Agent-Based Container Terminal Optimisation

Guannan Li (Peter)

A thesis submitted for the partial fulfillment of the requirements

for the degree of

Master of Applied Science

At the University of Otago

Dunedin New Zealand

November 2010

Acknowledgements

I am particularly indebted to my principal supervisor Associate Professor Michael Winikoff for his excellent guidance throughout, and his patience in correcting every piece of code and report I submitted to him. I had a very good research year with Michael. I will never forget about this Masters' journey.

I would also like to thank my associate supervisor Associate Professor Stephen Cranefield. He gave me such a good opportunity to work this Master project. He also was the supervisor for my Honours degree in 2008.

I am grateful to Jade Software Corporation; special thanks to Roger Jarquin for his advice on this project. I would also like to thank the members of the JMT department of the Jade Software Cooperation for their valuable input data.

I would like to thank Dr. Brent Martin, Thomas Young and Hanno-felix Wagner for their contribution to this project.

In addition, I would like to thank my parent's and friend's support. Special thanks to my girlfriend Chanel for her support and patience throughout my Masters' journey.

Abstract

Container terminals have become important components in global logistics and transportation. The continuing growth of container transport is adding pressure to container terminals to improve efficiency. The problem of managing container terminals has a number of characteristics which make agents a suitable technology to consider applying. There is operation of distributed entities (e.g. quay cranes, straddle carriers) during container terminal operation; furthermore, the entities are autonomous, and coordinate to achieve competing goals in a dynamic environment.

This research is a joint industry-university project which has explored the applicability of agent technology to the domain of container terminal management. The aim of this research is to develop an agent-based solution to allocating containers to straddle carriers in order to improve efficiency and reduce the ship turnaround time.

We developed an agent-based mechanism for allocating container moves to straddle carriers. The mechanism uses a negotiation process to allocate containers to straddle carriers. We have implemented this negotiation-based approach for container management using of a Tabu Search framework (OpenTS).

The results of the research indicate that the performance of the port can be improved by an agent-based approach. In particular, an agent-based technique is a natural choice for a container terminal operation which has high level of detail and is a dynamic environment.

Table of Contents

Chapter 1: Introduction

Approximately 90% of non-bulk cargo worldwide is transported by container [1]. According to IHS Global Insight's World Trade Service, world containerised trade volumes are forecast to reach nearly 10% growth in 2010 [2]. The total number of containers shipped internationally is expected to grow to 177.6 million TEU (twenty-foot equivalent unit) by 2015 [3]. Container terminals play a critical role in global logistics and shipping since they are the interface between sea and land transport.

The continuing growth of container transport is adding pressure to container terminals to improve efficiency. Container terminal operation managers must find an efficient solution and operating strategies to improve the performance of ports. Over the last decade, container terminal operations have gained increased attention from researchers. Many approaches have been proposed for ship berthing, container handling, routing of equipment and so on. The focus of this research is to investigate how to apply agent–based technology to optimise a local port's container management and improve efficiency of the port. A higher operational efficiency can increase terminal performance and reduce congestion of containers. Increasing the productivity and operating efficiency provides the local port with a direct increase in profitability.

Measuring efficiency is extremely important to terminal operation and helps terminal operators to ensure that they are getting optimal use from their equipment. A key measure of efficiency of terminal service is the ship turnaround time. The objective of optimisation of performance is to minimise the ship turnaround time. Therefore, it is very important to make ship turnaround time as short as possible.

In order to improve ship turnaround time, one key problem is allocation of container moves to straddle carriers. There are other problems that will be discussed in Chapter 2. Allocating container moves to straddle carriers is complex, involving various constraints. For example, straddle carriers need to deal with trucks and trains that arrive. In the case of trucks, their arrivals are not predictable.

One particular challenge is that a port terminal is a dynamic environment with limited information. It is possible to solve the logistics problems when there is complete information and no change in the environment. However, the reality is that complete information is not usually available and that the environment changes in ways which are not predictable. Traditional solutions (such as operation research, or manual approaches) are insufficient to effectively handle the dynamic environment in which unloading and loading of containers takes place.

We therefore use agents to optimise container terminal operation. Agents are suited to dynamic environments because they offer an effective approach for conceptualising and developing software that is to operate in such dynamic environments, and specifically agents are able to proactively pursue goals whilst responding to changes in their environment.

According to Wooldridge and Jennings [4, page 34] "An agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives". Agents are defined as having the following characteristics: being situated in an environment; being autonomous; being able to proactively pursue goals; being reactive in responding to changes; and being social, i.e. interacting with other agents. A multi-agent system (MAS) consists of a number of agents which interact with each other. Agent-based systems are widely used in different applications such as air traffic, manufacturing and robotic control [5, 6].

1.1 Aim and outline of the thesis

A container terminal is a complex system with dynamic interactions between the different types of machines. This thesis aims to develop an agent-based solution to allocating containers to straddle carriers in order to improve efficiency and reduce the ship turnaround time. Our research question is how agents can be used to deal with container move allocation in a dynamic environment. Three steps of our approach are:

1) Design algorithm: the first proposed step is to design an algorithm to allocate containers to straddle carriers.

2) Implement: the second proposed step of this approach is to develop a negotiation-based approach for container allocation to straddle carriers, and then using a Tabu Search framework (OpenTS[1]) to implement the approach.

3) Evaluate: the last proposed step of this research is to use real data from the local port to evaluate the proposed solution.

The thesis is organized as follows. Chapter 2 outlines the details of container terminal operation and presents the relevant aspects of agents as well as an introduction to Tabu Search. Chapter 3 is a review of related literature, particularly focusing on research using agent-based approaches. Chapter 4 describes the proposed solution and a brief description of its implementation. Chapter 5 presents our evaluation which describes the experiment and its results. The last chapter summarises the thesis' contribution and makes concluding remarks as well as recommendations for future research.

[1] http://www.coin-or.org/Ots/index.html

Chapter 2: Research Background

This chapter presents an overview of some of the basic concepts needed to understand this research. Section 2.1 outlines the domain of container terminal operation. After the overview of container terminal operation, Section 2.2 describes the architecture of agents, and argues why software agents' characteristics are a good match for the domain of container terminal operation. The last section (Section 2.3) gives an overview of Tabu Search, which we use to implement our algorithm.

2.1 Overview of container terminal operation

A container terminal is an interface between ships and land-based transport. As shown in Figure 2.1, a container terminal can be described as having two main functions, quayside and landside. Quayside activities involve using quay cranes to load and unload containers to and from ships, while landside activities deal with loading and unloading containers on or off external trucks and trains. First we introduce the container terminal domain before we describe the container terminal operation.

Figure 2.1 Container Terminal

2.1.1 Containers

The advantages of containerized cargo are that they can be loaded or unloaded in a short time and that they can help protect cargo against accidental damage. Containers are steel boxes of a standard size. Volumes of containers are expressed in TEU, which standard for Twenty Foot Equivalent Unit. There are three different lengths, 20, 40 and 45 ft, but 20 ft and 40 ft are the most popular. The standard width of a container is 8 ft and containers are 8.6 ft high. There are special types of containers such as refrigerated containers ("reefers") which require power and are monitored. There are two types of reefers: frozen reefer and chilled reefer. The port aims to have reefers off power for no longer than 10 minutes, with a hard maximum of 60 minutes.

2.1.2 Container terminal areas

Container terminal is usually divided into four subsystems (see Figure 2.2). The four main subsystems are:

1) Apron: