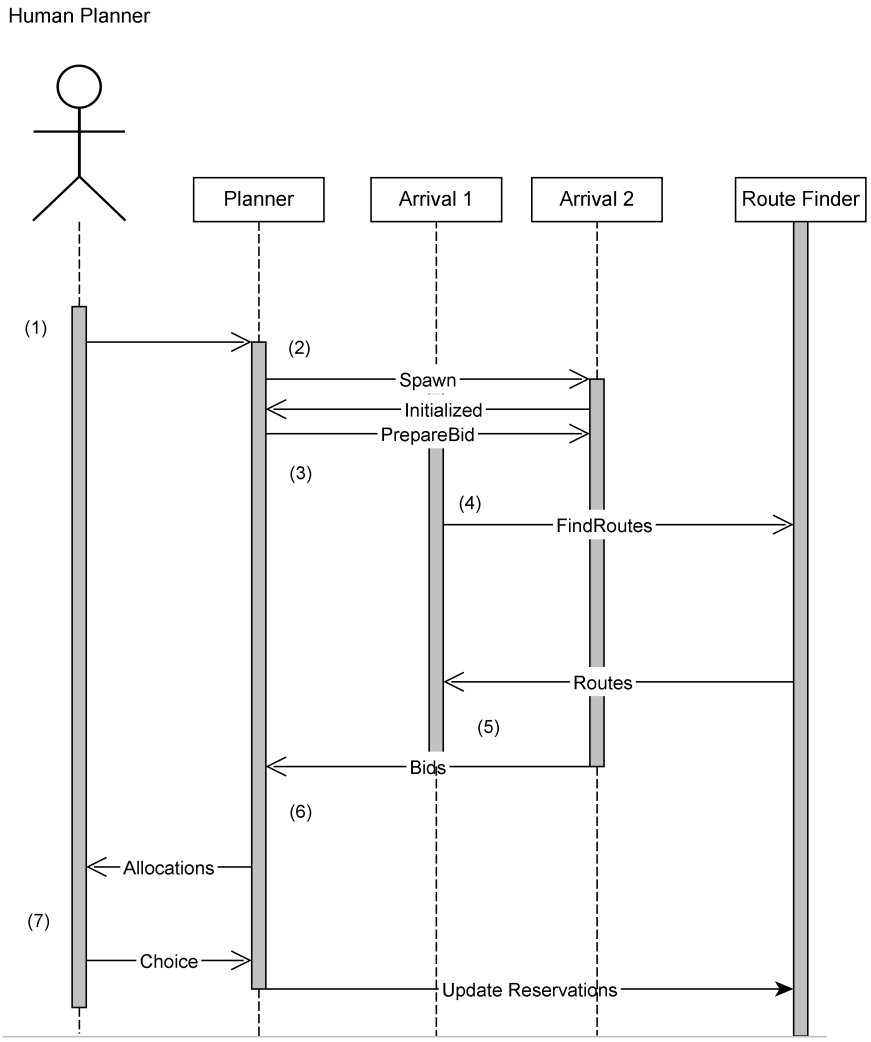


Figure 6.2: Sequence Diagram of algorithm functionality

time and are therefore typically not part of multiple scheduling rounds the agents representing them need to be spawned and destroyed quite often. On the contrary, the Route Finder role typically runs for the entire duration of the system, and only needs change in the rare case the infrastructure at the terminal changes. Furthermore, several optimizations on the Route Finder can be made such as the application of memoization, which results in the Route Finder remembering routes he has searched for in the past and thereby increasing the speed of the algorithm over time.

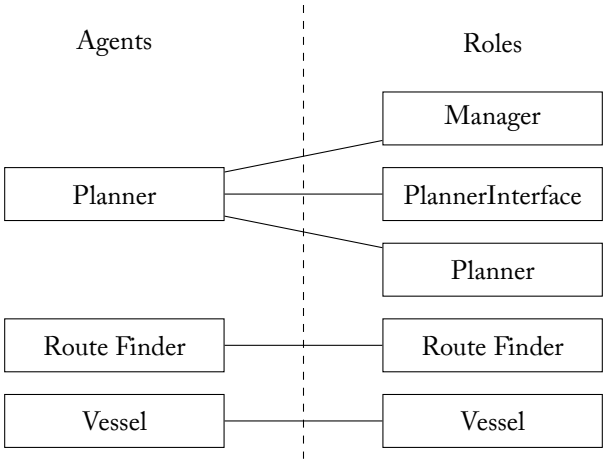


Figure 6.3: Agent model

6.4 Services model

The services model in Gaia specifies the main functions each agent role performs. For the Route Finder agent, four main services have been identified. These are building its internal model of the terminals infrastructure, pre-processing the model for the specific route finding problem instance, finding the routes, and filtering infeasible routes. The Route Finder determines whether a route is feasible solely on fixed, non-schedule related, information such as maximum berth lengths and product incompatibilities. The Vessel agent only has two services: Obtaining feasible routes and evaluating them. Finally the Planner agent has four main services: Starting a new scheduling round, finding an allocation, evaluating an allocation, and determining whether an allocation is feasible.

Appendix F contains a table summarizing all services the agents can perform including inputs, outputs, pre-conditions, and post-conditions.

6.5 Acquaintances model

The Acquaintances model defines the communication links between agent types, typically represented as a directed graph. The acquaintances model for this application is visualized in Figure 6.4. This Acquaintance model introduces a Route Finder mailbox, which receives all messages intended for the Route Finder agents to aid with distribution and parallelization.

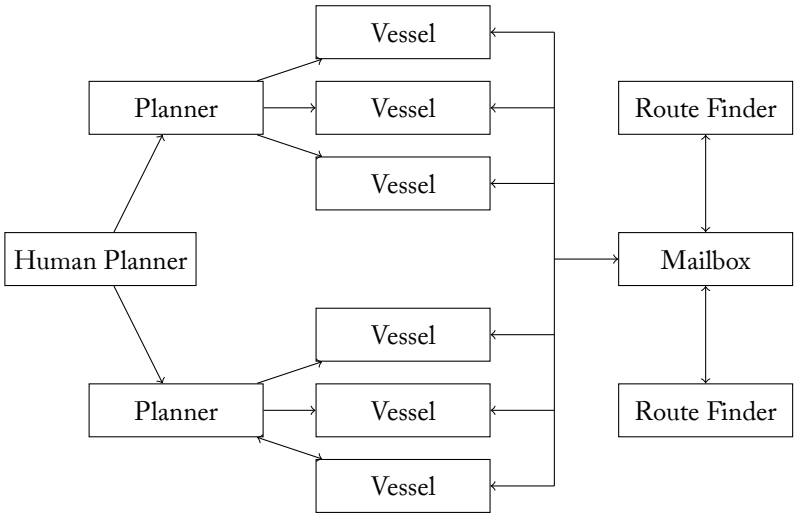


Figure 6.4: Acquaintances model

6.6 Conclusion

The goal of this chapter was to transform the requirements and general design directions from the first part of this thesis into a intermediate level design for the basic functionality and interactions of the MAS. This was done by using the Gaia model which defines roles, agents and acquaintances. Five roles were identified: the Manager role, Planner role and PlannerInterface role, which were compacted into a single Planner agent; the Vessel role, which was translated one-to-one to a Vessel agent; the RouteFinder role, which also was translated one-to-one into a RouteFinder agent.

The upcoming three chapters will describe for each of these agent types their detailed design and implementation.

The Route Finder is the first main element of the liquid bulk scheduling solution. The Route Finder should be able to give, for a specific vessel arrival, a set of feasible and efficient routes for loading or discharging product. As discussed in the previous chapter, the Route Finder agent is not a direct representation of a stakeholder in the liquid bulk scheduling process, and therefore does not have desires or goals.

This section will describe in detail the design and implementation of the Route Finder agent based on the role description from the previous chapter (Table 7.1). It starts by discussing the overall requirements of the Route Finder agent, it then goes on with describing the implementation used to fulfil those requirements.

7.1 Requirements

The main goal of the Route Finder agent is to find, for a specific parcel at a point in time, feasible and efficient routes for processing that parcel. Theoretically it would not be required for the Route Finder agent to provide efficient routes as the rest of the scheduling system should enforce the efficiency of allocations. However, a strictly worse route (a route that requires all the infrastructure elements of another route, along with extra infrastructure) will never be preferred over its dominating counterpart. By keeping these types of routes out of the scheduling process the overall performance of the system can be improved.

Table 7.1: Route Finder role

Role: Route Finder
Description: The Route Finder uses task details to find possible routes on the terminal
Protocols and Activities: AwaitAskRoutes, FindRoutes, SubmitRoutes
Permissions: Read: TerminalInfrastructure Generate: Routes
Responsibilities: ROUTE FINDER = {AwaitAskRoutes, FindRoutes, SubmitRoutes}ω

A second requirement is that the Route Finder should be able to work around existing, frozen, reservations. Furthermore, it should be able to deal with other hard constraints such as berth restrictions for certain vessel lengths. Once again, this could be handled further along in the scheduling process but imposing hard scheduling constraints as early as possible will improve the overall performance of the solution.

Another important aspect of the Route Finder agent is that it should be able to give accurate estimations of processing times. These processing times will depend partly on the type of ship, product, and operation as well as on the route suggested by the Route Finder.

Valves in lines are considered out of scope for the Route Finder agent. It is assumed that each single defined infrastructure element can be used independently of all other defined infrastructure elements.

The main non-functional requirement is that the Route Finder is fast. Since the Route Finder will be called by several vessel agents at once at the start of a planning round this could for example be achieved by making it parallelizable.

7.2 Implementation

As described in Chapter 6, the responsibilities of the Route Finder are (Await-AskRoutes, FindRoutes, SubmitRoutes)ω. This functionality is further expanded into a *Finite State Machine* (FSM) as shown in Figure 7.1. This figure shows that after startup, the Route Finder agent will retrieve the infrastructure definition of the terminal. It then goes to an idle state until it has received a AskRoutes message. After receiving the message it goes on with computing route options, processing time estimations, and route timeslots. If no route options or timeslots are found a "No Routes" message is sent to the vessel. This section describe the implementation of these parts in depth.

The Route Finder agent is implemented as a type of model-based agent as described by Russel and Norvig (2009) where the internal model of the agent is a representation of the terminals infrastructure and existing reservations.

7.2.1 Infrastructure Definition

In order for the Route Finder agent to find routes at the terminal it needs to know the exact infrastructure situation at the terminal. Retrieving this infrastructure information, and transforming it into a model the Route Finder agent can use, is done in the "get infrastructure definition" state.

The scheduling support solution as under development by Systems Navigator will include a detailed infrastructure specification which the Route Finder agent can use for generating its model. For this thesis, a simple, fictitious terminal layout is used which is visualized in Figure 7.2. This fictitious terminal layout is chosen for its simplicity which makes understanding the system easier, as well as because of

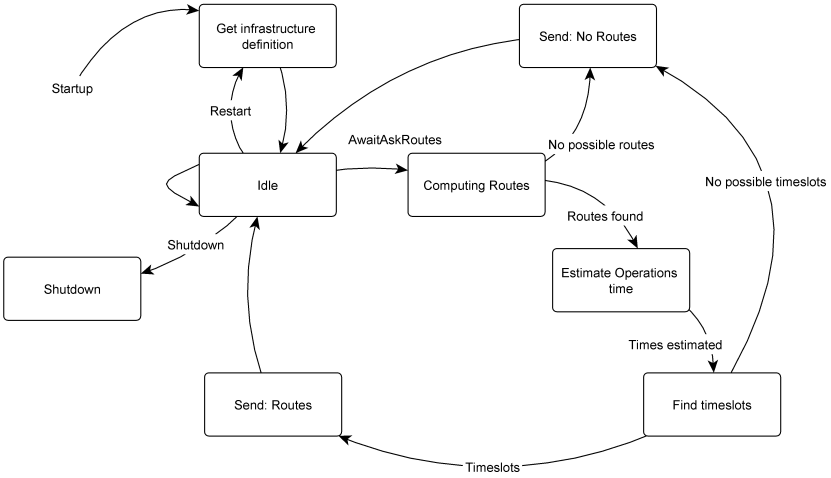


Figure 7.1: FSM Diagram of Routefinder agent functionality

confidentiality issues with real layouts. Appendix E provides the full definition of this example terminal including all relevant infrastructure attributes.

After reading in the infrastructure elements definition the Route Finder agent needs to transform this data into a model which it can use to perform efficient route finding calculations with. The infrastructure at a terminal is a network consisting of many, connected, infrastructure elements. Therefore, an obvious choice is to represent the infrastructure at a terminal as a graph since that allows us to use existing graph algorithms to perform the actions as required of the Route Finder.

Since the infrastructure surrounding pumps is one-directional, as pump headers can be suction pipe connections, pressure pipe connection or bidirectional connections, the infrastructure graph would be a directed graph. At first it seems obvious to model lines as edges of the graph and the other components as vertices. However, since lines can be connected to numerous other lines, it is more useful to model all elements as vertices with infinite capacity connector edges. The weight of the vertices will be used to represent rough estimates of the pumping capacity of an infrastructure element. An example graph representation from the example terminal is shown in Figure 7.3.

7.2.2 Route Finding

After an AskRoutes message is received by the Route Finder agent, it will attempt to find routes for that specific parcel. The Route Finder agent starts out by performing a preprocessing task on the infrastructure graph, in which a global source and sink node are added to the graph and connected to all possible tanks for the parcel, and all possible berths the vessel can be allocated. Furthermore, infrastructure which is

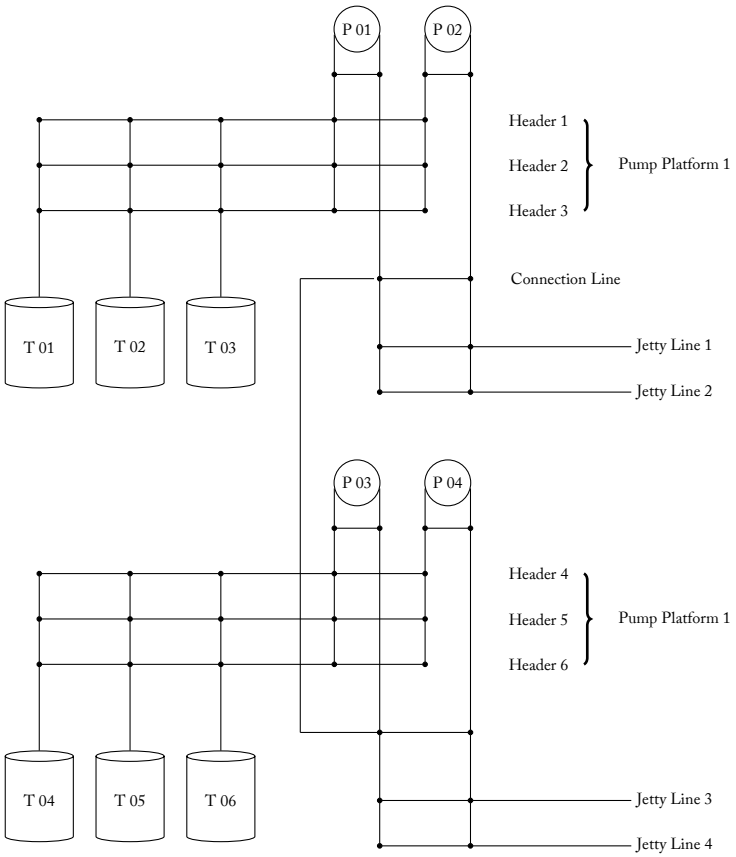


Figure 7.2: Common infrastructure layout

dedicated to product types other than the product type of the parcel are removed to increase efficiency.

The route finding problem itself is a so-called widest path problem, or bottleneck shortest path problem. The goal of this problem is to find a path between two vertices in a graph which maximizes the minimum-weight edge of the graph. An efficient algorithm which runs in $O(n)$ time where n is the number of vertices in the graph is given by Punnen (1991). However, this algorithm only works on undirected graph which makes it unsuitable for application in this scenario.

Another method for solving the widest path problem is an obvious modification of Dijkstra's shortest path algorithm (for example described in (Blum, 2008)), in which the addition operator in the distance estimator is replaced with a **min** function, and the **min** function by a **max** function. Using an efficient implementation of Dijkstra using priority queues this algorithm runs in $O(n \log n)$ where n is the

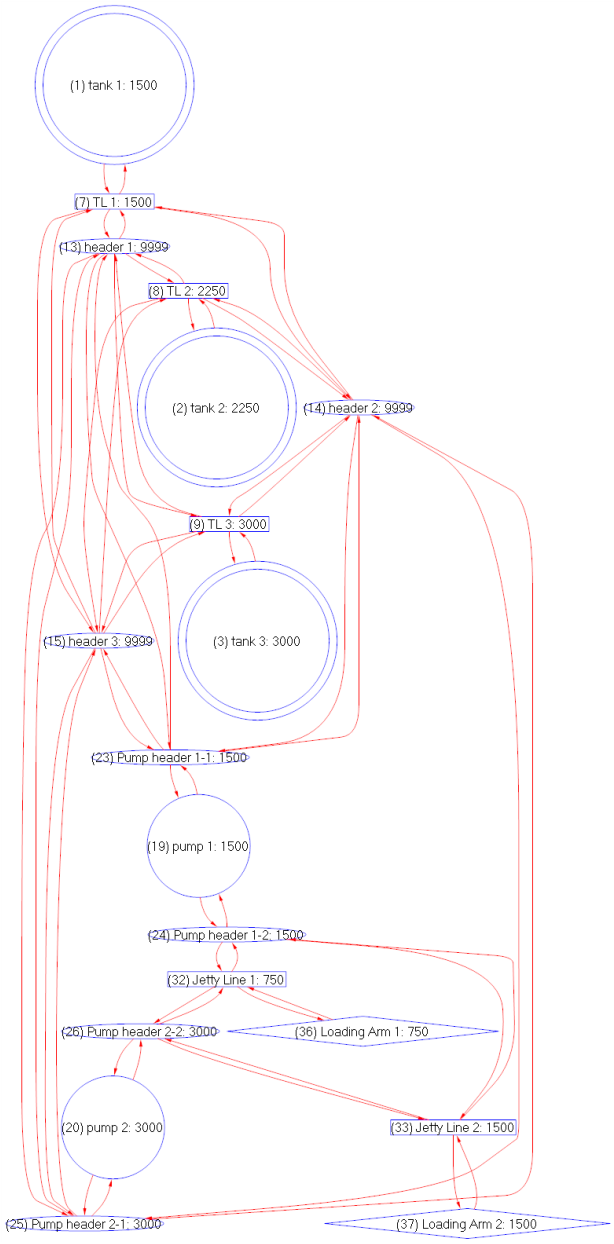


Figure 7.3: Example graph representation of terminal layout

number of vertices.

As it is unlikely that a single shortest path for a parcel will always be available, the Route Finder agent should be able to find multiple shortest routes. To do this, Yens algorithm is used to efficiently find k-shortest loopless paths (Yen, 1971). The total runtime complexity of this algorithm becomes $O(kn(m + n \log n))$ where n is the number of vertices in the graph, m the number of edges, and k the number of paths required.

After the routes are found, some post-processing is required to filter infeasible routes. The rules used for filtering, based on the background research described in Part I of this thesis, are:

- Discard the route B if there exists an $A|A \subset B$;
- Discard the route if it contains a land-side pump and is for a discharge operation;
- Discard the route if it does not contain a land-side pump and is for a loading operation.

Another post-processing step is checking the availability of parallel pumps. In certain cases it is possible to load the vessel by using two pumps in the route simultaneously. If this is possible, an extra route is added containing the two pumps with adjusted pumping speeds.

7.2.3 Turnaround time estimation

After routes are found the next step for the Route Finder is to accurately estimate the turnaround time for those routes. This section describes estimations for turnaround times for the four main tasks: arriving, pre-pumping, pumping, and post-pumping. For a full description of the individual steps in the operation phases the reader is directed to Chapter 3.

Arriving, Pre-, and Post-Ops

Operations in the arriving, pre-pumping, and post-pumping step are largely independent of the chosen route and can therefore be estimated solely based on the parcel and vessel properties. For these three stages, planned times based on these attributes will be used. These timings are based on data collected from various terminals studied earlier by Systems Navigator BV.

Arriving takes approximately 2-3 hours depending on vessel size. During this time no vessel movement can take place on all berths surrounding the target berth. Pre-pumping takes 1-3 hours depending on the parcel size and product types and post-pumping takes 2-3 hours depending on vessel size. Manoeuvring out of the port after post-pumping is about 0.5 hours. More detailed information regarding operations timing is given in Appendix B.

Pumping
The largest component of the turnaround time consists of pumping time, which depends heavily on the chosen route. Several options for calculating the pumping time of a route are available, which vary in computational complexity and accuracy.

The simplest option for pumping time estimations is using fixed capacity values for all infrastructure elements. Most terminals have a general idea on the maximum flow rates of their infrastructure per product type, and using this as an estimation will generally give reasonable answers. Initially, the Route Finder agent will use these estimations as baseline values in the infrastructure graph to quickly find routes with an indication of their pumping speed. After this initial pass-through is done and the total routes are known, it is possible to use fluid dynamic calculations to further improve the accuracy of the pumping time estimation. Important parameters for the flow rate are the pipe dimensions (diameter and length), oil product properties, difference in elevation, and pressure difference. A well known formula that relates these parameters to the flow rate is the Darcy Weisbach equation.

Apart from the solution from the Darcy Weisbach equation there are several practical constraints which need to be considered. In reality the smaller vessels (< 20 kDWT) and barges are not loaded with a rate higher than 700 m^3 per hour to avoid instability of the barge. Furthermore, the unloading pumping rate is limited due to the pumping facilities on the tanker. These practical constraints are implemented in the model by putting a cap on the flow rate for small tankers.

In practice it turned out that using the Darcy Weisbach equation did not result in major accuracy gains compared with the simple case where flow rates were set as averages per product type. A major factor in the inaccuracies of the Darcy Weisbach equation for liquid bulk most likely originated from the fact that it does not take into account the connections between different infrastructure components. At these connections there are often many losses due to bends and decreasing pipe diameters. A full network flow solver which does take into account these types of connections would possibly increase the accuracy of the calculations. However, this is out of scope for this thesis.

7.2.4 Timeslot finder

The final step for the Route Finder agent is to use the found routes and operations times to find available timeslots given earlier reservations.

The first step in this process is to find, for each infrastructure element in the route, possible earlier reservations and schedule these time periods on a time line which starts from the *Estimated Time of Arrival* (ETA) of the vessel and ends at the end of the planning horizon. The periods on these time line are then ordered and transformed into a time line balance as shown in Figure 7.4. Each period where the time line balance is at zero, is a possible timeslot for the route. These gaps are then compared with the estimated turnaround time and filled using a discretized time period of one hour. All these possible timeslots are then submitted to the vessel agent.

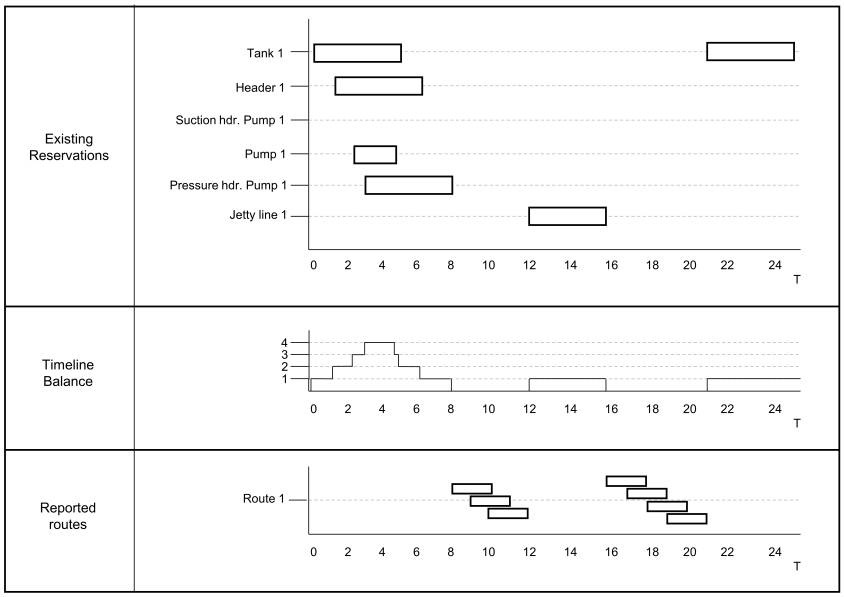


Figure 7.4: Timeline balance and reported routes

7.3 Conclusion

The Route Finder agent is an implementation of the Route Finder role as described in Chapter 6. It is implemented as a *Finite State Machine* (FSM) which finds widest-path routes, estimates processing times, and finds available timeslots for the routes, given the processing times.

The Route Finder uses a internal model of the terminal which is represented by a directed, cyclic graph in which each infrastructure element is a (weighted) vertex and connections between infrastructure elements are given by infinite capacity edges. the widest-path problem is solved by preprocessing the graph, using Yen's algorithm in combination with a modification of Dijkstra's shortest path, and post-processing the routes to filter infeasible options.

The estimation of turnaround times is done based on predetermined values for arriving, pre-pumping, and post-pumping depending on the vessel type and the product it is carrying. The pumping time estimation can be done based on predetermined values, or using the Darcy-Weisbach equation, depending on terminal preferences.

The Route Finder then uses the estimations on turnaround time to find available gaps for the routes given the existing reservations. This is discretized to one hour time periods and implemented using a time-line balance.

Vessel Agent 8

The Vessel agent is the implementation of the *Vessel* role as described in Chapter 6. The Vessel agent represents the customer stakeholder identified in Chapter 3. Its main goal is to represent the customers values in the scheduling process. The main activity performed by the vessel agent is the valuation of route options received from the Route Finder agent.

The main goals identified from the customer stakeholder are:

1. Short turnaround times;
2. Short dwell times;
3. Maintaining product quality.

The Vessel agent will mostly focus on the first two of these goals as it does not have the information needed for the third goal.

In the current implementation of the vessel agent, the route valuation is relatively simple and only based on the departure time of the vessel, since more detailed implementations would require more interaction with actual terminal customers which wasn't feasible within the time-frame of this thesis. However, in the future this can be easily expanded to take more features into account. Bids for the routes are currently calculated as follows:

Table 8.1: Vessel role

Role: Vessel
Description: The vessel participates in planning rounds and submits bids on required operation resources.
Protocols and Activities: AwaitTask, AskRoutes, AwaitRoutes, EvaluateRoutes
Permissions: Read: TaskDetails Generate: Bid
Responsibilities: VESSEL = {AwaitTask, AskRoutes, AwaitRoutes, EvaluateRoutes, SubmitBid}

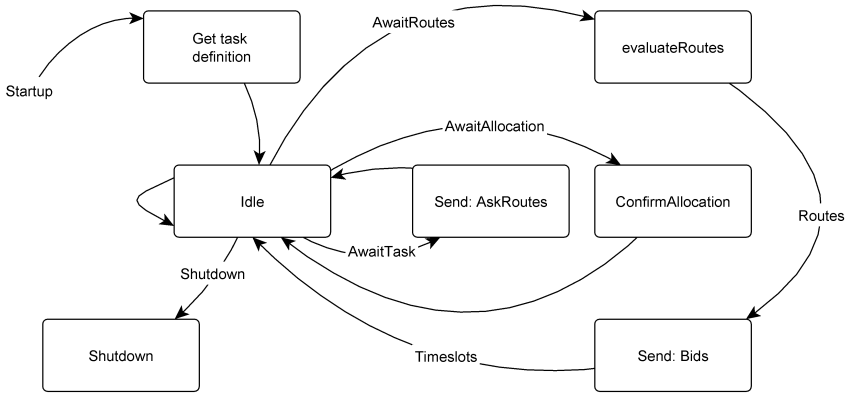


Figure 8.1: FSM diagram of Vessel agent

given an ordered set $R = \{r_0, r_1, ..., r_n\}$ containing route * timeslot combinations where $r_i.End \leq r_{i+1}.End$

$$v_{ri} = 1000 * (1 - \frac{0.2 * r_i.End - (r_0.Span + Arrival.ETA)}{(r_0.Span + Arrival.ETA)}) \tag{8.1}$$

A resulting set of bids is visualized in Figure 8.2

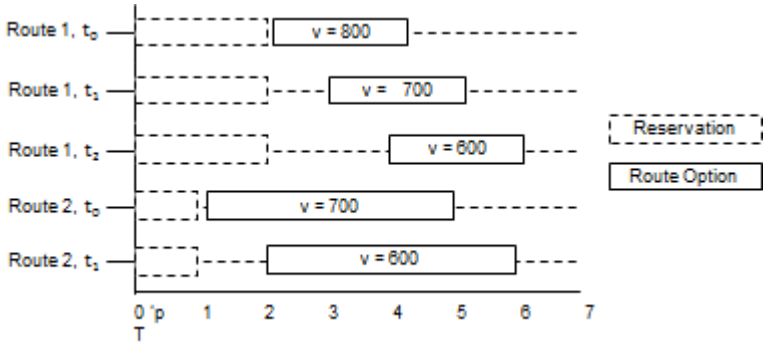


Figure 8.2: Bid making

Planner Agent 9

The Planner agent is a combination of the *Manager* role, *Planner* role, and *PlannerInterface* role. It is responsible for taking the bids from the Arrival agents and combining these into a feasible allocation taking into account the valuations of the bids by the agents as well as the preferences of the terminal. In *Combinatorial Auction* (CA) literature this type of problem is often referred to as the *Winner Determination Problem* (WDP). The Planner agent is a representation of the terminal stakeholder as discussed in Chapter 3 and is designed to represent its goals. The goals from the terminal side can be summarized as follows:

1. High terminal utilization;
2. Low impact on surroundings;
3. Low pressure on operations;
4. Not paying demurrage costs;
5. Receiving despatch costs.

As was discussed in Chapter 5, a *Discrete Choice Model* (DCM) will be used for determining the value of possible allocations. The agent architecture therefore will be a utility-based architecture which can function using either *(Random) Utility Maximization* ((R)UM) or *(Random) Regret Minimization* ((R)RM) models.

This chapter will describe in detail the functioning and implementation of the planner agent. It starts out by exploring several possibilities for solving the WDP, it then goes on to describe the method by which the feasibility of an allocation is determined. In the third section, the implementation of the terminal preference elicitation is described and finally in the conclusion a comparison of the various design choices is given.

9.1 Winner Determination Problem

We have seen in Chapter 5 that winner determination is NP-Hard (Lehmann et al., 2006), which means that in order to still find a solution we need to give up one of three desired features (Kleinberg & Tardos, 2011):

1. Solve arbitrary instances of the problem;
2. Solve the problem in polynomial time;
3. Solve the problem to optimality.

The remainder of this section will explore for each of these features whether releasing them is a suitable approach for the WDP in liquid bulk scheduling. When evaluating solution strategies there are four main criteria which are interesting to consider: (Russel & Norvig, 2009).

1. **Completeness:** Is the strategy guaranteed to find a solution if it exists?
2. **Time Complexity:** How long does it take to find a solution?
3. **Space Complexity:** How much memory is needed to find a solution?
4. **Optimality:** Is the highest-quality solution found when there are multiple solutions?

9.1.1 Problem set limitation

Muller (2006) and Rothkopf, Pekeč, and Harstad (1998) describe several cases where restrictions on the auction can ensure that the winner determination problem is tractable. [1]

For example, in the case of a CA with heterogeneous items, OR bids and a unit supply, the WDP can be solved effectively using a *Linear Programming* (LP) solver, as long as the LP has an integral optimal solution. Literature suggests two main ways to enforce this integral optimal solution. The first way limits the items on which agents can bid on by for example completely ordering them and enforcing that bids are only placed on adjacent sets of items. The second method restricts bid values so that they imply an integral optimal solution. An example of bid value restriction is to make bid values completely additive for single items. [2]

Unfortunately, in the work described by and referred to by Muller and Rothkopf, no feasible constraints were found that limit the solution space in a way that the winner determination problem for the case in liquid bulk scheduling could be made tractable. This means that releasing the first feature will not result in finding a solution for the WDP in liquid bulk scheduling.

[1] "There exists an algorithm that optimally solves any instance in the class requiring a number of operations that is polynomially bounded in the encoding length of the instance" (Muller, 2006, p. 319)

[2] Given a bundle of items $S = \{m_1, m_2, m_3\}$ for bidder i with valuation $v_i(S)$, the bid values are completely additive when, $v_i(S) = v_i(m_1) + v_i(m_2) + v_i(m_3)$

9.1.2 Non polynomial runtimes

The second desired property which can be given up when attempting to solve NP-hard problems is solving the problem in polynomial time. Even though releasing this constraint will cause the run time of the solution to grow very fast as the problem size increases, the solution might still be fast enough in practice. This section will explore two common methods which work by releasing the run time constraint: Exhaustive Search and *Branch and Bound* (B&B).

9.1.2.1 Exhaustive Search

Even though the problem is NP-hard it can still be feasible to simply calculate all the options, given that the necessary operations per option are fast enough. Simply calculating all options and then selecting the best one is referred to as exhaustive search, or brute forcing. Determining whether brute forcing a solution to this winner determination problem is feasible depends on two factors: The number of allocations which needs to be checked for a certain problem size, and the time needed to process an allocation.

An allocation consists of a selected route and timeslot for each vessel in the set $V = \{v_1, v_2, v_n\}$. When considering a reasonably small terminal with dedicated infrastructure for all different product types, we find a set R of about 20 feasible routes per arrival: $R = \{r_1, r_2, ..., r_{20}\}$. Given a planning horizon of three days and an average of 12 hours of pumping time we get a timeslot set S consisting of the elements $S = \{s_1, s_2, ...s_{60}\}$. A route allocation is then the cross product of set S and R.

$$S \times R = \{(r_1, s_1), (r_1, s_2), ..., (r_{20}, s_{60})\} \tag{9.1}$$

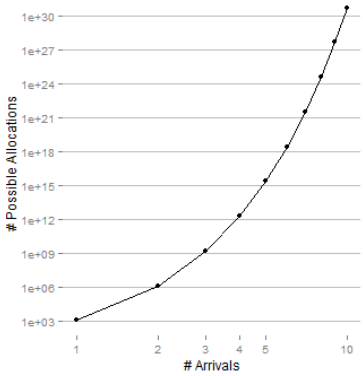
Given that the order in which vessels appear in an allocation is not relevant (the allocation $\{(v_1, r_1, s_1), (v_2, r_2, s_1)\}$ equals the allocation $\{(v_2, r_2, s_1), (v_1, r_1, s_1)\}$), the total number of possible allocations is $|(S \times R)|^{|V|}$.

For our reasonably small terminal the number of allocations which needs to be checked for a varying number of arrivals is given in Figure 9.1.

For each possible allocations there are two main steps. First, it must be determined whether the allocation is feasible with regards to infrastructure constraints (described in detail in Section 9.2). Second, the value of the allocation needs to be calculated (described in detail in Section 9.3).

Even though efficient algorithms are available for performing these steps, they are still relatively costly which means that although brute forcing is a feasible alternative for small problem sizes (a small terminal and less than five vessels), it will not be sufficiently fast in practice.

Figure 9.1: Number of possible allocations at a small terminal for a varying number of arrivals



Arrivals	Allocations
1	1200
2	$1,44 * 10^{06}$
3	$1,73 * 10^{09}$
4	$2,07 * 10^{12}$
5	$2,49 * 10^{15}$
10	$6,19 * 10^{30}$
15	$1,54 * 10^{46}$
20	$3,83 * 10^{61}$

9.1.2.2 Branch and Bound

B&B is a search paradigm for combinatorial optimization problems, which was first introduced by J. D. C. Little, Mury, Sweeney, and Karel (1963) as an efficient solution for the Travelling Salesman Problem. A B&B algorithm uses a systematic method to explore candidate solutions and a bounding function, which checks these candidate solutions against estimated lower and upper bounds and discards them when they are expected to not yield a better solution than the currently found solution.

At any given moment during the runtime of a B&B algorithm, the current state can be described as a pool of yet unexplored parts of the solution space and the best currently known solution. At the start of the run, only one subset exists - the complete solution space - and the best known solution is ∞ if a lowest cost solution is desired, or 0 for a highest value solution. The unexplored parts of the solution space can be represented as nodes in a search tree, and each iteration of the algorithm will process one of these nodes (Clausen, 1999). Each iteration consists of three parts: selecting a node to process, calculate the bounds, and branch the solution tree. Hence, a B&B algorithm consists of three main components:

1. **A bounding function:** provides for a certain subspace of the solution space a bound for the best solution value obtainable;
2. **A selection strategy:** Selects the solution subspace to be selected for investigation in the current iteration;
3. **A branching strategy:** The strategy by which to further investigate a subspace when the subspace cannot be discarded.

B&B algorithms are common for solving optimization problems as they always converge to an optimal solution if the solution space is finite and are therefore complete. The worst-case time complexity of B&B is equal to that of an exhaustive search, but in practice they mostly depend on the bounding function and selection strategy which will be described in the upcoming paragraphs.

Bounding function
The bounding function is one of the key elements of any B&B algorithm since it most strictly determines its performance (Clausen, 1999). The goal of the bounding function is to give, for a given subset of the solution space, a value which is as close as possible to the best feasible solution of the sub-problem. If the bounding function generally finds values close to the optimum it is called *strong*, if it is generally far away from the optimum it is *weak*. There is often a trade-off between the strength of the bounding function and its runtime. The more time available for calculating bounds, the better it often is. Spending more time on calculating a stronger bound is often beneficial as reducing the number of nodes to be processed often yields larger runtime improvements than making the bounding function slightly faster.

Several special purpose bounding methods are proposed for use in the WDP (Sandholm, 2002a; Sandholm et al., 2002; Fujishima, Leyton-Brown, & Shoham, 1999) which are based on using as an upper bound the sum an item's maximum contribution for all unallocated items (Sandholm, 2000, p.347):

$$\sum_{i \in A} c(i), \text{where } c(i) = \max_{j | i \in S_j} \frac{p_j}{|S_j|} \tag{9.2}$$

Where A is the set of unallocated items, S_j a set of items and $p_j \geq 0$ is the price for that set. This upper bound can then be calculated by an LP of the following form:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} p_j x_j \\ \text{s.t.} \quad & \sum_{j | i \in S_j} x_j \leq 1, \forall i \in \{1..m\} \\ & x_j \geq 0 \\ & x_j \in \mathbb{R} \end{aligned} \tag{9.3}$$

which can be solved in polynomial time in the size of the input using *Common of the Shelf* (COTS) software. A major weakness of this approach is that, since it does not take time into account, it can vastly underestimate the upper bound as in this scheduling solution items are re-used over time. This effect can be mitigated by instead of using the set of infrastructure elements M as A, using the cartesian product of the set M with all possible hourly timeslots in the planning horizon $M \times T$. When considering a planning horizon of 5 days with time discretized to

hourly intervals, this means that the input size of the problem would be increased 120 times, thereby increasing the run time of the bounding function.

Another commonly used method for determining an upper bound is to solve a simplified version of the problem in which most of the constraints of the original problem are released, while maintaining the essence of the problem. In liquid bulk scheduling this could be done by releasing all infrastructure constraints except for the berths. This transforms the scheduling problem from a *Job-shop Problem with Multiprocessor Tasks* (JMPT) problem to a regular job-shop problem which can be solved more efficiently than the original problem.

A job-shop problem is characterized by a set of jobs $J = \{j_1, j_2, ..., j_n\}$ which need to be scheduled on m identical machines, while minimizing makespan. A small extension on this problem needs to be made as jobs have a release date, the earliest moment the job can start. Although a job-shop problem with $m \geq 2$ was proven to be NP-Complete by Garey, Johnson, and Sethi (1976) it can still be solved relatively efficiently using for example LP solvers (Bülbül & Kaminsky, 2013) or heuristics (Jansen, Solis-Oba, & Sviridenko, 2003) which makes it a good candidate to use for upper bound calculations.

The observant reader might at this point have noticed that both these upper bound calculations are solely focused on the bids the planner received from the arrival agents. Since the additional attributes are only able to reduce the total allocation value, not increase it, this still ensures that the bounding function will be an upper bound, although it will be relatively weak.

To date, several studies (Buer & Pankratz, 2010; Sourd & Spanjaard, 2008; Rollon & Larrosa, 2009; Bérubé, Gendreau, & Potvin, 2009) have proposed methods to deal with multi-attribute constraints in B&B applications using for example the ϵ-constraint method to calculate Pareto boundaries as introduced by Chankong and Haimes (1983). However, as was already discussed in Section 5.4, the Pareto optimum does not per definition equal the desired allocation. Therefore, using the methods suggested in literature can result in a bounding function which does not give a strict upper bound, thereby reducing the ability of the B&B to find optimal solutions.

Experimentation with the discussed upper bounding functions will be performed and discussed in Chapter 11.

As a lower bound of the function it is intuitive to use a simple heuristic function to generate a feasible solution. In this case, a *First-come, first-served* (FCFS) allocation heuristic will be used to determine the lower bound of a node.

Selection Strategy

The selection strategy of a B&B algorithm determines the next sub-problem which gets checked in an iteration. Any standard search tree algorithm such as *Best-first Search* (BeFS), *Breadth-first Search* (BFS), and *Depth-first Search* (DFS) can be used in B&B algorithms. The choice between these strategies is usually made on a trade-off between time and space complexity. Since the selection strategies are

mostly similar to those described in the upcoming informed tree search section, the reader is assumed to be familiar with the different strategies.

Branching Strategy
The branching strategy decides how to subdivide a part of the search space when it is not discarded by the bounding function. In case the subspace is divided into two parts, it is referred to as *dichotomic* branching, else it is referred to as *polytomic* branching. In this case it is logical to apply polytomic branching and divide the subsets based on the still to be assigned vessels.

9.1.3 Releasing optimality

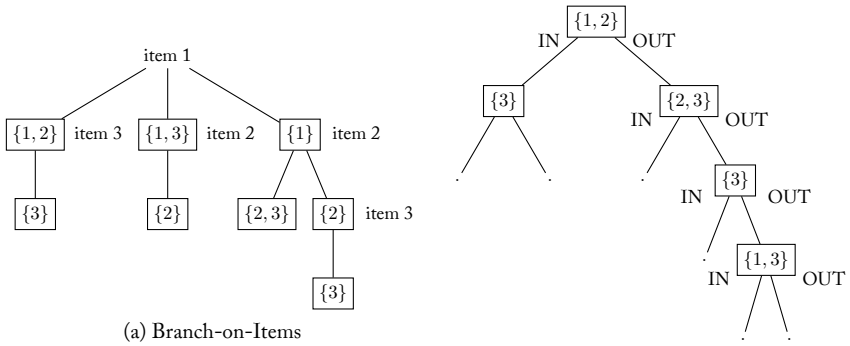
So far we have seen that releasing the first constraint will not help in finding a solution for the WDP in liquid bulk scheduling. Releasing the second constraint is only feasible when applying a B&B search strategy, which heavily depends on the quality of the bounding function. Releasing the third constraint - finding an optimal solution - is a common and promising approach for solving NP-hard problems, as it allows us to efficiently search the solution space in order to find possible high-valued solutions.

In order to apply efficient search techniques it is necessary to have a structured representation of the solution space. By representing the WDP as a search tree we can use existing tree search algorithms to effectively search for feasible and high-valued allocations of bids. Literature regarding the WDP suggest two main methods for representing the solution space as a tree: branch-on-items, and branch-on-bids.

Branch-on-items was the basis of the first generation of search algorithm for winner determination (Sandholm, 2002a; Fujishima et al., 1999). This approach is characterized by the following branching rule: for each node it is decided to which bid the item represented by the node should be assigned to. A simple branch on items search representation is shown in Figure 9.2a. In the worst case, the size of a branch on items tree would be polynomial in the number of bids, and exponential in the number of items (Sandholm, 2000). Since the items in liquid bulk scheduling represent the total number of infrastructure elements, which are typically several hundreds, this will lead to a very large search tree and thereby reduces performance.

The second generation search algorithms for winner determination shifted from using a branch-on-item approach to a branch-on-bids approach (Sandholm & Suri, 2003). These second generation algorithms are typically faster than their branch-on-items counterpart. In this branching strategy, the question for each node is: should this bid be accepted or rejected (see Figure 9.2b). This approach is typically faster than branch-on-items as it commits to the principle of least commitment described in (Russel & Norvig, 2009, p.346) stating that it is beneficial to only make choices which are currently important and leaving the rest to be dealt with later.

For application in liquid bulk scheduling an even more natural tree analogy than the previous two is possible which will be referred to as branch-on-parcels. In this strategy the main question at each node is: which parcel should be allocated now.



(a) Branch-on-Items

(b) Branch-on-Bids

Figure 9.2: WDP Branching Strategies with bids: {1,2} {2,3} {3} {1,3} (after Sandholm, 2000, p.341)

This is a more natural analogy to liquid bulk scheduling as it better represents the single-mindedness of bidders who submit multiple bids but are only interested in obtaining a single route.

A pruned and small example of such a search tree for application in liquid bulk scheduling is shown in Figure 9.3. In this tree, the nodes represent the latest (vessel, route, timeslot) tuple added to the allocation. A full allocation is obtained by moving up in the tree until the root node is reached.This tree shows a part of the search tree for a simple scenario where three vessels need to be scheduled with three routes each. At each level down into the tree an extra vessel will get added to the valuation and checked for feasibility. If the node is not feasible, a delayed version of the allocation is added as shown in figure.

There are several tree search algorithms available in the literature such as IDA*, backtracking, branch-and-bound, and simulated annealing which will be described in the upcoming paragraphs.

9.1.3.1 Informed search

Arkiletian tree search or tree traversal refers to systematic methods for visiting nodes in a tree structure. This section covers several informed search methods which can be used to find feasible solutions for the WDP in liquid bulk scheduling

Best first search

In a *Best-first Search* (BeFS) the algorithm will always select the sub-problem with the lowest bound. In practice, a BeFS often performs well since it tends to find a solution relatively quickly (Russel & Norvig, 2009). The worst-case time complexity for BeFS is $O(b^m)$ and since all nodes are retained in memory the space complexity

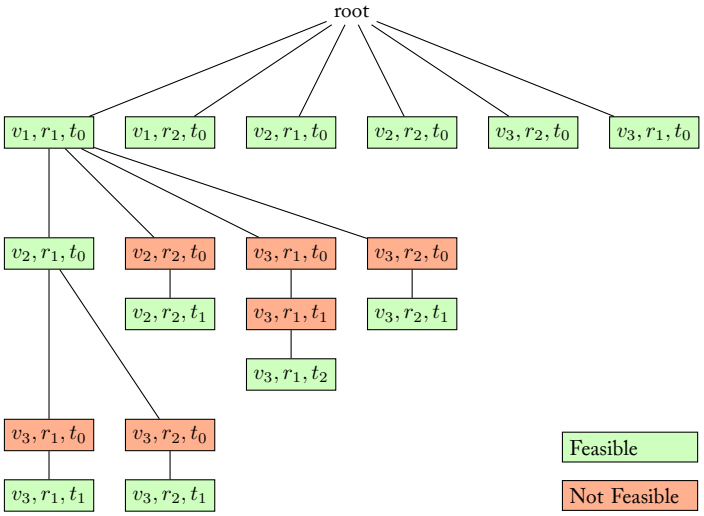


FIGURE 9.3: Search Tree representation of WDP where each node is the latest (vessel, route, timeslot) tuple added to the allocation

is the same. However, a good node value estimation function will drastically reduce the average space and time complexity.

BeFS is one of the most well-known heuristic functions and is applied often for its simplicity and its capability to generally find good enough results.

Breadth First Search

In *Breadth-first Search* (BFS) all nodes on level n are expanded before the nodes at level $n + 1$. Figure 9.4b shows the progress of a BFS on a simple tree. BFS is complete since it will keep traversing the tree until a solution is found. Furthermore, it is optimal in case the path cost is a non-decreasing function of the depth of the node. Unfortunately, this constraint does not hold in the winner determination problem as described in this chapter.

The worst-case time complexity of BFS is $O(b^d)$ where b is the branching factor on each node and d the depth of the tree. The space complexity is the same as all nodes are retained in memory. Given that in the case of liquid bulk scheduling the interesting solutions are at the bottom of the tree since these have allocated most parcels, BFS is not a suitable search strategy.

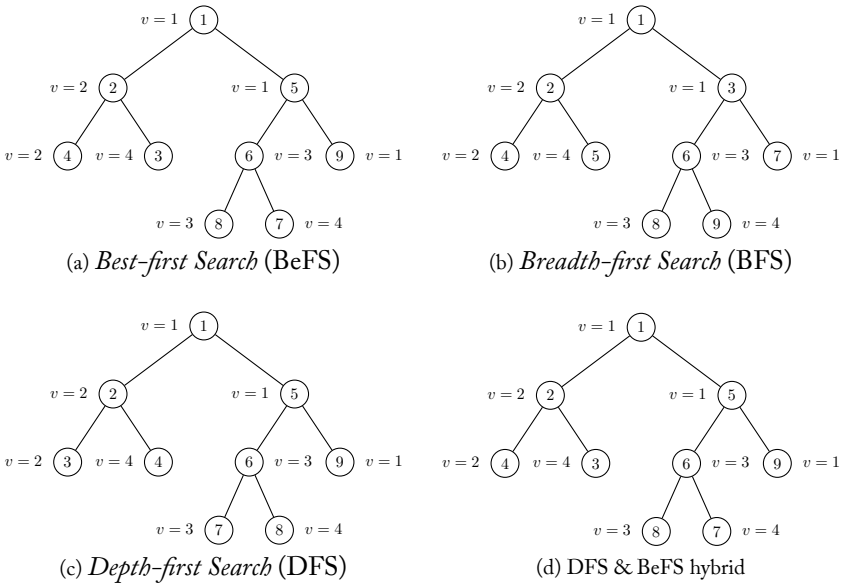
Depth first search

A DFS will always expand one of the nodes at the bottom level of the tree. The search will back up when it hits a dead end (a non-goal node which cannot be expanded further. Figure 9.4c shows the progress of a DFS on a simple tree. A major drawback of DFS is that it is prone to getting stuck down a wrong path and

thereby converging to a local optimum, without considering global optima. Since in our allocation problem all nodes on the lowest level of the tree are goal nodes, a DFS will simply return the first path down as the best solution, without considering their valuations.

A common mitigation strategy for this problem is to use a combination of DFS as the overall principle and BeFS when a choice has to be made between nodes at the same level in the tree. Figure 9.4d shows this type of search strategy.

In order to improve the quality of a solution found using a DFS and BeFS combination it is possible to let the algorithm look ahead to deeper nodes when choosing which child node to visit first. It then expands the node who's child node has the highest value. This method increases accuracy at the cost of increase runtime.



(a) *Best-first Search* (BeFS)

(b) *Breadth-first Search* (BFS)

(c) *Depth-first Search* (DFS)

(d) DFS & BeFS hybrid

Figure 9.4: Informed Search Methods

9.1.3.2 Backtracking

Backtracking is an algorithm for finding all or some solutions to constraint satisfaction problems, which incrementally builds up candidate solutions and abandons them as it determines the candidate cannot produce a valid solution. Backtracking traverses the tree in a DFS manner and keeps searching until a solution is found.

Backtracking is a complete strategy as it will keep searching until a feasible solution is found. Since backtracking is based on DFS, the same time and space complexity features hold which is $P(b^m)$ in worst case for time complexity and

$O(bm)$ for space complexity where b is the branching factor on each node and m the maximum depth of the tree.

Since backtracking is designed to find feasible, instead of optimal, solutions, it offers no guarantees for optimality of the found solution and does not attempt to maximize the optimality of the found solution. It is therefore less suitable for application in the winner determination problem.

9.1.3.3 IDDFS & IDA*

Iterative Deepening Depth-First Search (IDDFS) is a space search strategy in which a DFS is run a multitude of times, each time increasing the maximum depth at which the DFS can search. The order in which the nodes are accessed is effectively BFS. IDDFS mitigates some of the pitfalls of regular DFS by imposing a cut-off on the maximum depth of a path (Russel & Norvig, 2009).

*Iterative Deepening A** (IDA*) is a graph traversal algorithm which is suited to find the shortest path between a start node and a member of a set of goal nodes in a weighted graph. IDA* is based on iterative deepening search but concentrates on expanding most promising nodes instead of opening them in a BFS manner.

In iterative deepening search approaches, the nodes on the bottom level of the graph are expanded once, on the next to bottom level twice, and so on, which means that the time complexity is $O(b^d)$ and the space complexity $O(bd)$ where b is the branching factor on each node and d the search depth limitation. Iterative deepening searches give no optimality boundaries but are guaranteed to find a solution if it exists (Russel & Norvig, 2009).

In general, iterative deepening approaches are preferred when the search space is large and the depth of the solution in the tree is not known. However, in our scheduling problem, we know that preferred solutions are always at the bottom of the tree, as these have an allocation for all arrivals.

9.1.3.4 Simulated Annealing

Simulated Annealing (SA) is a stochastic optimization method based upon the physical process of annealing in metallurgy and was first described by Kirkpatrick, Gelatt, and Vecchi (1983). SA combines a hill-climbing search with a random walk in a way that the ratio between hill-climbing operations and random-walk operations varies from low to high as the search progresses. This combination of methods helps the heuristic to mitigate one of the major problems of using only hill-climbing, since hill-climbing often rapidly converges to a local optimum, but is often unable to find a global optimum.

Simulated annealing can be implemented as both a constructive search, in which the solution is built up step by step as introduced in the introduction of Section 9.1 (Queue-based SA) or as a search among alternative complete solutions (Node-based SA). An example of a neighbour-based search among alternative complete solutions is Kirkpatricks solution of the Traveling Salesman Problem (Kirkpatrick et al., 1983).

In this algorithm a random neighbour solution s' is generated in each iteration. The algorithm then moves to this neighbour either if its cost is lower than the current node's value, or when the following equation evaluates to true.

$$e^{\frac{-(s'-s)}{T}} < Rand(0,1) \tag{9.4}$$

In a queue-based approach, the simulated annealing makes a random selection from a queue of sorted nodes. This random selection is usually taken from an exponential distribution so that the better nodes are more likely to be chosen. The temperature of the annealing is used to control the mean of the distribution. The effects of this selection method are shown in Figure 9.5 which shows a chart with the selection chance for a node from the queue $\{1000, 900, ..., 100\}$ for varying temperatures. This approach is for example described in (Collins, 2002).

Both variations of SA have their merits and it is unclear based on literature review alone which will perform better in this case. Therefore, both approaches will be implemented and tested.

SA is due to its mix between random walk and hill climbing not a complete strategy. Depending on the chosen temperature and temperature profile, it could spend too much time random walking and thereby missing the solution. However, in practice this is easily mitigated by correctly setting the initial temperature and its decay function. The average time and space complexity grow polynomial with the input size (Sasaki & Haje, 1988). SA gives no guarantees on finding the optimal solution, however, due to the combination of random walk and hill climbing it is less prone to get stuck in local optima than for example a heuristic search.

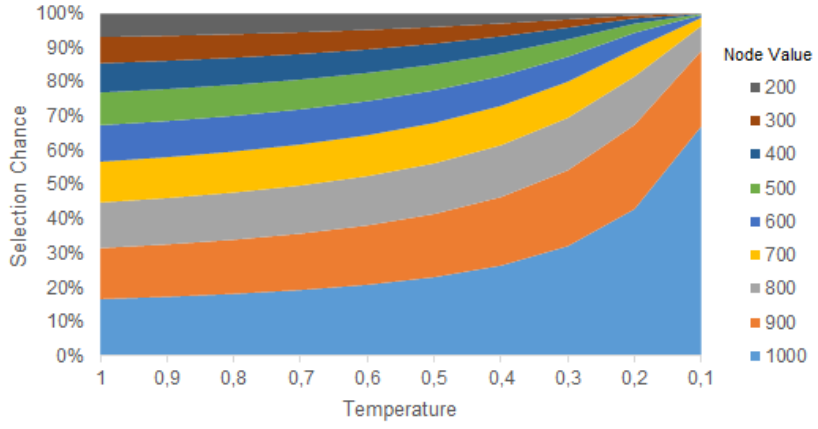


Figure 9.5: Chance for a node with a certain value to be selected by SA for varying temperatures (assuming a queue with $v \in \{1000, 900, ..., 100\}$)

9.1.3.5 Performance Tuning

Stochastic search algorithms such as SA can often be tweaked in a number of different ways. Although the literature regarding SA identifies many possible tuning parameters, it unfortunately offers little guidance on how to do the tuning. This section explores several significant parameters that can be altered in this implementation based on work by Collins (2002), van Laarhoven and Aarts (1987), Tan (2012).

Temperature Profile
The temperature of an iteration determines the ratio between random behaviour and hill-climbing. Choosing the correct starting temperature and rate of decay is an important parameter when tuning SA algorithms. The implementation used in this thesis allows for a maximum temperature of 1.0 and a minimum of 0.0. A common problem in SA implementations is that the starting temperature is chosen too high which causes too much perturbation for the algorithm to converge to optimal values (Ledesma, Avina, & Sanchez, 2008).

Apart from the initial temperature problem it is also essential to choose the correct temperature decay function. Two typical temperature decay functions exist in the literature: linear and exponential decay. A linear temperature decay function will spend equal time at each temperature in the process. In an exponential temperature decay scenario, the algorithm spends only a little time at high temperatures, and spends longer on each temperature as it decreases. Therefore, linear cooling should generally be preferred in the case of several optimal peaks close by while exponential cooling should be preferred if that is not the case.

Apart from linear and exponential decay functions it has been discussed in the literature that a temperature decay function which is inversely proportional to time will guarantee a convergence to a global optimum. In practice however, this type of decay function is typically too slow (R. Reed & Marks, 1999).

In order to determine the temperature profile for this application several experiments have been set up with varying problem sizes, starting temperature and temperature decay. The results from this experiments are further explained in Chapter 11.

Stopping Criteria
SA does not impose a natural stopping criteria, and will thus keep searching until the entire search space is exhausted. Since exhaustively searching the solution space is not feasible due to its large nature, external stopping criteria are required. Literature suggest three main stopping criteria: Time limits, Acceptance probability bounds, and Identical Configuration Score Bounds (Otten & Ginneken, 1988).

In a time limit or iteration limit based stopping criteria the SA will run for a predetermined amount of iterations or time period. When this limit is reached, the current best node is given as a final solution. A main advantage of this method is that it imposes a hard limit on the amount of time the algorithm runs. The main

disadvantage of this method is that it is possible for the algorithm to stop while it can still easily generate better solutions than the current best node. This type of stopping criteria is typically used in cases where (knowing) the runtime is much more important than the quality of the result.

The second strategy is based on the probability that a move from a node to another node with the lowest score (White, 1984). This method typically requires extensive knowledge on the problem type and instance in order to set reasonable bounds.

The method which will be employed for this implementation is setting bounds on identical configuration scores. This means that the run is stopped when a better solution is not found for a predetermined amount of iterations. This boundary can be implemented as a fixed number or as a ratio. In a fixed boundary the patience of the run is updated whenever a new solution is found as such: $patience = i_{now} + n$ where n is a preset value. When using a patience ratio the patience is updated like: $patience = i_{now} * ratio$ where the ratio is a preset value. The SA run then ends whenever the current iteration number is higher than the patience. Typically the ratio based mechanism scales better on varying problem sizes (Collins, 2002).

Queue Length
One of the SA implementations discussed in this thesis uses a queue-based approach. Each iteration of the SA starts by selecting an entry from this queue. In order to ensure performance, the total length of this queue is limited. A larger queue means more memory usage and typically slower node selection. However, if the queue becomes too short, it is possible for it to become filled with mostly similar entries.

Restarts
Since SA is a stochastic search method it is not guaranteed (and typically not the case) that two consecutive runs provide the same answer. This can be used to our advantage since literature in general suggests that it is more useful to perform several short searches rather than performing a single large search (Alfonzetti, Dilettoso, & Salerno, 2006; Mendivil, Shonkwiler, & Spruill, 2001). Another major advantage of applying restarts is that it is an embarrassingly parallel problem[3], which can be used to increase performance.

Problem Size Scaling
This section has discussed several key tuning parameters in SA algorithms. When considering these parameters in the domain of liquid bulk scheduling it should be noted that there is a large variation in problem size and complexity. To accommodate for this variation in problem size it will be necessary to adjust the actual tuning parameters based on the input size of the problem by using a scale-factor. This scale-factor is calculated at the beginning of the run and is based on the number of parcels

[3] A problem for which little or no effort is required to separate into parallel tasks with no dependencies between those tasks

which are to be allocated, as well as an estimation on the number of overlapping arrival windows between those parcels. A larger problem size will generally result in a slower decreasing temperature profile, a higher stopping condition ratio, and a larger queue length.

9.2 Allocation Evaluation

All search heuristics presented in this chapter depend on an accurate valuation of each decision node in the tree. The valuation of individual nodes depend on three main attributes:

1. The value of allocated bids in the node;
2. The completeness of the node;
3. The value of terminal preference attributes in the node.

While the dependence on the bid valuations is straightforward, the implementation of both others are not. The upcoming sections describe for attribute 2 and 3 the encountered problems as well as possible solutions.

9.2.1 Bid valuation on incomplete nodes

Due to the specific structure of the search problem an interesting feature emerges when designing a search heuristic regarding the completeness of a node. For simplicity, this section will assume searching with a BeFS strategy and only considers the valuation of a node based on the value of the allocated bids, but the conclusions are equally valid for other heuristics such as the queue-based SA. Other approaches such as the node-based SA or a DFS do not suffer from this issue as they only compare nodes on the same level.

This feature is easily discovered when considering the search tree as depicted in Figure 9.3. In this scenario, we consider the value of a node to be the sum of the allocated bids in that node. Figure 9.6 shows a parallel coordinates plot of the node valuation of a sample of $2,000,000$ possible allocations in an allocation problem with 4 vessels with their values at each level in the tree. It is clearly visible in this case that the search will be strongly similar to a DFS, as the valuation of nodes can only increase when the level of the node increases. Due to this DFS-like behaviour, the heuristic is likely to end up in a local maximum without properly considering other possible nodes.

An easy and intuitive method to counteract this behaviour is by taking the average of bid values rather than the sum. However, as the problem size increases, it is expected that bids lower in the tree have lower values due to there being more conflicts leading to less preferred routes to be planned (Figure 9.7). In this case, the heuristic will have a strong BFS-like behaviour, leading to very long running times.

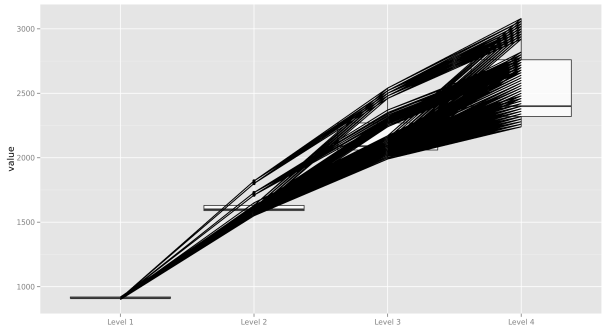


Figure 9.6: Node valuation per tree level where $v = \sum_{b \subseteq B} b_i$

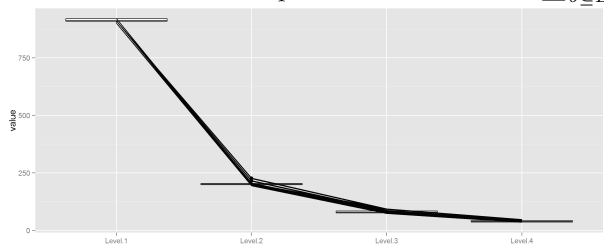


Figure 9.7: Node valuation per tree level where $v = \frac{1}{|B|} \sum_{b \subseteq B} b_i$

Scaling solutions

It is clear that the optimal valuation of incomplete nodes is somewhere in the middle between the sum of bids and the average of bids. In order to compare the nodes on different levels on the tree it is needed to scale the attribute values to a common scale (Leskovec, Rajaraman, & Ullman, 2014). An often used method for feature scaling is Z-score standardization. A Z-score standardization will rescale the values of an attribute so that they have the properties of a standard normal distribution with $\mu = 0$ and $\sigma = 1$ (Equation 9.5).

$$z = \frac{x - \mu}{\sigma} \tag{9.5}$$

An alternative to Z-score normalization is the so-called Min-Max scaling. In Min-Max scaling the data is reduced to a fixed range, usually from 0 to 1. Min-Max scaling differs from the Z-score normalization in that it ends up with smaller standard deviations, thereby suppressing the effects of outliers (Equation 9.6).

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{9.6}$$

In order to determine the optimal scaling method several brute-force searches were done on a relatively small problem with 4 parcels and ± 40 routes per parcel. To keep the dataset manageable a stratified sample set was extracted from these nodes

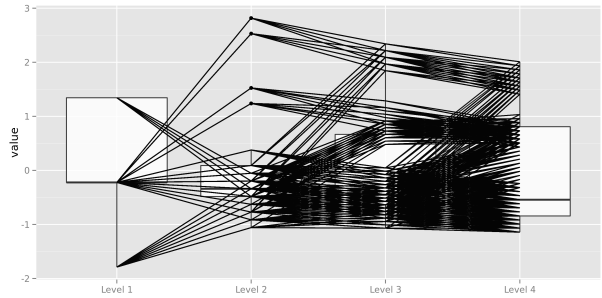


Figure 9.8: Feasible Z-score scaled observations

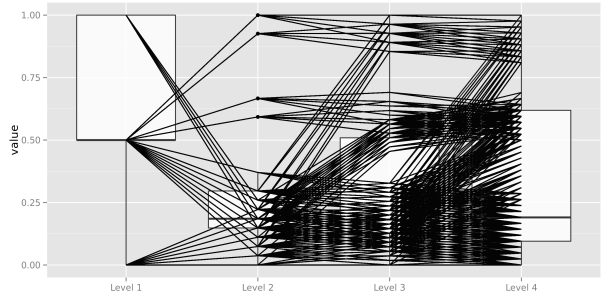


Figure 9.9: Feasible Min-Max scaled observations

while preserving the overall distribution of the data (Leskovec et al., 2014). The unscaled node valuations in this sample set are shown in Figure 9.6. Figures 9.8 and 9.9 show the scaled versions of these observations both for all nodes and filtered on only feasible nodes. These figures clearly show the differences between Z-Score and Min-Max scaling. All nodes in the Min-Max scaling approach are between 0 and 1, and the effect of the outliers on level 2 is much less severe than in the Z-Score scaling.

It was found that the Z-score value of a random encountered node correlated stronger (coef: 0.42) with the final result value than the Min-Max score of that same node (coef: 0.33) at $p < 0.05$.

Scaling implementation

It is clear that introducing scaling the values of nodes drastically improve the comparability between layers in the solution space tree and that using Z-Score scaling seems to be the best predictor of the final allocation value. Unfortunately, the implementation of such a scaling is not straightforward.

In order to ensure efficiency of the algorithm it is not feasible to recalculate the standard deviation and mean over the entire set every time a node gets added. While computing a cumulative moving mean is trivial (Equation 9.7) implementing

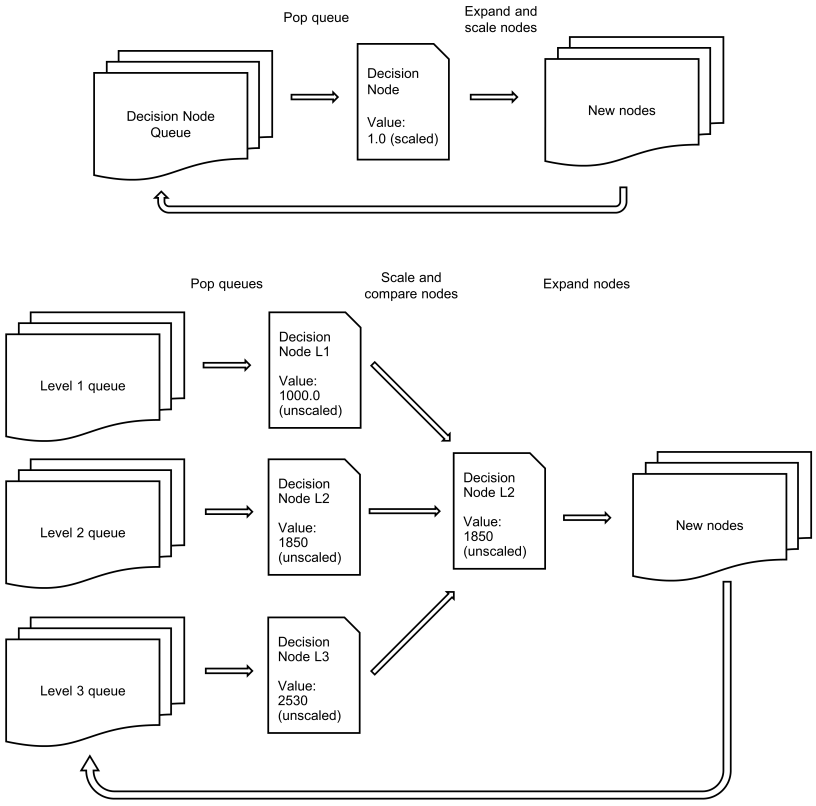


Figure 9.10: Scaling upon node insertion (top) vs. Scaling at node selection (bottom)

a numerically stable moving standard deviation is not. Knuth (2011) proposes a numerically stable algorithm to compute both the mean and standard deviation which has been shown to perform well (Ling, 1974).

$$\mu_{n+1} = \frac{n * \mu_n + x_{n+1}}{n+1} \tag{9.7}$$

A second challenge arises due to the incremental build-up nature of this solution. When opening the first nodes on a layer,it is not yet possible to scale these node valuations due to there not being enough data to accurately predict a mean and standard deviation for that level of the tree, causing the node valuation to be very unreliable and inaccurate. Two solution strategies to deal with this issue were explored. First, the use of historical data was tested by storing the means and standard deviations of levels which were explored in earlier runs of the algorithm.

This results in much more stable values which makes the algorithm more predictable. However, it turns out that the means and standard deviations can vary largely across different runs due to differences in congestion at the terminal and was therefore not deemed suitable.

The second solution which was explored delayed the scaling of the nodes from the moment it is added to the graph to the moment it is up for expansion. Figure 9.10 shows the differences between these two scaling moments. In the top version, all nodes are entered into a queue based upon their scaled value. When a node is up for expansion, its child nodes are visited, valued, and scaled and inserted at their correct place in the original queue. This version will produce very unstable and inaccurate results when opening the first nodes on a new level. The bottom version solves this issue by delaying the scaling. In this version, all nodes are entered into different queues based on their level in the tree. The top node from each of these queues is popped, scaled and compared to find the best node at that moment. The main advantage of this method is that it ensures that nodes on a level are always scaled correctly with respect to each other. A disadvantage of this method that it adds more scaling and comparison operations, thereby decreasing the algorithms' performance. However, this effect is expected to be minimal.

9.2.2 Terminal preference valuation

Determining the value of a node with regards to the terminal preferences will be done using a DCM method as introduced in Section 5.4.3. Both a linear additive (R)UM as well as a (R)RM are implemented and compared in Chapter 11 of this thesis.

The attributes which were found relevant to consider when determining the valuation of an allocation from the perspective of the terminal which were introduced in 3.1 are:

1. High terminal utilization;
2. Low impact on surroundings;
3. Low pressure on operations.

The first step in order to evaluate these attributes is operationalizing them. This refers to the process of making initially unmeasurable, high-level, variables into a set of measurable sub-components, and is visualized in Figure 9.11. There are two main methods which can be used to determine the value from the perspective of the terminal regarding these attributes: *Revealed Preference* (RP) methods and *Stated Preference* (SP) methods.

In a RP research the choices people actually made are examined and generalized into a choice model. RP methods are especially suitable when there are no new types of alternatives introduced and preferences do not shift heavily over time. Generally, RP methods are preferred over SP methods since they are less likely to be subject to

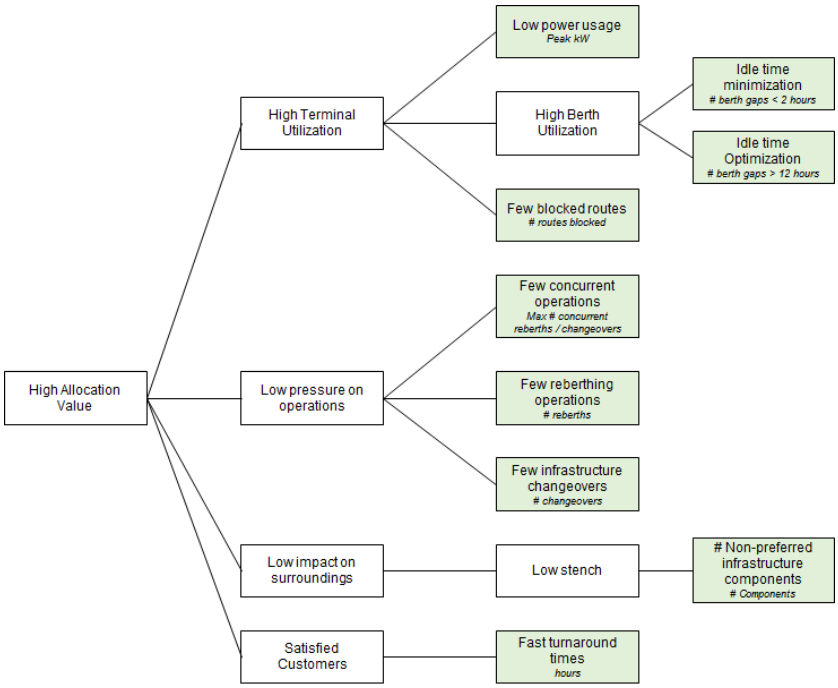


Figure 9.11: Operationalization of terminal preferences

response-bias as present in surveys. Unfortunately, it was not possible to access RP data on a high enough level of detail which made initial RP research infeasible.

In an SP experiment the target group is asked directly about their preferences through surveys or interviews. Designing such a survey is often referred to as experimental design. When drafting an experimental design for SP research, several topics need to be considered. These are:

- How many factors are relevant in the design?
- How many attribute levels should be considered?
- Are there relevant interactions between factors?
- What is the sample size?

Table 9.1 shows the attributes as well as their possible ranges, and whether they are expected to be a linear relation. It is expected that there is a relation between the "planning horizon" attribute and most other attributes since when planning short-term operations the allocation value becomes less important than actually being able to make the allocation.

Table 9.1: Terminal preference attributes with working ranges and assumed weights

Attribute	Range	Linear	Weight
Power Usage	0 - 7000 kVA	✗	-0.0001
Non-min. or optimal idle times	0 - $parcels + 2$	✓	-0.5
Blocked Routes	0 - $10 * parcels$	✓	-0.5
Concurrent Operations	0 - $parcels$	✗	-1
Re-berthing Operations	0 - $parcels$	✓	-1
Infrastructure Changeovers	0 - $parcels + 2$	✓	-0.02
Non-preferred infra. components	0 - $15 * parcels$	✓	-1
Plan horizon	$0 - 4$ weeks	✓	
Customer Satisfaction	$0 - parcels * 1000$	✓	0.002

Due to the amount of attributes and attribute levels it is not feasible to use a full-factorial survey design as this would require each respondent to rate over 50.000 alternatives. A common-used survey design is an orthogonal fractional factorial design, which greatly limits the number of choice sets. A main disadvantage of this method is that it is not suitable to estimate interaction effects such as identified with the plan horizon parameter. In order to still identify interaction effects a foldover design can be added, which "mirrors" the original fractional factorial design. A foldover design has double the number of choice sets that a normal factorial design has, but this is still much more feasible than a full-factorial design.

The survey is expected to be conducted among ± 5 planners and each survey can consist of about ± 20 questions. Unfortunately this limited sample size is not enough in practice to find rigid preference factors which results in SP not being feasible for determining preference data within the scope of this research.

Since initial RP data is not available, and SP methods not being feasible due to the limited sample size the only option which remains is using educated estimates for attribute weights. These estimates are drafted in collaboration with experts from Systems Navigator in the liquid bulk domain and can be found in Table 9.1. Since the planning system is part of an operational planning support system, it offers major possibilities in letting the system learn from the planners' actual choices. This could be implemented as a ongoing RP research which gets updated each time a planner chooses a new final allocation. Although this implementation is relatively simple, it still requires much time for it to have collected enough choices from planners to have a significant learning effect which means it is considered out of scope for this thesis.

9.2.3 Regret model in scheduling

As discussed in the choice model section of Chapter 5 both a *(Random) Utility Maximization* ((R)UM) and a *(Random) Regret Minimization* ((R)RM) approach can be applied as a method for weighing terminal preferences in the allocation process. While the implementation of (R)UM models is fairly straightforward there are several difficulties with implementing its regret-based counterpart, which originate from its exhaustive comparison of alternatives. This section will briefly describe these problems as well as possible solution directions.

The classical regret model as introduced in (Chorus, 2010) was the first attempt of implementing a generic approach for regret-based discrete choice behaviour, and is based on the notion that the improvement of one attribute of an alternative cannot necessarily counteract an equally large decline in another attribute. The regret of an alternative can be calculated using Equation 9.8 where R_i is the regret of the alternative which is observed, A the set of alternatives, M the set of (relevant) alternative attributes and β_m the weight for attribute $m \in M$.

$$R_i = \sum_{j \in A | j \neq i} \sum_{m \in M} \ln(1 + e^{(\beta_m (x_{jm} - x_{im}))}) \tag{9.8}$$

The first problem encountered is an obvious one and has to do with the computational complexity of calculating a regret-based choice model. In a (R)UM model the value of an alternative is calculated simply by taking a weighted sum of the attribute values of that alternative. This means that computing the values for a set of alternatives can be done in O(n) operations where n is the number of alternatives. A (R)RM model on the other hand, compares all attribute values for an alternative with all other alternatives in the set, making the run time of the calculation O(n^2) where n is the number of alternatives (Chorus, 2012c).

This problem is amplified by a feature of this scheduling system which is similar to a problem encountered in exploring potential solutions for the incomplete node valuation problem described earlier in this chapter. Since the search space gets explored gradually not all alternative allocations are known up front. This means that a valuation of all previously found nodes should be re-evaluated upon the entry of a new alternative to the solution set, making the total run time of the computation O(n^3). As we have already seen the costs for computing the value of an alternative becomes prohibitively high as the size of the alternative set increases. Since this characteristic only amplifies this problem it will be completely infeasible to navigate the search tree using simply (R)RM.

An alternative to the regular (R)RM model called *Pure Random Regret Minimization* (P-RRM) is proposed in (Cranenburgh, Angelo, & Chorus, 2015), which differs from the regular model in that it shows more pure regret-minimizing behaviour. A side-effect of this model is that it offers the possibility to pre-compute a vector on the alternative space which can be used for the evaluation of singular alternatives in constant time instead of being dependent on the total number of

alternatives (Equation 9.9 & 9.10). P-RRM cannot be directly applied to our case as the set of alternatives differs for each scheduling job, essentially removing the benefits of pre-computation.

$$R_i^{P-RRM} = \sum_m \beta_m x_{im}^{P-RRM} \tag{9.9}$$

$$x_{im}^{P-RRM} = \begin{cases} \sum_{j \neq i} \max(0, x_{jm} - x_{im}) & \text{if } \beta_m > 0 \\ \sum_{j \neq i} \min(0, x_{jm} - x_{im}) & \text{if } \beta_m < 0 \end{cases} \tag{9.10}$$

This size-related problem is most prominent with solution strategies like BeFS and the queue-based SA, as these strategies typically evaluate a large number of nodes on each iteration. Other strategies like DFS and the node-based SA suffer much less from this problem as the number of nodes evaluated is typically much smaller. A simple method to solve the computational problems for the first type of strategies is to apply a hybrid of (R)UM and (R)RM in which the navigation through the search space is done using a (R)UM and all solution nodes are kept in memory. These final solution nodes which represent a full allocation are then compared using (R)RM (Figure 9.12). By only applying (R)RM to the set of full allocation nodes the size of the alternative set becomes manageable for computing. A clear advantage of this approach is that it is simple and fast. The main disadvantage is that since in this approach the WDP solver would use a (R)UM but the final weighing is done using (R)RM it is possible that the solver does not explore nodes in the solution space which are lowly valued by the (R)UM, but would be highly valued by the (R)RM, thereby decreasing the overall quality of the solution. This effect is partially mitigated by the fact that a good solution using regret is also a good solution using utility, albeit with some added nuance.

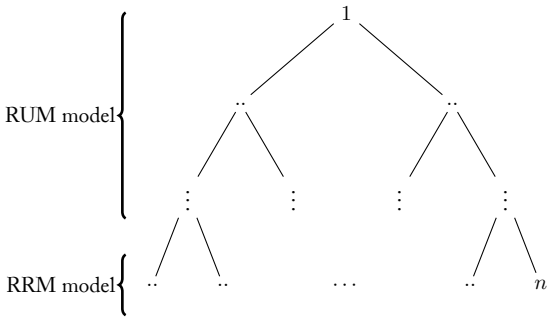


Figure 9.12: Hybrid (R)UM & (R)RM model

A second topic which needs to be considered when assessing the applicability of (R)RM in scheduling problems is typical searches in scheduling are non-exhaustive. In a non exhaustive search, not all alternatives will be observed and the observed alternatives can differ for subsequent runs. In the case of for example SA the observed solution space can differ greatly for the same scheduling job due to its stochastic nature which can cause alternative C to be of high quality the first run, and of low quality in the second. It is currently unclear how large this effect will be but it is expected to be relatively small due to the large number of nodes which are explored although further experimentation is needed to support this claim.

9.3 Feasibility detection

Not all nodes present in the search tree will yield feasible allocations. This section describes in detail what constitutes a feasible allocation and in which cases it should be marked as infeasible. It then goes on describing how the feasibility check is implemented. Finally, some attention will be given to the concept of probabilistic feasibility.

The following constraints are implemented into the feasibility check of the planner agent:

- No regular infrastructure can be in use by two tasks simultaneously;
- The total vapour return capacity cannot exceed the maximum capacity of the *Vapour Return System* (VRS);
- When a vessel has two tasks scheduled at different berths, there needs to be at least 1 hours of reberthing time;
- When a berth is scheduled for two different vessels there needs to be at least 1 hour of reberthing time;
- A vessel cannot be allocated to simultaneously offload two parcels;
- When a vessel is approaching or departing from a berth, there can be no vessel movements at adjacent berths;
- If an infrastructure element is last used for black product and is subsequently scheduled for white product, an extra half hour is needed for infrastructure changeover.

Currently these constraints are hard-coded into the feasibility check. An obvious way to facilitate easier reuse of the system is to allow the users to more flexibly define scheduling constraints through for example a *Domain Specific Language* (DSL). Several DSL were evaluated for their applicability in the liquid bulk scheduling domain including (Gartner, Musliu, Schafhauser, & Slany, 2011; Sobernig, Strembeck, & Beck, 2013) but none were found to be capable of dealing with the JMPT problems as found in liquid bulk scheduling.

The feasibility check uses an interval tree which is based on a red-black tree to efficiently find overlapping time periods in O($\log n$) (Cormen, Leiserson, Rivest, & Stein, 2009). The total complexity of the feasibility check is therefore O($nm \log n$) where n is the number of routes and m the number of infrastructure elements in those routes. To speed up the solver, the feasibility checker is able to give a suggested minimum delay which is a lower bound on the number of hours the new allocation should be delayed to become feasible. This allows us to skip certainly infeasible nodes when exploring the solution space.

For this thesis all constraints and processing times were assumed to be fixed, based on averages found by data analysis at several liquid bulk terminals. However, as Appendix B shows, large variations in the operations time were found when examining the available data. This can be incorporated in the system by instead of representing the feasibility of an allocation by a boolean, representing it as a chance the allocation is feasible. Such a allocation feasibility chance can be easily calculated using for example a Monte-Carlo simulation at the expense of some run speed. This feasibility percentage is then an extra attribute defining the allocation which can be incorporated in the decision modelling as described in the previous section.

9.4 Conclusion

This chapter described several design dimensions when considering the implementation of an auctioneer agent in liquid bulk scheduling. The first section explored three main strategies for solving the generally intractable WDP in liquid bulk combinatorial auctions. In general one of three desired features need to be given up in order to deal with intractable problems:

1. Solve arbitrary instances of the problem;
2. Solve the problem in polynomial time;
3. Solve the problem to optimality.

It was shown that there are no specific instances of the WDP which can be solved in polynomially bounded time which were applicable for the case presented in this thesis. Therefore, releasing constraint 1 provided us with no clear benefits for solving the WDP. By representing the problem efficiently as a search tree there are several possibilities with releasing constraint 2 and 3. *Branch and Bound* (B&B), informed search as well as *Simulated Annealing* (SA) algorithms seem promising methods for efficiently finding good allocations although several tuning questions remain. Both a B&B, a SA, as well as a DFS informed search are implemented and experimental findings on these three algorithm types will be discussed in the upcoming chapter. Furthermore, a FCFS heuristic is implemented mainly to serve as a baseline for comparing the other methods.

The second section went into detail on the valuation of nodes in the search space. It was found that the valuation of a node depends on three properties:

1. The value of allocated bids in the node;
2. The completeness of the node;
3. The value of terminal preference attributes in the node.

While the first of these properties is straightforward, the other two are not. The first part of this section went into detail on evaluating incomplete nodes. It was found that when evaluating nodes on different levels in the search tree a scaling was needed to efficiently compare nodes. Both a Z-Score and a Min-Max scaling were tested and it was found that the Z-Score scaling of a node correlated higher with the final allocation value at that node.

The second part of the section went into detail on how to incorporate terminal preferences in the allocation of a node. An operationalization into measurable attributes was made based on the terminals preferences as described in Chapter 3. Both an *Stated Preference* (SP) and an *Revealed Preference* (RP) method for measuring terminal preferences were tried but both were found infeasible to carry out within the scope of this thesis. For experimentation purposes, attribute weights were drafted in collaboration with experts from Systems Navigator which, in the future can be expanded using real-life data from the terminals.

The third part of the section explored the possibilities for implementing a regret-based choice model in agent-based scheduling systems. The two main topics which need consideration when applying regret models in such a context are the computational complexity of the valuation function and the non-exhaustiveness of the search.

The computational complexity of the model is mostly prohibitive in cases where a large number of nodes is evaluated on each iteration of the search algorithm. This is the case with for example a *Best-first Search* (BeFS), and a queue-based SA. For these cases a hybrid (R)UM & (R)RM model can be applied which uses a (R)UM to navigate the tree and a (R)RM to do the final value calculation.

The second topic which needs to be considered is that search strategies are typically not exhaustive, meaning not all alternatives are found and evaluated. Theoretically, this can cause differences in the alternative value found in the run compared to when all nodes would be known. The expected impact of this feature is low, as the number of found alternatives is generally still very high, although further research would be required to support this claim.

The final section of this chapter described the method used for verifying whether an allocation is feasible. These checks are fairly straightforward and are currently hard-coded in the system. To improve re-usability, it could be beneficial to use a *Domain Specific Language* (DSL) for describing the constraints, but no available DSL offered the required flexibility.

Part III

Validation and Evaluation

System Testing 10

This thesis began with a top-down approach to identify the relevant stakeholders in the liquid bulk domain, the processes at liquid bulk terminals, and the ecosystem in which the system is supposed to operate.

Part II used this information to draft several detailed designs which can be applied in liquid bulk scheduling. Several knowledge gaps still remain on the effectiveness and efficiency of the discussed solutions, such as the differences in application between *(Random) Utility Maximization* ((R)UM) and *(Random) Regret Minimization* ((R)RM) models for eliciting terminal preferences, and the qualities and speed of *Branch and Bound* (B&B), *Simulated Annealing* (SA), and heuristic search solvers.

Before employing the *Multiagent System* (MAS) for scheduling support at liquid bulk terminals it is necessary to ensure that the system is verified and valid. In this phase the implementation is verified with the conceptual model as designed in Part II of this thesis and validated with the requirements as found in Part I. For this chapter we will borrow heavily from simulation verification and validation theory.

10.1 Verification

The main goal of the verification step is to check whether the conceptual model was correctly translated to an implementation. In order words, "Did we build the thing right?" (Van Dam, Nikolic, & Lukszo, 2013, p.98). Based on literature surrounding Agent-based modelling verification three main categories of verification tests have been identified which will be described in the upcoming sections

- **Single-agent level** - Verifies behaviour of individual agents;
- **Minimal interaction level** - Verifies individual interactions between agents;
- **Multi-agent level** - Verifies the overall functioning of the system.

10.1.1 Single-agent level

The model contains three types of agents: The Vessel agent, the Route Finder agent, and the Planner agent. Each of these will be verified separately.

The Vessel agent represents a vessel which is owned or operated by the terminals' customers. The main function of the vessel agent is to calculate bid values for possible route options and submit these to the auctioneer. This functionality is tested by comparing a set of bids generated by the vessel agent by a pre-computed set of bids for the same route options in which no differences were found.

The Routefinder agent represents the physical infrastructure available at the terminal. Its main goal is to find possible routes and timeslots for arrivals to perform operations. Route options were checked manually for a small example terminal and no infeasible routes were found.

The Planner agent represents the terminal and planner stakeholders. Its main function is using a set of bids to find feasible allocations. For the Planner agent the discrete choice models, as well as the constraints were verified by comparing generated datasets.

Apart from these specific test cases mentioned, the behaviour at the single-agent level is verified by a suite of unit-tests and FsCheck, an automatic test generator for F# based on Haskell's QuickCheck (Claessen & Hughes, 2000).

10.1.2 Minimal Interaction level

After verification is complete on the single-agent level we can proceed to testing the minimal interactions between agents. The minimal interaction level refers to simple 2-way communication between two agents. Testing on the minimal interaction level is mostly a robustness check which ensures that failures are handled gracefully. Several possible failures are slow start-ups of agents, agent failures, slow responding agents and deadlocks. In order to verify the system on the minimal interaction level, a number of test-cases were set up featuring randomly failing agents, agents starting with long delays, and agents which are randomly set to sleep indefinitely. In these

tests, all failures were handled gracefully either by restarting the failing agent, or restarting the entire scheduling job.

10.1.3 Multi-agent level

Verification on the multi-agent level concerns itself with the overall observed behaviour of the system. Common verification tests for this level include sensitivity analysis and extreme conditions testing (Carson, 2002; Sargent, 2011).

With a sensitivity analysis, small variations in the systems' input parameters are made and their effect on the systems' outcomes are measured. These effects are then compared with expected results. Extreme conditions testing stress tests the model under large parameter changes.

Sensitivity analysis was performed on four main types of solution strategies discussed in Part II of this thesis: *First-come, first-served* (FCFS), Node-based SA, Queue-based SA, and *Depth-first Search* (DFS) with lookahead. During this validation step it was found that the bounding function of the B&B was too weak to find results efficiently, making it infeasible to consider in this validation test. This will be further described in the upcoming chapter.

The findings from the sensitivity analysis and extreme conditions testing are visualized in Tables 10.1 to 10.4. The first column shows the parameters which are subject to the validation test. These parameters are chosen as they are independent of the chosen solution method and have a clear impact on the system. The second column shows the *Key Performance Indicators* (KPIs) on which the outcomes will be measured. The following five columns show the outcomes on the KPIs for different alterations of the parameters. The base-case considered for the validation tests is a case consisting of 10 arrivals, ± 2 parcels per vessel, 20 reservations, and a planning horizon of 10 days. To increase the comparability of the results, the input arrival and reservations set have been kept constant over the different solution strategies. The reported values are the means over 15 replications, together with the Standard Error of the Mean.

From this sensitivity and extreme conditions test we can draw the following conclusions: The run time of the system is highly dependent on the number of arrivals which are to be scheduled. The total utility of a run is also highly dependent on the number of arrivals, since adding more arrivals means that a higher total bid value can be obtained. The number of reservations only tend to influence the system minimally, although a slightly decreasing utility value is observed when increasing the number of reservations. Another main factor in the quality of the results is the planning horizon. This can be explained by the system being able to allocate vessels better when they arrive over a larger period of time. The main weakness of this sensitivity analysis lies in the limited number of performed replications, as can be seen from the relatively high Standard Error. This causes these findings to be less generalizable than desired.

All tests on the multi-agent level indicate that the system behaves as expected and is a valid implementation of the proposed design.

Table 10.1: Sensitivity analyses & Extreme conditions testing FCFS

Parameter / KPIs	-75%	-50%	0%	+100%	+400%
Arrivals:					
Runtime *(MM:SS)*	00:00 ±00:00	00:01 ±00:00	00:01 ±00:00	00:02 ±00:00	00:05 ±00:00
(R)UM value	5,641 ±293	8,000 ±390	9,828 ±760	8,545 ±1,191	10,531 ±2,884
Unplanned vessels	0 ±0	0.6 ±0.5	2.1 ±1.2	2.9 ±2.1	3.6 ±2.6
Reservations					
Runtime *(MM:SS)*	00:01 ±00:00	00:01 ±00:00	00:01 ±00:00	00:01 ±00:00	00:01 ±00:00
(R)UM value	8,484 ±823	8,544 ±740	8,275 ±645	9,078 ±721	7,758 ±857
Unplanned vessels	0 ±1.22	0.8 ±1.39	1.2 ±0.38	1.4 ±1.43	1.6 ±1.14
Parcel Size:					
Runtime *(MM:SS)*	00:01 ±00:00	00:01 ±00:00	00:01 ±00:00	00:01 ±00:00	00:01 ±00:00
(RUM) value	9,013 ±799	9,784 ±804	9,352 ±774	6,965 ±593	6,131 ±511
Unplanned vessels	0 ±0	1.1 ±0.5	1.6 ±1.45	2.1 ±1.46	2.8 ±1.34
Planning Horizon:					
Runtime *(MM:SS)*	00:01 ±00:00	00:01 ±00:00	00:01 ±00:00	00:01 ±00:00	00:01 ±00:00
(RUM) value	7,854 ±534	8,509 ±456	9,139 ±661	9,764 ±741	10,490 ±762
Unplanned vessels	2.6 ±1.41	1.8 ±1.19	1.5 ±0.86	0.9 ±0.43	0 ±0

Table 10.2: Sensitivity analysis & Extreme conditions testing Queue-based SA

Parameter / KPIs	-75%	-50%	0%	+100%	+400%
Arrivals:					
Runtime *(MM:SS)*	01:34 ±01:02	11:18 ±00:46	11:27 ±00:47	13:41 ±01:12	15:00 ±03:48
(R)UM value	4,617 ±403	4,606 ±562	9,387 ±1,232	12,315 ±2,801	13,812 ±3,462
Unplanned vessels	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
Reservations:					
Runtime *(MM:SS)*	11:20 ±00:48	10:58 ±01:08	10:31 ±00:54	10:01 ±01:07	11:13 ±01:02
(R)UM value	7,338 ±	8,772 ±	6,686 ±	7,005 ±	5,379 ±
Unplanned vessels	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
Parcel Size:					
Runtime *(MM:SS)*	10:12 ±01:01	10:31 ±00:56	11:22 ±00:48	09:51 ±01:08	09:21 ±01:05
(R)UM value	6,640 ±1,131	6,791 ±1,212	7,222 ±879	6,696 ±931	6,530 ±847
Unplanned vessels	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
Horizon:					
Runtime *(MM:SS)*	09:42 ±01:20	10:03 ±01:01	09:58 ±01:03	10:41 ±00:59	09:20 ±00:50
(R)UM value	5,078 ±876	5,850 ±688	6,427 ±937	7,249 ±951	7,208 ±1,088
Unplanned vessels	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0

Table 10.3: Sensitivity analysis & Extreme conditions testing Node-based SA

Parameter / KPIs	-75%	-50%	0%	+100%	+400%
Arrivals:					
Runtime *(MM:SS)*	01:05 ±00:08	02:10 ±00:34	03:31 ±00:45	03:44 ±00:31	03:29 ±02:34
(R)UM value	4,871 ±292	8,000 ±381	12,413 ±672	10,368 ±993	10,531 ±4,072
Unplanned vessels	0 ±0	0 ±0.48	0 ±0.31	0 ±0	0 ±0
Reservations:					
Runtime *(MM:SS)*	03:04 ±00:40	02:06 ±00:48	04:29 ±00:36	04:50 ±00:39	09:06 ±00:56
(R)UM value	10,407 ±1,076	10,830 ±838	11,895 ±704	10,094 ±800	7,692 ±803
Unplanned vessels	0 ±0.81	0 ±0.91	0 ±0	0 ±0	0 ±0
Parcel Size:					
Runtime *(MM:SS)*	03:13 ±00:51	03:01 ±00:38	02:38 ±00:51	02:49 ±01:00	03:30 ±00:24
(R)UM value	10,866 ±878	10,531 ±815	9,135 ±946	7,005 ±700	6,875 ±662
Unplanned vessels	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
Horizon:					
Runtime *(MM:SS)*	03:40 ±00:56	02:54 ±00:43	02:52 ±00:41	4:28 ±00:50	03:12 ±00:40
(R)UM value	7,886 ±723	8,697 ±604	10,523 ±834	11,618 ±786	12,402 ±803
Unplanned vessels	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0

Table 10.4: Sensitivity analysis & Extreme conditions testing DFS with lookahead

Parameter / KPIs	-75%	-50%	0%	+100%	+400%
Arrivals:					
Runtime *(MM:SS)*	00:10 ±01:02	11:08 ±00:46	18:10 ±00:46	30:00 ±03:48	30:00 ±03:12
(R)UM value	2,386 ±268	6,025 ±287	9,135 ±412	13,314 ±604	20,756 ±2,711
Unplanned vessels	0 ±0.14	0 ±0.39	1 ±0.54	3 ±1.06	32 ±6.79
Reservations:					
Runtime *(MM:SS)*	10:25 ±00:48	10:02 ±01:08	9:58 ±00:54	10:01 ±01:05	10:28 ±01:02
(R)UM value	8,962 ±845	10,140 ±605	8,907 ±813	10,367 ±792	9,616 ±779
Unplanned vessels	0 ±0.91	0 ±0.57	1 ±1.14	2 ±0.77	3 ±1.02
Parcel Size:					
Runtime *(MM:SS)*	12:21 ±01:00	10:34 ±00:56	11:10 ±00:48	12:07 ±01:08	11:15 ±01:05
(R)UM value	11,561 ±796	10,293 ±806	9,842 ±818	9,246 ±609	9,875 ±803
Unplanned vessels	0 ±0.29	0 ±0.66	1 ±0.58	2 ±0.79	3 ±1.23
Horizon:					
Runtime *(MM:SS)*	00:51 ±00:19	11:43 ±09:01	11:31 ±01:03	12:18 ±00:59	12:01 ±00:50
(R)UM value	6,405 ±801	11,467 ±682	11,457 ±673	11,242 ±799	11,385 ±721
Unplanned vessels	5 ±1.74	2 ±1.0	0 ±0.8	0 ±0.6	0 ±0.6

10.2 Validation

The validation phase is concerned with establishing whether the results from the scheduling system are applicable to the situation at liquid bulk terminals. The main question in the validation step is: "Did we build the right thing?" (Van Dam et al., 2013, p.98). To determine this, two different validation tests have been performed: Face validation and replicative validation.

10.2.1 Face Validity

A structured walkthrough is a structured session in which experts evaluate whether the systems structure corresponds sufficiently to the real system. In this structured review, experts from Systems Navigator were asked to conduct this validation test. The main topics of focus in this walkthrough are listed below. During this test, no major discrepancies were found which could impact the systems validity.

- Are routes generated correctly?
  - Are enough routes generated?
  - Are no infeasible routes generated?
  - Are the processing times realistic estimates?
- Are the allocation attributes sufficient to represent the terminals' preferences?
  - Are the allocation attributes calculated correctly?
  - Are the allocation attribute weights a good enough estimate for representing the terminals' preferences?
- Are the route feasibility rules correct?
  - Are no infeasible allocations generated?
  - Are the feasibility rules on the correct level of detail?

10.2.2 Data and Assumption Validation

When translating the real-world into a model several inferences and assumptions need to be made based on the real-world data. This section will describe the assumptions made throughout the designing of the MAS and determine how these influence the validity of the system.

During the validation test several assumptions were found which in some cases can influence the validity of the system. The first of this assumption is that the the processing times of all operations except for the pumping step in the system are considered fixed, while these can vary quite heavily in reality. This was already briefly discussed in Section 9.3 where it was mentioned that adding stochasticity can add much insight on the robustness of the optimized plan. Furthermore, in the current implementation berths cannot block other berths, such as present in the

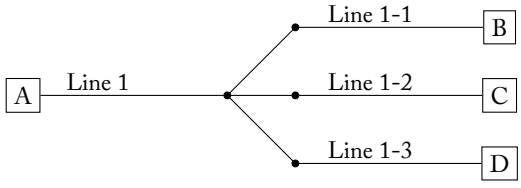
winged-berth concept as described in Chapter 3. This should be added in the future if the system gets implemented at a terminal with such berths. Finally, the system is currently not able to deal with using lines as temporary storage , as it is currently unclear how this should be implemented.

10.2.3 Replicative validation

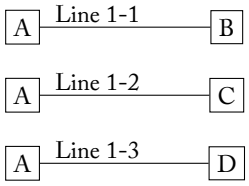
Replicative validation is a method of validation which compares the systems' data to real-world or previously generated and validated data. Unfortunately, there are no complete scheduling systems for liquid bulk terminals available with which this system can be compared, although some components do exist.

The first available component for replicative validation is a (high-level) route-generation model, which was developed earlier by Systems Navigator for application in simulation studies. Several test cases were set up in which the routes generated by the Route Finder agent were compared with the generated routes. Several minor differences were found due to the difference in detail level between both tools which were caused by the old model being unable to deal with certain infrastructure set-ups as shown in Figure 10.1.

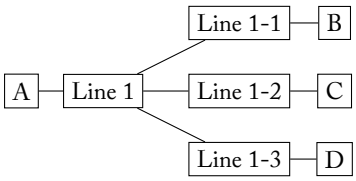
The second available component is a high-level berth selection model, which was also developed by Systems Navigator for application in simulation studies. Once again, several test cases representing the example terminal described in this thesis were set up and manually compared. Minor differences were observed in the results from the model and results from the FCFS heuristic implemented in the MAS, which originated from the MAS using discrete time steps with an hourly interval, while the simulation component uses a continuous time domain.



(a) Original layout



(b) Modeled in route generator



(c) Modeled in Route Finder agent

Figure 10.1: Infrastructure modelling differences

10.3 Conclusion

Overall, the verification and validation tests suggest that this system is valid for proposing scheduling options to planners at liquid bulk terminals. Furthermore, it can be used to propose simple route options, perform internal re-optimization and handle some propagation effects.

There are some topics which need to be considered when applying this system for scheduling at liquid bulk terminals. First of all, it considers all operations times to be fixed per arrival category, without considering stochasticity or external influences such as weather. Furthermore, it does not yet allow certain berths to block other berths when serving a vessel such as in the winged berth example described in Chapter 3. Finally, all berths in a group will be blocked for manoeuvring when an arriving or post-operations operation is being performed in that group, without considering waterways. Finally, the system cannot deal with using lines as a type of intermediary storage.

Experimentation 11

Now that a prototype implementation for a *Multiagent System* (MAS) scheduling system in the liquid bulk domain is designed, implemented, and validated, it can be used to run experiments to help answer the main question of this thesis:

Can a *Multiagent System* (MAS) be efficiently applied in a scheduling support system for liquid bulk terminals?

The main goal of this chapter is to evaluate and experimentally support the proposed solution strategies from Part II of this thesis. It starts by describing an experimental framework for testing the discussed implementations. It then goes on by identifying the exact attributes for experimentation, and finishes by discussing the results.

11.1 Experimental Design

The experimental set-up consists of four parts:

- **A Task Generator:** Generates pseudo-random arrival and reservation lists based on pre-set parameters on plan complexity;
- **The MAS:** Uses the input from the Task Generator to solve the scheduling problem;
- **A Database:** Stores the experimental results;
- **A Control Script:** Controls all components for running the experiments;
- **An analytical component:** Allows the user to easily interpret the experimental results.

The Task Generator is written in F# and uses a pseudo-random number generator to generate an arrival list containing the vessels which should participate in the scheduling process, as well as a reservations list to indicate pre-existing reservations on infrastructure. The input parameters needed for the task generator as well as their corresponding ranges are shown in Table 11.1. Varying these attributes is done to test the system with a varying problem size as well as problem complexity.

The second part of the experimental setup is the MAS containing the solution strategies discussed in part II of this thesis: queue-based *Simulated Annealing* (SA), node-based SA, *Depth-first Search* (DFS) with lookahead, and *First-come, first-served* (FCFS), as well as their regret-based counterparts. The MAS will solve the scheduling problem using each of the solution strategies for each of the problem

Table 11.1: Task Generator Inputs

Attribute	Range
Arrivals	0 - 50
Avg. parcels / vessel	1-5
Pre-existing reservations	0 - 50
Planning Horizon	1-10
Avg. parcel size	15000-45000

Table 11.2: Solution specific Attributes

Solver	Attribute	Range
B&B	Selection strategy	{BeFS, BFS, DFS}
	Bounding function	{LP,Job-shop, Knapsack}
SA	Initial Temperature	1 - 1000
	Temperature decay	0.001 - 0.1
	Queue Length	500 - 5000
	Initial Patience	100 - 1000
	Patience Ratio	2 - 100
	Scaling	{None, Z-Score, Min-Max}
	Search Method	{Queue, Neighbours}
	Starting Node	{FCFS, DFS}
FCFS	Scaling	{None, Z-Score, Min-Max}
DFS	Lookahead levels	{0 - 2}

instances generated by the task generator to improve the comparability of the results. Apart from the problem instance the MAS uses several attributes which are specific for the chosen solution strategy. These attributes with their ranges are shown in Table 11.2.

After the MAS has completed a run, the generated schedules and attributes are inserted into a database, from which further exploration of the data can be done using an analytical component. The experimentation will be controlled by an R script which will determine the attribute values and interact with the task generator and the MAS. Attributes were pulled from a latin hypercube with 250 replications.

All experiments were run on a dedicated Windows 8.1 machine running on a Intel Core i5 4690K processor running at 4.4 GHz with 16 GB of RAM. Timings are given in wall-clock time. The task generator is written in R and the MAS is written in F#. If a *Linear Programming* (LP) solver was needed the Microsoft Solver Foundation solver was used. Experiments are terminated if their runtime is over 1 hour.

Some of the discussed solvers, such as SA, require random numbers. In order to increase repeatability we maintain seperate random number generator streams for the task generator and the solver algorithm. To reduce variability on the solvers, common random numbers are used (Owen, 2013).

The main *Key Performance Indicators* (KPIs) which are relevant in discussing the performance of the scheduling system are the solution quality and the runtime needed to find a solution. Since these are both highly dependent on the problem size and complexity it is difficult to directly compare results. Therefore, the solution quality will be given as a percentage improvement compared with a FCFS heuristic for the same problem instance.

Furthermore, there is no easy method available for comparing the results from a *(Random) Utility Maximization* ((R)UM) and a *(Random) Regret Minimization* ((R)RM) model since these are both different scales of solution quality. The reported solution qualities in the results section of this chapter will all be calculated using the (R)UM model, as the (R)UM model does not allow the comparison of alternatives over different alternative sets. In order to compare the (R)UM and (R)RM outcomes each run of the system will report its top ten best solutions using both models. We can then report a difference measure between the (R)UM and (R)RM outcomes to determine whether future empirical research in this area can be beneficial. For measuring the edit distance[1] the Levenshtein distance (Equation 11.1) will be used on the top 10 outcomes for each modelling technique.

$$\text{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j)+1 \\ \text{lev}_{a,b}(i,j-1)+1 \\ \text{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \tag{11.1}$$

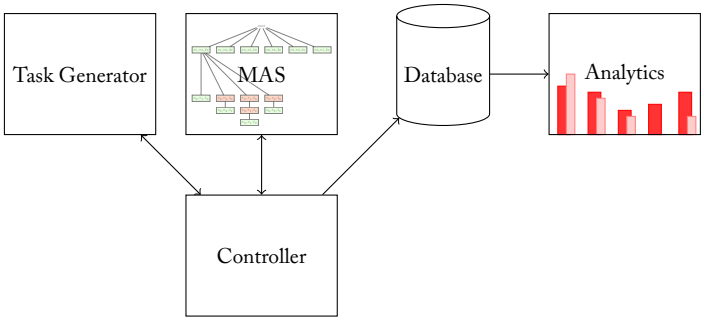


Figure 11.1: Experimental Setup

[1]The edit distance is a quantification of the dissimilarity of two lists, calculated by counting the number of operations needed to transform one into the other

11.2 Results

Having defined the experimental design, this chapter will now move on to discuss the findings from the performed experiments. An initial exploration of the systems results was done by plotting the relative improvements of the different solution strategies combined with the runtime of the solution strategy in a faceted scatterplot for varying problem instance complexities. These plots are shown in Figure 11.3. Furthermore, in order to get a more in-depth look on the allocation, the various solutions for a specific problem instance can be visualized in a Gantt chart which shows the allocation of parcels on the infrastructure at the terminal, as well as some properties of the overall allocation (Figure 11.2).

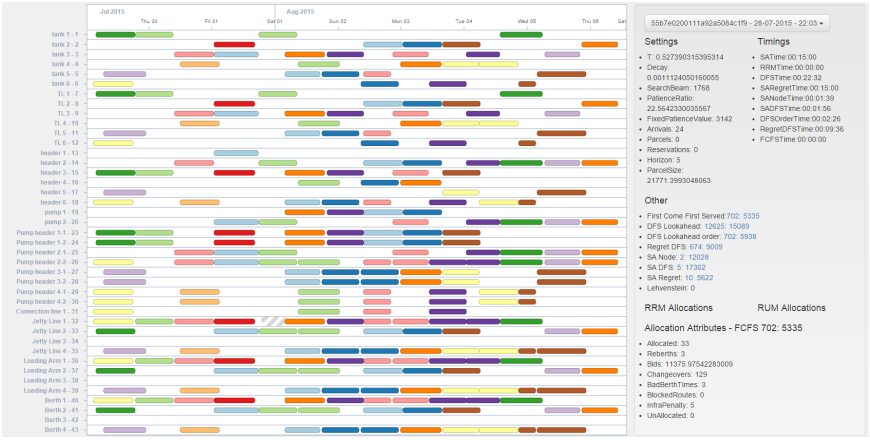


Figure 11.2: Interactive interface for exploration of solutions

One of the first observations which can be made from the results is that, in case of simple problem instances, the spread with regards of the solution quality is very low (top left of Figure 11.3). This is due to all algorithms being able to find high valued solutions in these instances. However, as the problem grows more complex, some interesting patterns emerge, which will be further described per solution strategy in the upcoming sections.

11.2.1 Branch & Bound

The observant reader may have noticed at this point that the *Branch and Bound* (B&B) solution strategy is absent from Figure 11.3. Early on in the experimentation it became clear that none of the bounding functions which were proposed for application in the B&B strategy provided a good bound to efficiently navigate the search tree. The special purpose *Winner Determination Problem* (WDP) bounding LP model provided a too low estimate of the upper bound. As a consequence of this low estimate, the upper bounds of all the nodes became lower than the current

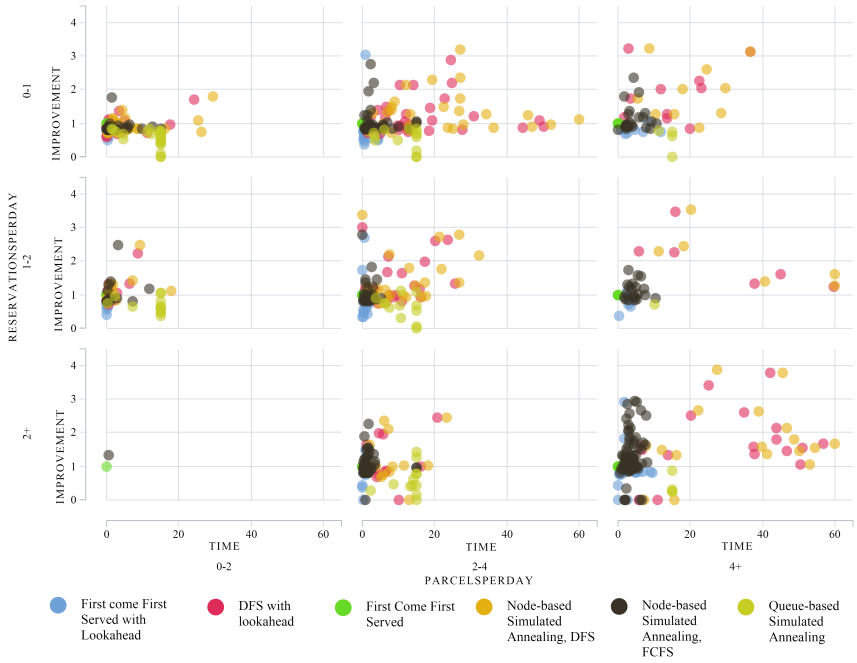


Figure 11.3: Relative solution improvement and runtime for varying problem instance complexities for different solution strategies

best lower bound, causing all nodes to be bounded and the search to stop without a final allocation. This low estimate can be explained by the fact that the LP model does not incorporate the arrival and departure of vessels over time, resulting in the model thinking that all infrastructure had been allocated and therefore being unable to provide new allocations to vessels. Adjusting the special purpose LP model to include a time dimension made the runtime of the bounding function too long to be feasible in practice.

The second bounding function strategy was based on relaxing the problem instance so that only the berths were considered as a restricting resource. This made it possible to represent the upper bounding function as a job-shop problem or an even more relaxed online knapsack problem. In practice, these models provided a too weak upper bound, resulting in infeasibly long run times for all but the simplest problem instances. This result is easily understood when we consider a problem in which we have only four arrivals. Since the example terminal used in this experimental setup consists of four berths, the upper bound function will always be able to allocate all vessels optimally, resulting in no bounded nodes. However, as we have already seen in Chapter 9, there are already $2,07 * 10^{12}$ possible allocation options in such a case, making the run time of the algorithm infeasibly long.

11.2.2 Simulated Annealing

Although in initial testing, the queue-based SA implementation seemed promising, it quickly proved to be a relatively slow solution strategy as the problem size increased. This was mainly due to the fact that the queue-based implementation sometimes behaved too much like a *Best-first Search* (BeFS) strategy which resulted in it being unable to pass over difficult allocations while navigating the search tree (Figure 11.4). Furthermore, the necessity of scaling the values caused the implementation to be much slower than intended, which was partly caused by an inefficient implementation. While this implementation could be improved in the future it is expected that focusing on other areas would yield much better results.

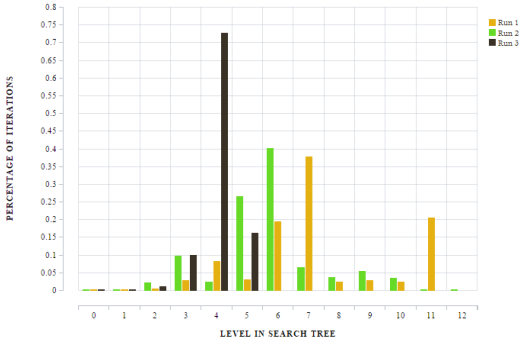
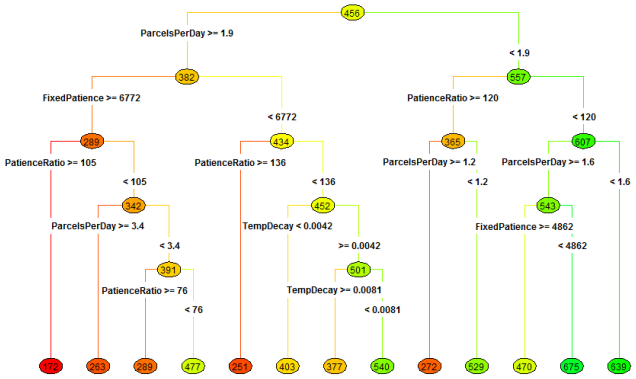


Figure 11.4: Queue-based SA - Percentage of iterations per level in search tree

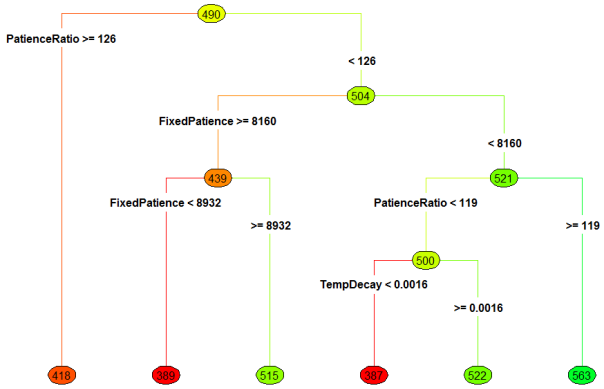
The more traditional node-based SA implementation however provided much better results, as it shows to be relatively predictable in both quality and run time. Furthermore, an advantageous property of the node-based SA is that it, since it always starts with a solution, always has a solution available at any time throughout the run. This means that the system can be stopped at any moment and still present a better solution to the human planner.

As was already discussed in Chapter 9 there are several tuning dimensions for SA implementations. In order to find the optimum parameters, a set of C4.5 decision trees were set up using the RPart R Package. These are displayed in Figure 11.5. In order to avoid overfitting on the data, the tree was pruned based on a lowest cross-validation error. Based on the data, the following tuning parameters were chosen as final values:

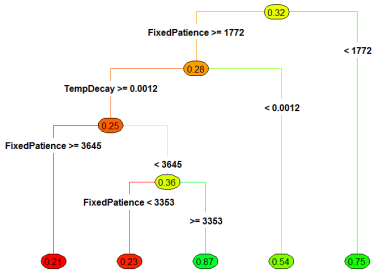
- **Scaling Factor:** $0.1 * (0.6 * parcels + 0.4 * reservations)$
- **Patience Factor:** 100 + 5 * Scaling factor
- **Initial Temperature:** 500 + 3 * Scaling factor
- **Temperature Decay:** 0.006 - 0.1 * Scaling Factor



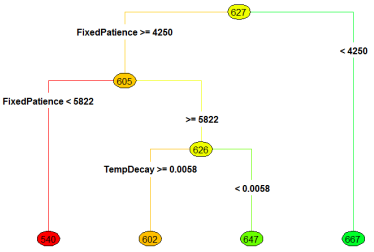
(a) For all problem instances



(b) For complex problem instance



(c) For simple problem instance



(d) For normal problem instance

FIGURE 11.5: Simulated Annealing Parameter tree per complexity class

The SA algorithm was run with both a FCFS result as starting node and with a DFS with lookahead result as starting node. While using the DFS result as a starting node provided better results, it is not feasible in practice due to the large runtime of the DFS model.

Based on these optimized parameters, several more runs are performed. Figure 11.6 shows a stratified sample of several of these runs. The figure shows the relative improvement obtained over the run time of the SA run. This image shows a clear pattern in the behaviour of the SA solver as it is often able to find large improvements relatively early and has several plateau sections in which no better solutions are found for a while. It also becomes clear from this image that the plateau phase at the end of the run is relatively short or even non-existent for certain runs. This can be explained by the chosen implementation of the stopping conditions. The SA run stops if a better solution is not found for n nodes, where n is determined by multiplying the iteration number when the previous best node was found with a fixed ratio. Therefore, the presence of plateaus early in the run can cause the SA to stop relatively early.

These results indicate that searching longer might provide better results for these runs. We can conclude from this that further tweaking of the performance parameters can possibly further increase the quality of the found solution. Further research on this subject could attempt to make the implementation of the algorithm more efficient so that a larger portion of the search space can be explored in the same time or implement Adaptive Simulated Annealing techniques such as Quenching, Re-annealing, and Fast Annealing (Ingber, 1996), which are better capable of adjusting the tuning parameters of SA than the scaling factor introduced in this thesis.

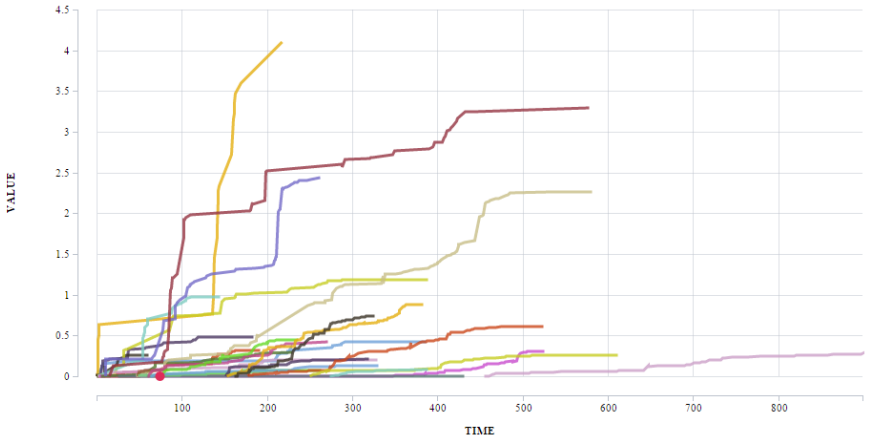


Figure 11.6: Stratified sample of SA runs for single problem instance

11.2.3 Informed Search

The third and final set of solution strategies contains the informed search approaches. Part of this set are *First-come, first-served* (FCFS), and *Depth-first Search* (DFS). Both these approaches were implemented using no lookahead, a single-level lookahead, and a two-level lookahead.

FCFS was mainly used as a baseline to compare the other solution strategies with as it is the heuristic which is most commonly applied by the terminals. Adding a single-level lookahead it was possible to increase the solution quality of the algorithm with about 10%. Adding another lookahead was able to increase the quality with another 10%, but drastically increased the run time.

DFS without any lookahead yielded very disappointing results. This can mainly be explained by the random nature of the first allocations it makes. Since the node values on the top layers of the tree are quite similar to each other, the DFS would often start with allocating a vessel which wasn't due to arrive until quite some time. However, this made allocating earlier vessels much more difficult due to the added constraint further on in the plan. Adding a single lookahead level mitigated much of these problems and the DFS with a single-level lookahead overall provided the highest quality solutions. However, the runtime of the algorithm was too high to be considered implementable in practice.

11.2.4 Regret vs Utility

As was already described in Chapter 9, all solution strategies were also implemented using their regret-based counterpart. While some of these strategies (*Best-first Search* (BeFS), queue-based *Simulated Annealing* (SA)) used the (R)UM model to navigate the tree and the regret model only to do the final allocation weighing, other strategies (node-based SA, *Depth-first Search* (DFS)) also used the (R)RM model to navigate search tree. Since there was no possibility of doing an empirical study within the time frame of this thesis it is not possible to answer which of the two methods is better suited for proposing scheduling options to human planners at liquid bulk terminals. However, it is possible to see whether there are differences and whether there are patterns distinguishing the two solution methods.

When comparing the top ten results from all solution methods a Lehvenstein distance from 3 to 7 was observed, with an average distance of around 5. Furthermore, it was found that regret models are much slower but apart from queue-based SA still feasibly implementable in practice. However, faster methods for calculating regret models would still greatly benefit their application in scheduling.

Chapter 9 raised the concern that applying a hybrid (R)UM/(R)RM approach could cause the differences between both approaches to be smaller than anticipated. When comparing the results it was found there was no significant difference in edit distance (with $p < 0.05$) between the solutions which used the full models, and the solutions using the hybrid models. This indicates that the mentioned effect

is relatively weak in practice, although further research would be needed to fully support this claim.

Apart from the edit distance between the top ten solutions it was also attempted to find patterns distinguishing the utility models' outcomes from the regret models' outcomes. Several hypotheses were drafted such as the relative "averageness" of the outcomes, but these could not be backed by the gathered data. However, for the sake of completeness a single run will be described. For a specific problem instance containing 10 vessels, each carrying a single parcel of $30,000$ m^3 of product, which are to be planned in a ten day planning horizon, the obtained results are shown in Table 11.3.

The lehvenstein distance between the two node lists can then be calculated using the Wagner-Fischer algorithm, which is shown in Table 11.4.

Table 11.3: (R)UM vs. (R)RM results for a specific problem instance

	Node	Reberths	Bids	Changeovers	Bad berth times	Blocked Routes	Infra. Penalty
(R)UM	1	2	13,347	63	4	0	3
	2	2	13,347	64	4	0	3
	3	2	13,347	65	4	0	3
	4	2	13,347	66	4	0	3
	5	2	13,347	67	4	0	3
	6	3	13,265	59	4	0	2
	7	2	13,347	68	4	0	3
	8	3	13,265	60	4	0	2
	9	2	13,347	69	4	0	3
	10	3	13,265	61	4	0	2
(R)RM	1	2	13,347	63	4	0	3
	2	2	13,347	64	4	0	3
	3	2	13,347	65	4	0	3
	4	2	13,347	66	4	0	3
	5	2	13,347	67	4	0	3
	7	2	13,347	68	4	0	3
	9	2	13,347	69	4	0	3
	11	2	13,347	70	4	0	3
	12	2	13,347	71	4	0	3
	13	2	13,347	72	4	0	3

Table 11.4: Lehvenstein distance between solution nodes for specific problem instance

	ϵ	1	2	3	4	5	7	9	11	12	13
ϵ	0	1	2	3	4	5	6	7	8	9	10
1	1	0	1	2	3	4	5	6	7	8	9
2	2	1	0	1	2	3	4	5	6	7	8
3	3	2	1	0	1	2	3	4	5	6	7
4	4	3	2	1	0	1	2	3	4	5	6
5	5	4	3	2	1	0	1	2	3	4	5
6	6	5	4	3	2	1	2	3	4	5	6
7	7	6	5	4	3	2	1	2	3	4	5
8	8	7	6	5	4	3	2	3	4	5	6
9	9	8	7	6	5	4	3	2	3	4	5
10	10	9	8	7	6	5	4	3	4	5	6

Given these results it would be interesting for a future empirical study to go in depth on the acceptability of solutions generated by utility models and their regret-based counterparts.

11.2.5 Terminal Preferences

During the experimentation and development process an interesting observation was made with regards to the addition of terminal preferences in the allocation value.

To quickly recap, one of the main goals of the terminal which was incorporated in the allocation value was to reach a high terminal utilization. This value was operationalized into four measurable components: Low power usage, idle time minimization, idle time optimization and low number of blocked routes. These components are added to the allocation valuation to ensure that, in a later stage of the planning process, more vessels can still be added to the schedule.

By adding these attributes to the search process, an interesting side-effect surfaces. The addition of these attributes caused the search algorithms to come up with better solutions in a shorter timespan, since the attributes guided them throughout the search process. Statistical testing using a paired student t-test showed that the average of bid values was statistically higher in cases where terminal preferences were included than in cases where they were not with $p < 0.05$. In conclusion, we can state that even in cases where additional attributes are not needed for the final allocation value determination, it can still be beneficial to add them to improve the run time and quality of the solution strategies.

11.3 Conclusion

The main goal of this chapter was to test the various WDP solver strategies proposed in Chapter 9 as well as the system as a whole on solution quality, run time, and general applicability in a scheduling support system for liquid bulk terminals.

It started by describing an experimental set up consisting of a task generator, the MAS, a database, an analytics component and a control system.

It was found that none of the proposed *Branch and Bound* (B&B) bounding functions provided a good bound for this problem. The special purpose WDP LP model does not take into account the arrival and departure of vessels over time, causing the upper bound to be estimated too low. Extending this model to take a time domain into account made the run time infeasible. The bounding functions which were based on a relaxed instance of the problem which only took into account berth restrictions provided a too weak bound, causing the overall run time of the algorithm to be too long.

Depth-first Search (DFS) with a single-level lookahead provided the highest quality results overall, but suffered from a relatively long run time. Removing a lookahead level however, drastically decreased the result quality.

Node-based *Simulated Annealing* (SA) seems like the best candidate for implementation in a scheduling support system at liquid bulk terminals, as it is able to provide substantial improvements over the *First-come, first-served* (FCFS) heuristic currently employed in a relatively short time frame. Furthermore, a large advantage of this algorithm is that a solution is always available, meaning the planner can interrupt the run of the algorithm but still be provided with a better than FCFS solution.

Furthermore, it was found that adding terminal preferences which focused on the high terminal utilization value were able to increase the performance of various search algorithms substantially, as they acted like guides through the search process.

Finally, the utility-based models were compared with their regret-based counterparts. In this comparison it was found that the top ten results typically differed from each other with an edit distance of 5. Also, while run times were typically longer than the utility model-based algorithms, they were still feasibly implementable in practice. Further empirical studies would be needed to determine whether regret-based models provide better accepted solutions in practice.

Conclusions 12

In recent years much literature emerged discussing the concept of *Multiagent Systems* (MASs) for planning and scheduling which proposes many benefits in composability and reusability as well as multi-objective optimization. The main research question of this thesis was:

Can a *Multiagent System* (MAS) be effectively applied in a scheduling support system for liquid bulk terminals?

The research started with an extensive background research, aimed at answering the question: "Who are the main stakeholders involved in liquid bulk scheduling"? During this research it was found that there are a number of stakeholders involved in the scheduling process, of which the most relevant are the terminal and customer, each consisting of several independent departments or entities. A terminal typically serves multiple customers which are largely uninterested in each others' performance. Furthermore, there are several conflicting goals between the terminal and customers, as the customers are typically only interested in achieving short turnaround times, while the terminal also has more global objectives such as a high overall utilization and low pressure on operations. Because of these conflicting interests, it is vital for the acceptability of the proposed schedules to incorporate the different goals of the stakeholders into the scheduling process.

Replacing human planners by scheduling algorithms is a difficult problem in practice, since planners spend much time on collaboration, communication, and negotiation about various (soft) constraints and goals. Therefore, one of the main requirements of the scheduling algorithm was that it should be implemented in a scheduling support system. This context analysis was part of the second research question: "What are the requirements for application of an optimization system in a scheduling support system"? For being applicable in a scheduling support system, the system should have several extra features such as supporting Freezing jobs, Re-optimization, Processing propagation effects, and Internal re-optimization.

By combining the answers of the first two questions it was possible to answer the third research question: "What are the main characteristics of scheduling problems in liquid bulk terminals"? It was found that the problem resembles a specific version of the job-shop problem: $JMPT|pmtn, prec, r_i, s_{ij}|multi$. In this type of problem, the tasks require multiple machines simultaneously to be processed, which resembles a liquid bulk terminal since all infrastructure elements which form a route are required simultaneously. Since this problem is strongly NP-hard it is generally infeasible to find an optimal solution as the problem complexity grows, and efficient algorithms are needed to find solutions.

After defining the specific scheduling problem in liquid bulk terminals, this thesis moved on to determining how MAS could be applied. This part answered the following research questions: "What are the main differences in applying MAS between liquid bulk scheduling problems and regular job-shop scheduling problems, and how can these differences be overcome in practice"?. Although MASs have been applied on numerous occasions for planning and scheduling problems, no literature was found in which the specific *Job-shop Problem with Multiprocessor Tasks* (JMPT) was discussed. This lack of literature on the subject most probably originates from the relatively low amount of JMPT problems which are encountered in real-world cases. Since JMPT differ from traditional Job-Shop problems on several key aspects, the literature did not provide a one-to-one fit for MASs on this problem. Several MAS strategies were evaluated for applicability based on four main attributes:

- Whether the solution strategy provided a good analogy to the model;
- Whether the strategy could deal with a time dimension and the departure and arrival of vessels over time;
- Whether the strategy could deal with shared infrastructure between routes;
- Whether the strategy was able to incorporate terminal-side preferences.

It was found that *Combinatorial Auctions* (CAs) provided the best fit on the scheduling problem since it is the only strategy which could deal with all these attributes. In this strategy the terminal takes the role of auctioneer and the vessels submit bids on bundles of infrastructure which together make up a route. The final allocation is then the result of solving a *Winner Determination Problem* (WDP) in which the auctioneer takes all bids and allocates the items so that the highest value is obtained. Typical CA incorporate a payment system in which bidders use virtual or real currency to pay for their items. It was not possible to implement a payment system in liquid bulk scheduling due to the existing contract structure between the terminals and its customers; therefore, a limitation of this auction strategy is that it is not strategy-proof and does not enforce truthfulness. Further research in this area could usefully explore how the proposed CA mechanism fits on other types of scheduling problems.

As was already discussed, the incorporation of multi-attribute decision making is vital in designing an acceptable scheduling solution in the liquid bulk domain. Several options for applying multi-attribute decisions in CAs were evaluated, including Pareto optima, hard constraints, percentage bids and choice models. Of these options only *Discrete Choice Model* (DCM) provided the needed flexibility. Two main types of DCM were proposed, the *(Random) Utility Maximization* ((R)UM) model, as well as a *(Random) Regret Minimization* ((R)RM) model. The (R)UM model is the most common choice model in which the total utility of an alternative is based on the sum of weighted attribute values. The (R)RM model is a relatively new type of decision model which is shown to better fit the preference data of humans in several cases. A downside of the (R)RM model is that it suffers from a relatively

high computational complexity $O(n^2)$ versus $O(n)$ for the (R)UM model. Since the goal of the MAS is to propose scheduling options to the human planner, it can benefit from a decision making technique which more closely resembles the choices the planner would make himself. It is currently unclear for which types of problems the (R)RM model provides a better fit. Since an empirical study was not feasible within the time frame of this thesis, both models were implemented and compared analytically. Even in the case where an empirical study is feasible for determining parameter weights, there are still some limitations to be considered. First of all, we have seen that the trade-off weights originate from many different departments and stakeholders within the terminal, which can cause conflict and hamper constructing precise trade-off weights. Furthermore, stakeholders can be uncertain about their feature weights, or unwilling to communicate these.

Solving the auction WDP is once again a NP-hard problem, for which several solution strategies were proposed. A *Branch and Bound* (B&B) algorithm was implemented using two types of bounding functions: a special purpose WDP *Linear Programming* (LP) model, and a relaxed problem instance bounding function. Both options were found to be inadequate as the first provided too strong boundary estimates, and the second too weak. *Depth-first Search* (DFS) with a single-level lookahead provided the overall highest quality results but suffered from a too long run time to be feasible in practice. Node-based *Simulated Annealing* (SA) was the most feasible solution strategy as it was able to provide solution qualities which are on average 50% better in on average three minutes. Furthermore, it has a desirable feature in that the algorithm can be halted during the run and still provide a better solution. A limitation of this method is that it does not provide an optimal solution to the scheduling problem. If optimality is desired, the most promising solution method would be to apply a B&B algorithm, although further research would need to be carried out to find suitable bounding functions.

All algorithms were implemented using an utility-maximizing strategy as well as a regret-minimizing strategy. For algorithms which typically need to compare large number of nodes during the run a hybrid (R)UM/(R)RM was set up which navigated the search space with the utility model and only did the final weighing of nodes with the regret model. Other solutions used the regret model for both navigating the search space and the final allocation. A comparison was set up in which the utility-based models were compared with their regret-based counterparts. In this comparison it was found that the top ten results of the (R)UM and the (R)RM model typically differed from each other with an edit distance of five. Also, while run times of the regret-based implementations were typically longer than the utility model-based algorithms, they were still feasibly implementable in practice. Further empirical studies would be needed to determine whether regret-based models provide better accepted solutions in practice.

In conclusion, we can state that Multiagent System can indeed be efficiently applied for proposing scheduling options in a liquid bulk terminal. Combinatorial Auction inspired control mechanisms provide a natural fit on the Job-shop Problem with Multiprocessor Tasks in the terminal, and offer much flexibility with regards

to the implementation of both terminal as well as customer preferences. Solving the WDP in auctions can be done using a variety of search strategies, of which Simulated Annealing performed the best in this case. Using SA, a 50% increase in solution quality was obtained in comparison with a First-come, first-served heuristic. Further gains are expected to be possible by for example applying Adaptive SA techniques. Several further extensions to the MAS can be made by implementing other search techniques such as Tabu Search, and Genetic Algorithms. Furthermore, the MAS in this thesis assumed deterministic processing times for operations, while in practice these are highly stochastic. Adding this stochasticity allows the MAS to determine the robustness of the allocation, and perhaps feed this back into the DCM as an additional attribute.

To further increase the reusability of the designed system, it would be greatly beneficial to allow users to easily change feasibility constraints set in the system. Designing a Domain Specific Language for schedule feasibility would be a helpful tool in achieving higher reusability.

The findings from this thesis are not only applicable in Liquid Bulk Terminals, but can be applied in various domains such as dry bulk handling and food processing, as these have similar scheduling problems with similar operational preferences.

Furthermore, this thesis has shown that it is possible to implement (Random) Regret Minimization Discrete Choice Models for scheduling optimization problems, although further empirical research is necessary to determine whether they provide a better fit on the problem. An interesting expansion of this MAS can be made by incorporating a learning component, which is able to update the DCM values based on the chosen proposed schedule option by the human planner.

The research presented in this thesis is scoped on the application of MAS as a suitable candidate for implementing a scheduling system in a larger scheduling support system. While this thesis has clearly shown that MAS can be successfully applied, there are obviously many other solution directions which could also yield positive results. While MAS provides an intuitive framework, in the end it does not matter whether the implementation uses agents, functions or objects, as they are simply abstraction layers. Furthermore, the scope of this study was limited in terms of assuming that a scheduling system is a vital component of a scheduling support system. Due to this assumption, several other possibilities for assisting the planner such as expert systems and unsupervised learning of schedule patterns, were not evaluated.

Bibliography

Agnetis, A., Billaut, J.-C., Gawiejnowicz, S., Pacciarelli, D., & Soukhal, A. (2014). *Multiagent Scheduling*. Berlin Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-41880-8

Alaska Tanker Company. (2012). Alaskan Legend. Retrieved June 18, 2015, from http://www.aktanker.com/fleet/

Aldewereld, H., Dignum, F., & Hiel, M. (2012). Decentralised Warehouse Control through Agent Organisations. In R. Hamberg & J. Verriet (Eds.), *Automation in warehouse development* (Chap. 3, pp. 33–44). Springer London. doi:10.1007/978-0-85729-968-0_3

Alfonzetti, S., Dilettoso, E., & Salerno, N. (2006). Simulated annealing with restarts for the optimization of electromagnetic devices. *IEEE Transaction on Magnetics*, *42*(4), 1115–1118.

Amir, O., Sharon, G., & Stern, R. (2015). Multi-Agent Pathfinding as a Combinatorial Auction. In *Twenty-ninth aaai conference on artificial intelligence (aaai-15)*. Austin, TX.

Archer, A. & Tardos, É. (2001). Truthful mechanisms for one-parameter agents. *Proceedings 2001 IEEE International Conference on Cluster Computing*. doi:10.1109/SFCS.2001.959924

Ashlagi, I., Fischer, F., Kash, I. a., & Procaccia, A. D. (2010). Mix and Match. In *Proceedings of the 11th acm conference on electronic commerce* (pp. 305–314). New York: ACM.

Aydin, M. & Öztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, *33*(2-3), 169–178. doi:10.1016/S0921-8890(00)00087-7

Baker, A. D. (1998). A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: Dispatching, scheduling, and pull. *Journal of Manufacturing Systems*, *17*(4), 297–320. doi:10.1016/S0278-6125(98)80077-0

Bárta, J., Stepankova, O., & Pechoucek, M. (2002). Distributed Branch and Bound Algorithm in Coalition Planning. In V. Mařík, O. Štěpánková, H. Krautwurmová, & M. Luck (Eds.), *Multi-agent systems and applications ii* (Chap. 8, pp. 159–168). Springer Berlin Heidelberg. doi:10.1007/3-540-45982-0_8

Ben-Akiva, M. E. & Steven, R. (1985). *Discrete choice analysis: theory and application to travel demand*. Cambridge, MA, USA: MIT Press.

Bérubé, J.-F., Gendreau, M., & Potvin, J.-Y. (2009). An exact -constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. *European Journal of Operational Research*, *194*(1), 39–50. doi:10.1016/j.ejor.2007.12.014

Blażewicz, J., Pesch, E., & Sterna, M. (2000, December). The disjunctive graph machine representation of the job shop scheduling problem. *European Journal of Operational Research*, *127*(2), 317–331. doi:10.1016/S0377-2217(99)00486-5

Blum, A. (2008). *Lecture 13: CMU 15-451 (Algorithms), Fall 2008*. Computer Science - Carnegie Mellon University.

Boutilier, C., Patrascu, R., Poupart, P., & Schuurmans, D. (2006). Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, *170*(8-9), 686–713. doi:10.1016/j.artint.2006.02.003

Boutilier, C., Sandholm, T., & Shields, R. (2004). Eliciting bid taker non-price preferences in (combinatorial) auctions. *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-2004)*, 204–211.

Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Cambridge: Harvard University Press.

Brucker, P. (2007). *Scheduling Algorithms*. Springer Berlin Heidelberg. doi:10.1007/978-3-540-69516-5

Brucker, P. & Krämer, A. (1995). Shop scheduling problems with multiprocessor tasks on dedicated processors. *Annals of Operations Research*, *57*(1), 13–27.

Buer, T. & Pankratz, G. (2010). Solving a bi-objective winner determination problem in a transportation procurement auction. *Logistics Research*, (439), 1–14. doi:10.1007/s12159-010-0031-8

Bülbül, K. & Kaminsky, P. (2013). A linear programming-based method for job shop scheduling. *Journal of Scheduling*, *16*(2), 161–183. doi:10.1007/s10951-012-0270-4

Burns, M. G. (2015). *Port Management and Operations*. Boca Raton, Florida: CRC Press.

Carson, J. S. (2002). Model verification and validation. In *Proceedings of the 2002 winter simulation conference* (pp. 52–58). doi:10.1109/WSC.2002.1172868

Chankong, V. & Haimes, Y. Y. (1983). *Multiobjective Decision Making - Theory and Methodology* (A. P. Sage, Ed.). New York, NY, USA: Elsevier Science Publishing Co. Inc.

Chorus, C. G. (2010). A new model of random regret minimization. *Ejtir*, *10*(10), 181–196.

Chorus, C. G. (2012a). Random Regret Minimization: An Overview of Model Properties and Empirical Evidence. *Transport Reviews*, *32*(December 2011), 75–92. doi:10.1080/01441647.2011.609947

Chorus, C. G. (2012b). Random Regret Minimization: An Overview of Model Properties and Empirical Evidence. *Transport Reviews*, *32*(March 2015), 75–92. doi:10.1080/01441647.2011.609947

Chorus, C. G. (2012c). *Random Regret-based discrete choice modeling : A tutorial*. Springer, Heidelberg, Germany.

Chorus, C. G., Annema, J. A., Mouter, N., & van Wee, B. (2011). Modeling politicians' preferences for road pricing policies: A regret-based and utilitarian perspective. *Transport Policy*, *18*(6), 856–861. doi:10.1016/j.tranpol.2011.05.006

Chorus, C. G., Arentze, T. a., & Timmermans, H. J. P. (2008). A Random Regret-Minimization model of travel choice. *Transportation Research Part B: Methodological*, *42*, 1–18. doi:10.1016/j.trb.2007.05.004

Chorus, C. G. & de Jong, G. C. (2011). Modelling experienced accessibility for utility maximizers and regret-minimizers. *Journal of Transport Geography*, *19*, 1155–1162.

Claessen, K. & Hughes, J. (2000). QuickCheck: a lightweight tool for random testing of Haskell programs. In *Icfp '00 proceedings of the fifth acm sigplan international conference on functional programming* (Vol. 35, pp. 268–279). doi:10.1145/351240.351266

Clausen, J. (1999). *Branch and Bound Algorithms -Principles and Examples*. Department of Computer Science, University of Copenhagen. doi:10.1.1.5.7475

Collins, J. E. (2002). *Solving combinatorial auctions with temporal constraints in economic agents* (Doctoral dissertation, University of Minnesota).

Collins, J. E., Gini, M., & Mobasher, B. (2002). *Multi-agent negotiation using combinatorial auctions with precedence constraints*. University of Minnesota, Department of Computer Science and Engineering.

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). *Introduction to Algorithms* (Third Edit). Cambridge, MA, USA: MIT Press.

Cramton, P., Shoham, Y., & Steinberg, R. (2006). *Combinatorial auctions*. Cambridge, MA, USA: MIT Press. doi:10.7551/mitpress/9780262033428.001.0001

Cranenburgh, S. V., Angelo, C., & Chorus, C. G. (2015). New insights on random regret minimization models. *Transportation Research Part A*, *74*, 91–109. doi:10.1016/j.tra.2015.01.008

de Vries, S. & Vohra, R. V. (2003). Combinatorial Auctions: A Survey. *INFORMS Journal on Computing*, *15*, 284–309. doi:10.1287/ijoc.15.3.284.16077

Diao, Y. (2014). *User interface of a planning and scheduling tool for oil terminal* (MSc. Thesis, Delft University of Technology).

Douma, A. M., Schuur, P. C., & Schutten, J. M. J. (2011). Aligning barge and terminal operations using service-time profiles. *Flexible Services and Manufacturing Journal*, *23*(4), 385–421. doi:10.1007/s10696-011-9080-9

Duffie, N. A. & Prabhu, V. V. (1994, January). Real-time distributed scheduling of heterarchical manufacturing systems. *Journal of Manufacturing Systems*, *13*(2), 94–107. doi:10.1016/0278-6125(94)90025-6

Dughmi, S. & Ghosh, A. (2010). Truthful Assignment without Money. In *Proceedings of the 11th acm conference on electronic commerce (ec)* (pp. 325–334). ACM. doi:10.1145/1807342.1807394. arXiv: 1001.0436

Dym, C. & Little, P. (2008). *Engineering Design: A Project Based Introduction*. Hoboken, New Jersey: Wiley.

Ergin, H. İ. (2000). Consistency in house allocation problems. *Journal of Mathematical Economics*, *34*(1), 77–97. doi:10.1016/S0304-4068(99)00038-5

Ferber, J. (1995). *Les Sistemes multi-agents: versune intelligence collective. Interedition*. InterEditions.

Finley, M. (2012). The Oil Market to 2030 — Implications for Investment and Policy. *Economics of Energy & Environmental Policy*, *1*(1), 25–36.

Fisher, M. (1985). Interactive Optimization. *Annals of Operations Research*, *5*(3), 539–556.

Fisher, M. & Wooldridge, M. (1993). Specifying and verifying distributed intelligent systems. In M. Filgueiras & L. Damas (Eds.), *Progress in artificial intelligence - sixth portuguese conference on artificial intelligence (lnai volume 727)* (pp. 13–28). Berlin, Germany: Springer-Verlag.

Fotakis, D., Krysta, P., & Ventre, C. (2013). Combinatorial Auctions without Money. In *Proceedings of the 2014 international conference on autonomous agents and multi-agent systems* (pp. 1029–1036). arXiv: 1310.0177v1

Freeman, R. E. (1984). *Strategic Management: A Stakeholder Approach*. Boston, MA: Pitman.

Fujishima, Y., Leyton-Brown, K., & Shoham, Y. (1999). Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Ijcai international joint conference on artificial intelligence* (pp. 548–553).

Gabel, T. & Riedmiller, M. (2008). Adaptive Reactive Job-Shop Scheduling With Reinforcement Learning Agents. *International Journal of Information Technology and Intelligent Computing*.

Gale, D. & Shapley, L. S. (1962). College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, *69*(1), 9–15.

Gale, D. & Sotomayor, M. (1985). Some remarks on the stable matching problem. *Discrete Applied Mathematics*, *11*, 223–232.

Gao, K., Suganthan, P., Pan, Q., Chua, T., Cai, T., & Chong, C. (2014, December). Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling. *Information Sciences*, *289*(1), 76–90. doi:10.1016/j.ins.2014.07.039

Garey, M. & Johnson, D. (1978). 'Strong' NP-Completeness Results: Motivation, Examples, and Implications. *Journal of the Association for Computing Machinery*, *25*(3), 499–508.

Garey, M., Johnson, D., & Sethi, R. (1976). The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, *1*(2), 117–129.

Gartner, J., Musliu, N., Schafhauser, W., & Slany, W. (2011). TEMPLE - A domain specific language for modeling and solving staff scheduling problems.

2011 IEEE Symposium on Computational Intelligence in Scheduling, 58–64. doi:10.1109/SCIS.2011.5976550

Geiger, M. (2006). Solving multi-objective scheduling problems - An integrated systems approach. *Artificial Intelligence in Theory and Practice*, *217*(10), 493–502.

Graham, R., Lawler, E. L., Lenstra, J. K., & Kan, a. H. G. R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. doi:10.1016/S0167-5060(08)70356-X

Graham, R., Lawler, E., Lenstra, J. K., & Kan, A. (1979). Sequencing and Scheduling: Algorithms and Complexity. In *Annals of discrete mathematics* (Chap. 9, Vol. 5, pp. 287–326). Annals of Discrete Mathematics. Elsevier. doi:10.1016/S0167-5060(08)70356-X

Guevara, C. A., Chorus, C. G., & Ben-Akiva, M. E. (2013). *Sampling of Alternatives in Random Regret Minimization Models*. Sampling of Alternatives in Random Regret Minimization Models.

Gusfield, D. & Irving, R. W. (1989). *The Stable Marriage Problem - Structure and Algorithms*. Cambridge, MA, USA: MIT Press.

Huber, M. (2001). *Tanker Operations: a handbook for the person-in-charge (PIC)*. Cambridge, MD: Cornell Maritime Press.

Hunsberger, L. & Grosz, B. (2000). A combinatorial auction for collaborative planning. In *Proceedings fourth international conference on multiagent systems*. doi:10.1109/ICMAS.2000.858447

Ingber, L. (1996). Adaptive simulated annealing (ASA): Lessons learned. *Control and Cybernetics*, *25*(1), 32–54. doi:10.1.1.15.2777. arXiv: 0001018 [cs]

ISA. (2010). *ANSI/ISA-95.00.01-2010 (IEC 62264-1 Mod) Enterprise-Control System Integration - Part 1: Models and Terminology*.

Jalilzadeh, B., Planken, L., & De Weerdt, M. (2010). Mechanism design for the online allocation of items without monetary payments. *Lecture Notes in Business Information Processing*, *59*, 74–87. doi:10.1007/978-3-642-15117-0_6

Jansen, K., Solis-Oba, R., & Sviridenko, M. (2003). Makespan Minimization in Job Shops: A Linear Time Approximation Scheme. *SIAM Journal on Discrete Mathematics*, *16*(2), 288–300. doi:10.1137/S0895480199363908

Jennings, N. & Bussmann, S. (2003). Agent-based control systems. *IEEE Control Systems Magazine*, *23*(1), 61–74.

Keeney, R. L., Raiffa, H., & Rajala, D. W. (1979). *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. New York: Cambridge University Press. doi:10.1109/TSMC.1979.4310245

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680.

Kleinberg, J. & Tardos, É. (2011). *Algorithm Design*. Boston, MA: Addison-Wesley.

Knuth, D. E. (2011). *The Art of Computer Programming - volume 2: Seminumerical Algorithms*. Boston, MA: Addison-Wesley Professional.

Koenig, S., Keskinocak, P., & Tovey, C. (2010). Progress on agent coordination with cooperative auctions. In *Twenty-fourth aaai conference on artificial intelligence*.

Kouider, A. & Bouzouia, B. (2012, January). Multi-agent job shop scheduling system based on co-operative approach of idle time minimisation. *International Journal of Production Research*, *50*(2), 409–424. doi:10.1080/00207543.2010.539276

Kramer, A. (1995). *Scheduling preemptive multiprocessor tasks on dedicated processors* (Doctoral dissertation). doi:10.1016/0166-5316(94)90058-2

Krysta, P. & Ventre, C. (2010). Combinatorial auctions with verification are tractable. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6347 LNCS*(PART 2), 39–50. doi:10.1007/978-3-642-15781-3_4

Lavi, R., Mu'alem, A., & Nisan, N. (2003). Towards a characterization of truthful combinatorial auctions. *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* 1–60. doi:10.1109/SFCS.2003.1238230

Ledesma, S., Avina, G., & Sanchez, R. (2008). Practical Considerations for Simulated Annealing Implementation. In C. M. Tan (Ed.), *Simulated annealing* (Chap. 20, pp. 401–420). Vienna, Austria: In-Teh.

Leffler, W. (2008). *Petroleum Refining in Nontechnical Language* (Fourth Edi). Tulsa, Oklahoma: PennWell Books.

Lehmann, D., Muller, R., & Sandholm, T. (2006). The Winner Determination Problem. In P. Cramton, Y. Shoham, & R. Steinberg (Eds.), *Combinatorial auctions* (Chap. 12, pp. 297–317). Cambridge, MA, USA: MIT Press.

Leskovec, J., Rajaraman, A., & Ullman, J. (2014). *Mining of Massive Datasets*. Cambridge University Press.

Ligteringen, H. & Velsink, H. (2012). *Ports and Terminals*. Delft, The Netherlands: VSSD.

Ling, R. F. (1974). Comparison of Several Algorithms for Computing Sample Means and Variances. *Journal of the American Statistical Association*, *69*(348), 859–866. doi:10.2307/2286154

Little, J. D. C., Mury, K. G., Sweeney, D. w., & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research*, *11*(6), 972–989.

Liu, N., Abdelrahman, M. a., & Ramaswamy, S. (2005). Robust and adaptable job shop scheduling using multiple agents. In *Proceedings of the thirty-seventh southeastern symposium on system theory, 2005. ssst '05.* (pp. 77–81). Cookeville, TN: Ieee. doi:10.1109/SSST.2005.1460875

Liu, N., Abdelrahman, M. a., & Ramaswamy, S. (2007). A complete multiagent framework for robust and adaptable dynamic job shop scheduling. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, *37*(5), 904–916. doi:10.1109/TSMCC.2007.900658

Loertscher, S. & Marx, L. M. (2014). *Can Auctioneers Effectively Favor Selected Bidders ? The Impact of Resale on Auctions with Bid Credits*.

Logenthiran, T., Srinivasan, D., Khambadkone, A. M., & Aung, H. N. (2010, December). Multi-Agent System (MAS) for short-term generation scheduling of a microgrid. In *2010 ieee international conference on sustainable energy*

technologies (icset) (pp. 1–6). Kandy, Sri Lanka: Ieee. doi:10.1109/ICSET.2010.5684943

Madureira, A., Santos, J., Gomes, N., & Ferreira, I. (2007). Developing a Multi-Agent System for Dynamic Scheduling through AOSE Perspective. In K. Elleithy (Ed.), *Advances and innovations in systems, computing sciences and software engineering*. Springer Netherlands.

McKay, K. N., Buzacott, J. A., & Safayeni, F. R. (1989). The Scheduler's Knowledge of Uncertainty: The Missing Link. In: Browne J.(Ed.) Knowledge Based Production Management Systems. In *The ifip wg5.7 working conference on knowledge based production management systems*. Elsevier, Amsterdam.

Mendivil, F., Shonkwiler, R., & Spruill, M. (2001). Restarting Search Algorithms with Applications to Simulated Annealing. *Advances in Applied Probability*, *33*(1), 242–259.

Muller, R. (2006). Tractable Cases of the Winner Determination Problem. In P. Cramton, Y. Shoham, & R. Steinberg (Eds.), *Combinatorial auctions* (pp. 319–336). Cambridge, MA, USA.

Nicholas, B. (2012). The relevance of efficiency to different theories of society. In *Economics of the welfare state* (5th ed., p. 46). Oxford University Press.

Nijdam, M. & Romochkina, I. (2012). *Cassandra - D6.1 - Final - Stakeholder Analysis*. CASSANDRA.

Nisan, N. (2000). Bidding and Allocation in Combinatorial Auctions. In *Proceedings of the 2nd acm conference on electronic commerce* (pp. 1–12).

Nisan, N. (2006). Bidding Languages for Combinatorial Auctions. In P. Cramton, Y. Shoham, & R. Steinberg (Eds.), *Combinatorial auctions* (Chap. 12, pp. 297–317). Cambridge, MA, USA: MIT Press.

Nisan, N., Roughgarden, T., Tardos, É., & Vazirani, V. V. (2010). *Algorithmic game theory*. doi:10.1145/1785414.1785439. arXiv: 0907.4385

Otten, R. & Ginneken, L. V. (1988). Stop criteria in simulated annealing. *Proceedings 1988 IEEE International Conference on Computer Design: VLSI*, (1), 5–8. doi:10.1109/ICCD.1988.25760

Owen, A. B. (2013). Variance Reduction. In *Monte carlo theory, methods and examples*.

Parkes, D. C. (1999a). iBundle: An Efficient Ascending Price Bundle Auction. In *In acm conference on electronic commerce (ec-99)* (pp. 148–157).

Parkes, D. C. (1999b). iBundle : An Iterative Combinatorial Auction. In *1st acm conf. on electronic commerct (ec'99)* (pp. 148–157).

Pinedo, M. L. (2012). *Scheduling : Theory, Algorithms, and Systems*.

Pinedo, M. L. & Chao, X. (1998). *Operations Scheduling with applications in manufacturing and services*. New York, NY, USA: McGraw-Hill.

Punnen, A. P. (1991). A linear time algorithm for the maximum capacity path problem. *European Journal of Operational Research*, *53*, 402–404. doi:10.1016/0377-2217(91)90073-5

Qin, L. & Li, Q. (2012). A New Construction of Job-Shop Scheduling System Integrating ILOG and MAS. *Journal of Software*, *7*(2), 269–276. doi:10.4304/jsw.7.2.269-276

Rabelo, R., Camarinha-Matos, L., & Afsarmanesh, H. (1999). Multi-agent-based agile scheduling. *Robotics and Autonomous Systems*, *27*(1-2), 15–28. doi:10.1016/S0921-8890(98)00080-3

Reed, M., Graves, A., Dandy, N., Posthumus, H., Hubacek, K., Morris, J., & Stringer, L. (2009). Who's in and why? A typology of stakeholder analysis methods for natural resource management. *Journal of environmental management*, *90*(5), 1933–1949.

Reed, R. & Marks, R. (1999). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Cambridge, MA, USA: MIT Press.

Richter, J. (1993). Boeing Divisions Collaborate on Development. Implementation of Blackboard Technology on the Shop Floor. *FOCUS: Exceeding Partner Expectations*, 2–3.

Rollon, E. & Larrosa, J. (2009). Constraint Optimization Techniques for Multi-Objective Branch and Bound Search. In *Multiobjective programming and goal programming* (pp. 89–98). Berlin, Heidelberg: Springer Berlin Heidelberg.

Rothkopf, M. H., Pekeč, A., & Harstad, R. M. (1998). Computationally Manageable Combinational Auctions. *Management Science*, *44*(8), 1131–1147.

Russel, S. & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* (3rd) (M. Hirsch, T. Dunkelberger, & M. Haggerty, Eds.). Upper Saddle River: Pearson Education.

Sandholm, T. (2000). Optimal Winner Determination Algorithms. In P. Cramton, Y. Shoham, & R. Steinberg (Eds.), *Combinatorial auctions* (Chap. 14, pp. 1–49). Cambridge, MA, USA: MIT Press.

Sandholm, T. (2002a). Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, *135*(1-2), 1–54. doi:10.1016/S0004-3702(01)00159-X

Sandholm, T. (2002b). Distributed Rational Decision Making. In G. Weiss (Ed.), *Multiagent systems - a modern approach to distributed artificial intelligence* (Chap. 5, pp. 201–258). Cambridge, MA, USA: MIT Press.

Sandholm, T., Sandholm, T., Suri, S., Suri, S., Gilpin, A., Gilpin, A., … Levine, D. (2002). Winner determination in combinatorial auction generalizations. In *Proceedings of the first international joint conference on autonomous agents and multiagent systems part 1 - aamas '02* (pp. 69–76). doi:10.1145/544741.544760

Sandholm, T. & Suri, S. (2003). BOB: Improved Winner Determination in Combinatorial Auctions and Generalizations. *Artificial Intelligence*, *145*, 33–58.

Sargent, R. (2011). Verification and validation of simulation models. In *Proceedings of the 2011 winter simulation conference* (pp. 183–198). doi:http://doi.acm.org/10.1145/1162708.1162736

Sasaki, G. H. & Haje, B. (1988). The time complexity of maximum matching by simulated annealing. *Journal of the ACM*, *35*, 387–403.

Schummer, J. & Vohra, R. (2007). Mechanism design without money. In N. Nisan, T. Roughgarden, E. Tardos, & V. Vazirani (Eds.), *Algorithmic game theory* (Chap. 10). New York, New York, USA: Cambridge University Press.

Shehory, O. & Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, *101*(1-2), 165–200. doi:10.1016/S0004-3702(98)00045-9

Shoham, Y. & Leyton-Brown, K. (2009). *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations*. New York: Cambridge University Press.

Sipser, M. (2006). *Introduction to the Theory of Computation* (2nd) (M. Mendelsohn, Ed.). Boston, MA: Thomson Course Technology.

Smith, R. (1977). The CONTRACT NET: a formalism for the control of distributed problem solving. In *Proceedings of the 5th international joint conference on artificial intelligenc (ijcai-77)*. Cambridge, MA, USA.

Smith, R. (1980a). *A Framework for Distributed Problem Solving*. UMI Research Press.

Smith, R. (1980b). The contract net protocol. *IEEE Transactions on Computers*, *29*(12).

Sobernig, S., Strembeck, M., & Beck, A. (2013). Developing a domain-specific language for scheduling in the European energy sector. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8225 LNCS*, 19–35. doi:10.1007/978-3-319-02654-1_2

Sourd, F. & Spanjaard, O. (2008). A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem. *INFORMS Journal on Computing*, *20*(3), 472–484. doi:10.1287/ijoc.l070.0260

Tan, C. M. (2012). *Simulated annealing*. Vienna, Austria: In-Teh. doi:10.1007/978-3-540-92910-9_49

TankTerminals.com. (2014). Tank Terminals TankView. Retrieved February 10, 2015, from https://www.tankterminals.com/tankfinder%5C_open.php

Tripathi, A. (2014). Multi Agent System in Job Shop Scheduling using Contract Net Protocol. *International Journal of Computer Applications*, *94*(16), 24–29.

Umang, N., Bierlaire, M., & Vacca, I. (2011). *The berth allocation problem in bulk ports*. TRANSP-OR.

United Nations Conference on Trade and Development. (2010). *Review of Maritime Transport*. United Nations. United Nations.

Van Dam, K., Nikolic, I., & Lukszo, Z. (2013). *Agent-Based Modelling of Socia-Technical Systems*. doi:10.1007/978-94-007-4933-7

van Laarhoven, P. J. & Aarts, E. H. (1987). *Simulated Annealing: Theory and Applications*. Dordrecht, NL: Springer Netherlands.

van Wezel, W. & Jorna, R. J. (1999, July). The SEC-system reuse: support for scheduling system development. *Decision Support Systems*, *26*(1), 67–87. doi:10.1016/S0167-9236(99)00018-4

Verbraeck, A. (1991). *Developing an adaptive scheduling support environment*. Delft University of Technology.

Verheul, B. (2010). *Performance improvement of liquid bulk terminals An application of the OEE concept for liquid bulk terminals*. Tebodin Netherlands B.V.

Wang, Z. & Liu, Y. (2006). A multi-agent agile scheduling system for job-shop problem. In *Proceedings of the isda 2006: sixth international conference on intelli-*

gent systems design and applications (Vol. 2, pp. 679–683). doi:10.1109/ISDA.2006.253918

Weiss, G. (1999). *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence* (G. Weiss, Ed.). Cambridge, MA, USA: MIT Press.

White, S. (1984). Concepts of Scale in Sumilated Annealing. In *Proceedings of the ieee int. conference on computer design* (pp. 646–651). Port Chester.

Wooldridge, M. (2002). *An Introduction to Multiagent Systems*. Chichester, West Sussex: John Wiley & Sons Ltd.

Wooldridge, M. & Jennings, N. (1995). Intelligent agents: Theory and practice. *Knowledge engineering Review*, *10*(2), 115–152.

Wooldridge, M., Jennings, N., & Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, *3*(3), 285–312. doi:10.1023/A:1010071910869. arXiv: 958961.958963 [10.1145]

Yen, J. Y. (1971). Finding the K Shortest Loopless Paths in a Network. *Management Science*, *17*(11), 712–716. doi:10.1287/mnsc.17.11.712

Zhou, R., Nee, A., & Lee, H. (2009). Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problems. *International Journal of Production Research*, *47*(11), 2903–2920. doi:10.1080/00207540701644219

Zinkevich, M., Blum, A., & Sandholm, T. (2003). On Polynomial-Time Preference Elicitation with Value Queries. In *Proceedings of the 4th acm conference on electronic commerce* (pp. 176–185).

Appendix A: Operations at Liquid Bulk Terminals

This appendix contains detailed information on the operational steps at liquid bulk terminals.

Arriving / Pilotage In this step, the vessel approaches the terminal and should be guided to a berth. After this step is complete, the vessel is securely attached to a berth and the final preparations for pumping can start.

1. **Vessel arrival at buoy notification:** The vessel notifies the terminal that it is soon to arrive;
2. **Notice of vessel readiness:** The vessel informs the terminal that it has reached the Pilot Boarding Station and is ready to start navigating to berth;
3. **Terminal ready notification:** The terminal informs the vessel that it is ready to process the vessel;
4. **Pilot on Board:** A pilot, needed for securely guiding the vessel through the port, has boarded the vessel;
5. **Manoeuvring:** The vessel is manoeuvring to the berth. During this time, the immediate surrounding berths cannot be approached or departed from;
6. **First line ashore:** The first line securing the vessel to the berth is attached;
7. **All fast:** The vessel is now secured to the berth;
8. **Gangway ready:** Crew members can now enter and exit the vessel.

Prepumping / Pre-Operations In the Prepumping step, the vessel and product are inspected and the vessel is connected to the terminal in order to facilitate pumping.

1. Safety inspector on board;
2. Start safety inspection;
3. Completed safety inspection;
4. Surveyor on board;
5. *Marine Loading Arm* (MLA) connection;
6. line-up;
7. line packing;

Pumping The pumping step refers to the actual transfer of the product.

1. Start product flow;
2. Flushing;
3. (Optional) Intermediate stop product flow;
4. (Optional) Intermediate start product flow;
5. Complete product flow;
6. MLA stripping.

Postpumping / Post-Operations In the postpumping step, the vessel is disconnected from the terminal infrastructure in order to facilitate the departure of the vessel.

1. MLA disconnection;
2. No dispute received;
3. Vessel certification returned;
4. Port clearance received;
5. Pilot on board for departure;
6. All clear;
7. Departure.

Appendix B: Operations timing

In order to use provided data in the scheduling system, probability functions have been fitted. This is usually preferred over using the data directly, but only if a good probability function can be fitted on the data.

All fitted distributions are checked using a *Kolmogorov-Smirnov* (KS) test. A KS test is preferred over the more common χ^2 test since it is an exact test. However, it only applies to continuous distributions. If the P-Value of the KS test is greater than 0.05, the distribution is considered to fit well. The higher the value, the better the fit. The fitted distributions with KS statistics can be found in table 1.

Table 1: Operations timings (in minutes)

	Operation	Expression	R^2	Kolmogorov-Smirnov	
				Test Statistic	P-Value
DVE	NoR - All Fast				
	All Fast - Pumping				
	Pumping - Gangway off				
	Gangway off - departure				
LVE	NoR - All Fast				
	All Fast - Pumping				
	Pumping - Gangway off				
	Gangway off - departure				
DBA	NoR - All Fast				
	All Fast - Pumping				
	Pumping - Gangway off				
	Gangway off - departure				
LBA	NoR - All Fast				
	All Fast - Pumping				
	Pumping - Gangway off				
	Gangway off - departure				

Appendix C: Terminal Infrastructure

This appendix describes in detail the different types of infrastructure elements found at liquid bulk terminals as well as their role and impact in the scheduling problem.

Tanks and Tank Lines

Tanks are used for medium to long-term storage of product. Tanks typically hold only one product and are therefore dedicated. Tanks have a maximum capacity constrained by their physical size and have a maximum loading rate depending on whether the roof is fixed or floating.

Tank lines are used to connect the tanks to the different headers in te pump platform. Their capacity is mostly determined by their dimensions. A tank is typically connected to multiple tank lines.

Quays, Jetties, and Berths

A Berth is a location in a harbour or port that is designated for mooring vessels, usually for the purposes of loading and unloading the vessel. Most berths are located alongside quays and jetties and are often specific for the various types and sizes of vessels. Quays and jetties differ from each other in their placement. Quays are used to denote structures on the shore of a harbour, while jetties are used to denote structures that project out from the land into the water. In several cases terminals distinguish main berths and wing berths. These berths can take either 1 big vessel at main berth or 2 small vessels at wing berths (see figure 2).

Figure 1 shows an example of a port with several berthing location on both the quay and a jetty.

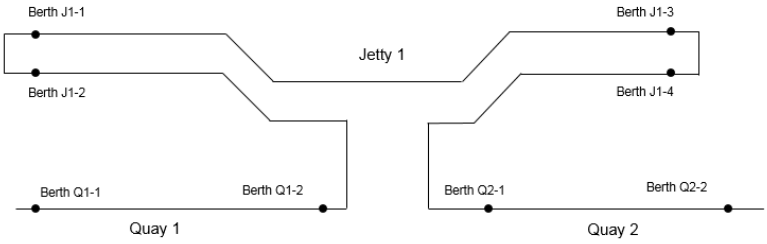


Figure 1: Jetty and Quays with Berths

There are several restrictions which need to be taken into account when scheduling berth operations. First, berths typically have a limited vessel size which they can cater. Furthermore, berthing large vessels can block neighbouring berths as shown

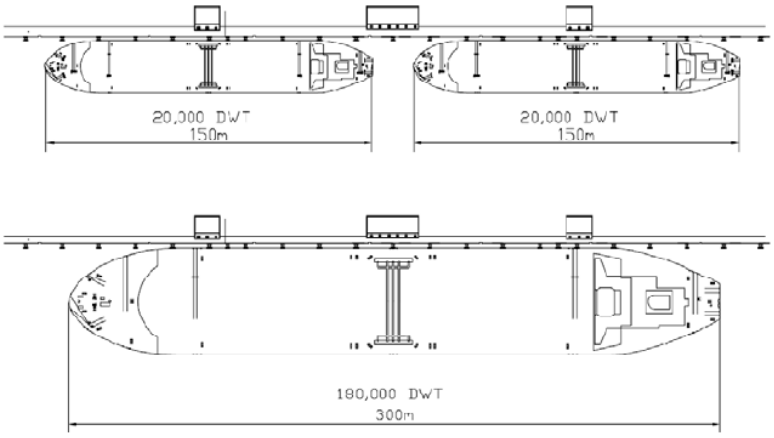


Figure 2: Wing Berths.

in the wing berth example. Third, the water depth around specific berths might be insufficient for large vessels to manoeuvre. Finally, manoeuvring at berths can only take place when nearby vessels are all fast.

Connection Lines
Connection lines are used to connect different pumping platforms to each other.

Pumps and pump headers
A pump is connected to two headers: A suction header and a pressure header. Liquid bulk is sucked through the suction header of the pump and leaves the pump using the pressure headers. Most headers are single directional but in some cases bidirectional pump headers are used. Most pumps have a short-circuit line in order to bypass the pumps for transporting product in the other direction. Pump headers can also be used to connect the different headers in a pump platform, or to connect a pump platform to jetty lines.

When loading vessels (transporting product from tank to vessel), land-side pumps on the terminal are used. Discharging vessels is often done through pumps available on the vessel itself.

Headers
The lines that connect pump headers to tanks are called headers. A group of headers together make up a pumping platform.

Jetty Lines
Jetty lines are used to connect to the loading arms at berthing locations

Loading Arms
A loading arm permits the transfer of product from the lines to the *Mode of Transport* (MOT). Loading arms specific for loading and discharging vessels are referred to as *Marine Loading Arms* (MLAs). A berth often features several loading arms, as to increase the loading and discharging capacity of that berth.

Valves
Valves are used to regulate the flow of product through the various infrastructure elements. For this application, we can assume that every single piece of infrastructure is connected to a valve at all of its connection points.

Route
A route is used to refer to a set of infrastructure which connect a berth to a berth, a tank to a tank, or a tank to a berth.

Vapour Return Systems
A *Vapour Return System* (VRS) (or vapour recovery system) is a system for recovering the vapours of gasoline and other petrochemical products in order for them to not escape into the atmosphere. Vapour return systems have a limited capacity expressed in $m^3/hour$ vapour they can process. A single VRS can process vapour from multiple routes simultaneously.

Appendix D: System Environment

This appendix will briefly explore the context in which the scheduling solution is to be applied. It is divided into two parts. First, the position of the scheduling support system is analysed based on the ISA-95 standard, a standard consisting of models and terminology for facilitating office and production automation. The second part will go in detail on the individual components making up the scheduling support system.

The main goal of a scheduling support system is to assist the operational planners with their daily tasks. Earlier work by Diao (2014, p. 85) shows that in the case of liquid bulk scheduling the most important daily tasks are receiving nominations, creating plans, communicating plans, and receiving and processing updates.

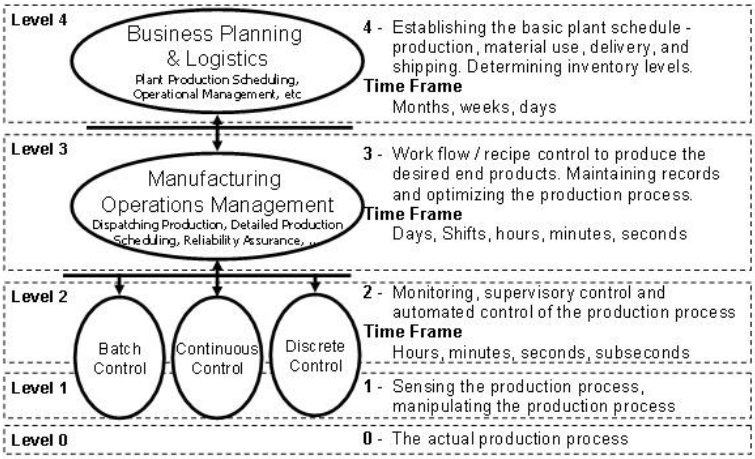


Figure 3: ISA-95 hierarchy model (ISA, 2010)

Scheduling support systems are typically designed to operate in level 3 of the ISA-95 control hierarchy level (ISA, 2010) (figure 3). The time frame of operations on this level typically varies from days to seconds, depending on the type of facility. In liquid bulk terminals a typical time frame would range from weeks to hours. Level 3 consists of a number of functions which together make up a Manufacturing Execution System (MES) (Visualized within the grey-dotted border in Figure 4). The functions which are typically represented in a scheduling support system are the production scheduling, material and energy control, maintenance management, and product inventory control.

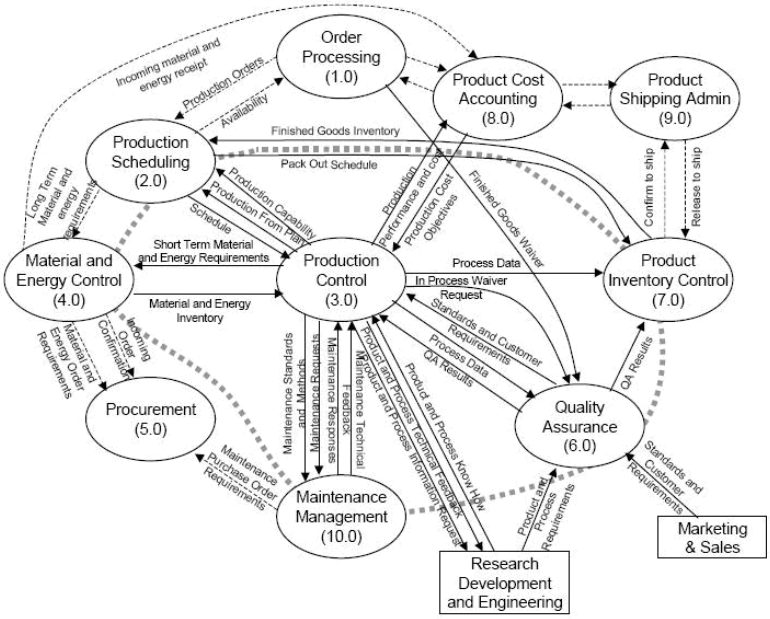


Figure 4: ISA-95 functional model (ISA, 2010)

Scheduling support systems typically consist of a number of components: A database, a data collection system, a graphical user interface, and an optimization engine. Apart from those, several additional features are sometimes available such as collaboration and communication components, experimentation functionality, and performance evaluation metrics.

The main ISA-95 functionalities a scheduling support systems needs to interface with are the order processing functionality, product inventory control functionality, and production control functionality. Several information systems are available in the liquid bulk and maritime sector that can act as information systems for these processes. Figure 5 shows a overview of the scheduling support system, its components, and its environment

Automatic Identification System (AIS) is an automated ship tracking system which can be used for identifying and locating vessels in real-time. There are several AIS data providers such as MarineTraffic which have large vessel databases including location, destination and ETA at destination. A scheduling support system can interface with AIS in order to get accurate ETA's of incoming vessels and notify the planners when the ETA starts to deviate largely from the planned ETA.

Modern terminals typically have terminal automation systems for controlling and monitoring tank levels, infrastructure status and power consumption. Typical suppliers of such equipment are Emerson and HMT. Although typically only modern

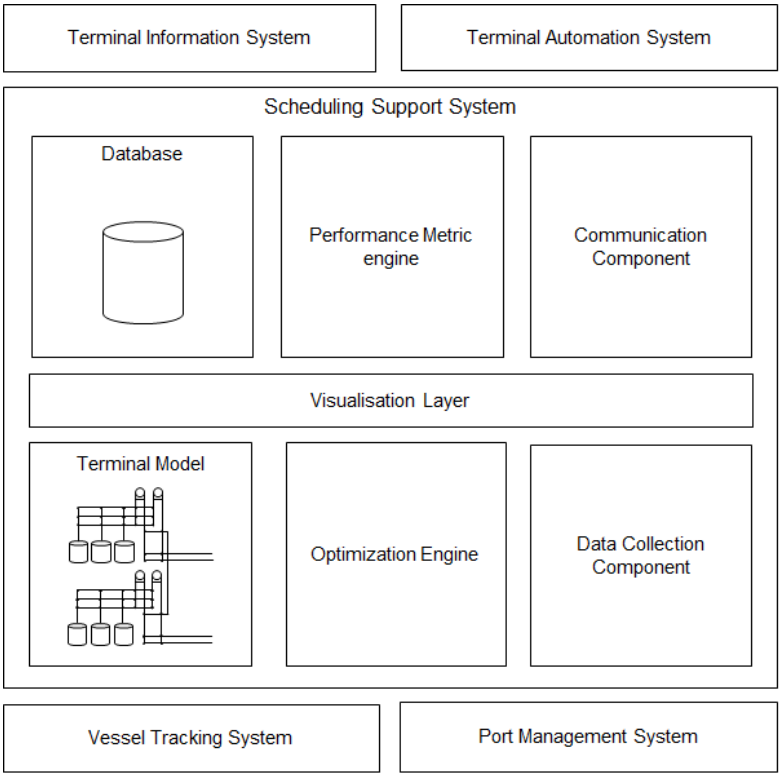


Figure 5: The scheduling support system environment

terminals have access to terminal automation systems, terminal information systems such as Tomcat are more ubiquitous. Terminal information systems capture almost all operations at terminals.

Apart from terminal specific information systems there are harbours which require the use of harbour-wide information systems such as PortBase. These port management systems are typically used for booking arrival windows and monitoring and updating customs status.

The user interface is an essential component of scheduling support systems and it is important that it supports the planners with their daily tasks. Common visualization elements in scheduling include Gantt charts, dispatch lists, performance metrics, throughput diagrams, and timetables. For a full design of a user interface for a scheduling support system for liquid bulk terminals the reader is directed to (Diao, 2014)

One of the main functions of the optimization engine in a scheduling support system is supporting interactive optimization. Optimization can be considered interactive when the user interacts significantly with either the solution process or

the post-optimization analysis (Fisher, 1985). Typical functionality includes functionality such as freezing specific jobs and re-optimizing the system and combining several distinct planning alternatives. Whenever the user has made changes to the schedule, the system should perform a feasibility analysis, process propagation effects and optionally perform internal re-optimization. Facilitating interactive optimization typically requires functionality which is not commonly found in traditional scheduling algorithms.

Typical performance metrics in scheduling include the number of late jobs, the maximum tardiness of jobs, the average tardiness, the total setup time, machine idle time, average job in system time, resource usage, and resource shortage.

Appendix E: Example Terminal Definition

This appendix contains the full definition of the example terminal used throughout this thesis for experimentation and validation. The first table shows all elements at the terminal with their flow-rates and any specific parameters such as product compatibilities. The second table gives all pairwise connections between these infrastructure elements as well as whether these connections are directional or uni-directional.

Table 2: Example terminal infrastructure elements

Element	Flow rate	Attributes
Tank 1	1500	Products: Black
Tank 2	2250	Products: White
Tank 3	3000	Products: White
Tank 4	1500	Products: Black
Tank 5	2250	Products: Black
Tank 6	3000	Products: White
Tank-line 1	1500	Products: Black
Tank-line 2	2250	Products: White
Tank-line 3	3000	Products: White
Tank-line 4	1500	Products: Black
Tank-line 5	2250	Products: Black
Tank-line 6	3000	Products: White
Header 1	9999	Products: White
Header 2	9999	Products: All
Header 3	9999	Products: All
Header 4	9999	Products: Black
Header 5	9999	Products: All
Header 6	9999	Products: All
Pump 1	1500	Products: All
Pump 2	3000	Products: All
Pump 3	1500	Products: All
Pump 4	3000	Products: All
Pump-header 1-1	1500	Products: All
Pump-header 1-2	1500	Products: All
Pump-header 2-1	3000	Products: All

Continued on next page

Table 2 – *Continued*

Element	Flow rate	Attributes
Pump-header 2-2	3000	Products: All
Pump-header 3-1	1500	Products: All
Pump-header 3-2	1500	Products: All
Pump-header 4-1	3000	Products: All
Pump-header 4-2	3000	Products: All
Connection-line 1	9999	Products: All
Jetty-line 1	3000	Products: All
Jetty-line 2	3000	Products: All
Jetty-line 3	3000	Products: All
Jetty-line 4	4500	Products: All
Loading arm 1	3000	Products: All
Loading arm 2	3000	Products: All
Loading arm 3	3000	Products: All
Loading arm 4	4500	Products: All
Berth 1	3000	Products: All Max LOA: 175
Berth 2	3000	Products: All Max LOA: 175
Berth 3	3000	Products: All Max LOA: 175
Berth 4	4500	Products: All Max LOA: 250

Table 3: Example terminal infrastructure connections

Element	Flow rate	Attributes
Tank 1	Tank-line 1	✗
Tank 2	Tank-line 2	✗
Tank 3	Tank-line 3	✗
Tank 4	Tank-line 4	✗
Tank 5	Tank-line 5	✗
Tank 6	Tank-line 6	✗
Tank-line 1	Header 1	✗
Tank-line 1	Header 2	✗
Tank-line 1	Header 3	✗
Tank-line 2	Header 1	✗
Tank-line 2	Header 2	✗

Continued on next page

Table 3 – *Continued*

Source	Sink	Directional
Tank-line 2	Header 3	✗
Tank-line 3	Header 1	✗
Tank-line 3	Header 2	✗
Tank-line 3	Header 3	✗
Tank-line 4	Header 4	✗
Tank-line 4	Header 5	✗
Tank-line 4	Header 6	✗
Tank-line 5	Header 4	✗
Tank-line 5	Header 5	✗
Tank-line 5	Header 6	✗
Tank-line 6	Header 4	✗
Tank-line 6	Header 5	✗
Tank-line 6	Header 6	✗
Header 1	Pump-header 1-1	✗
Header 1	Pump-header 2-1	✗
Header 2	Pump-header 1-1	✗
Header 2	Pump-header 2-1	✗
Header 3	Pump-header 1-1	✗
Header 3	Pump-header 2-1	✗
Header 4	Pump-header 3-1	✗
Header 4	Pump-header 4-1	✗
Header 5	Pump-header 3-1	✗
Header 5	Pump-header 4-1	✗
Header 6	Pump-header 3-1	✗
Header 7	Pump-header 4-1	✗
Pump-header 1-1	Pump 1	✓
Pump-header 2-1	Pump 2	✓
Pump-header 3-1	Pump 3	✓
Pump-header 4-1	Pump 4	✓
Pump-header 1-1	Pump-header 1-2	✗
Pump-header 2-1	Pump-header 2-2	✗
Pump-header 3-1	Pump-header 3-2	✗
Pump-header 4-1	Pump-header 4-2	✗
Pump 1	Pump-header 1-2	✓
Pump 2	Pump-header 2-2	✓
Pump 3	Pump-header 3-2	✓
Pump 4	Pump-header 4-2	✓
Pump-header 1-2	Jetty-line 1	✗
Pump-header 1-2	Jetty-line 2	✗

Continued on next page

Table 3 – *Continued*

Source	Sink	Directional
Pump-header 2-2	Jetty-line 1	✗
Pump-header 2-2	Jetty-line 2	✗
Pump-header 3-2	Jetty-line 3	✗
Pump-header 3-2	Jetty-line 4	✗
Pump-header 4-2	Jetty-line 3	✗
Pump-header 4-2	Jetty-line 4	✗
Pump-header 1-2	Connection-line 1	✗
Pump-header 2-2	Connection-line 1	✗
Pump-header 3-2	Connection-line 1	✗
Pump-header 4-2	Connection-line 1	✗
Jetty-line 1	Loading arm 1	✗
Jetty-line 2	Loading arm 2	✗
Jetty-line 3	Loading arm 3	✗
Jetty-line 4	Loading arm 4	✗
Loading arm 1	Berth 1	✗
Loading arm 2	Berth 2	✗
Loading arm 3	Berth 3	✗
Loading arm 4	Berth 4	✗

Appendix F: Gaia Models

The *Multiagent System* (MAS) described in this thesis was designed using the Gaia methodology for MAS design Wooldridge et al. (2000). This methodology consists of three phases (Figure 6), in which is worked from a set of requirements to a rudimentary agent design. This design can then be further specified using traditional software engineering methodologies. This appendix contains the full Role Models and Services Model as designed for the MAS in this thesis.

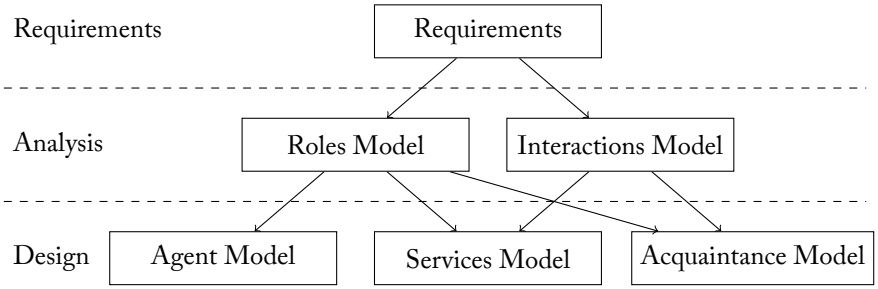


Figure 6: Gaia design steps (after Wooldridge, Jennings, & Kinny, 2000, p. 288)

Table 4: Manager role

Role: Manager
Description: The manager broadcasts new planning rounds to the vessels and receives their bids
Protocols and Activities: AwaitPlanning, BroadcastPlanning, awaitBids, AwardContracts
Permissions: Generate: AllocationList
Responsibilities: MANAGER = {AwaitPlanning, BroadcastPlanning, AwaitBids, AwardContracts}ω

Table 5: Vessel role

Role: Vessel
Description: The vessel participates in planning rounds and submits bids on required operation resources.
Protocols and Activities: AwaitTask, AskRoutes, AwaitRoutes, EvaluateRoutes
Permissions: Read: TaskDetails Generate: Bid
Responsibilities: VESSEL = {AwaitTask, AskRoutes, AwaitRoutes, EvaluateRoutes, SubmitBid}

Table 6: Route Finder role

Role: Route FinderVessel
Description: The route finder uses task details to find possible routes on the terminal
Protocols and Activities: AwaitAskRoutes, FindRoutes, SubmitRoutes
Permissions: Read: TerminalInfrastructure Generate: Routes
Responsibilities: ROUTE FINDER = {AwaitAskRoutes, FindRoutes, SubmitRoutes}ω

Table 7: Planner role

Role: Planner
Description: The planner is responsible for incorporating the terminal preferences in the allocation
Protocols and Activities: AwaitBids, FindAllocation, SubmitAllocation
Permissions: Read: TerminalPreferences Generate: AllocationOptions
Responsibilities: PLANNER = {AwaitBids, FindAllocation, SubmitAllocation}ω

Table 8: PlannerInterface role

Role: PlannerInterface
Description: The plannerinterface acts as an interface between the system and the human planner
Protocols and Activities: AwaitStartPlanning, StartPlanning, AwaitAllocationOptions, AwaitAllocationFinal, SpawnContractors
Permissions: Read: PlanningJob, FinalAllocation Generate: AllocationOptions
Responsibilities: PLANNERINTERFACE = {AwaitStartPlanning, SpawnContractors, StartPlanning, AwaitAllocationOptions, AwaitAllocationFinal}ω

Table 9: Services Model

Service	Inputs	Outputs	Pre-conditions	Post-conditions
BuildTerminalModel	terminal infrastructure	terminal model		
PreProcessModel	terminal model vessel properties product properties			
PostProcessRoutes	{route}	{route}		
AskRoutes	vessel properties product properties	{route}	Route Finder available	
FindRoutes	terminal infrastructure vessel properties product properties existing reservations	{(route,timeslot)}	Routefinder available	-
EvaluateRoutes	vessel properties product properties {(route,timeslot)}	{(route, timeslot, valuation)}	Routes ≠NIL	-
FinalAllocation	{allocation options}	finalAllocation	-	-
AskPrepareBids	planHorizon	{bids for each vessel}	Vessel available	-
FindAllocation	{bids for each vessel}	{allocation options}	Bids ≠NIL	-
EvaluateAllocation	{allocation}	allocationValue		
IsAllocationFeasible	{allocation}	boolean		

Appendix G: Set Theory Symbols

This chapter provides a small reference list of set theory and probability symbols used throughout this thesis.

Table 10: Set Theory Symbols

Symbol	Name	Definition	Example
$\{\}$	Set	Collection of elements	$A = \{3, 7, 9, 14\}$ $B = \{9, 14\}$
$\|$	such that	such that	$A = \{x \| x < 0\}$
$\|A\|$	cardinality	# elements in the set	$\|A\| = 4$
$A = B$	equality	both sets contain same elements	$\{1, 2\} = \{1, 2\}$
$A \cap B$	intersection	elements which belong to both sets	$A \cap B = \{9, 14\}$
$A \cup B$	union	Elements that belong to set A or B	$A \cup B = \{3, 7, 9, 14\}$
$A \subseteq B$	subset	subset of some or all elements of B	$\{9, 14\} \subseteq A$
$A \subset B$	strict subset	subset of some elements of B	$\{9\} \subseteq A$
$A \nsubseteq B$	not subset	not a subset of the right set	$\{9, 66\} \nsubseteq A$
$A \supseteq B$	superset	inverse subset	$\{9, 14, 28\} \supseteq A$
$A \supset B$	strict superset	A contains more elements than B	$\{9, 14\} \supseteq \{9\}$
$A \not\supseteq B$	not superset	not a superset of the right set	$B \not\supseteq \{9, 66\}$
$A \backslash B$	relative comp.	elements that belong to A but not B	$A \backslash B = \{3, 7\}$
$a \in A$	element of	single element from set	$\{9\} \in A$
$a \notin A$	not element of	not element from set	$\{88\} \notin A$
$A \times B$	cross product	all possible pairs of A and B	$\{(3, 9), \cdots, (14, 14)\}$

List of Figures

List of Tables

List of Acronyms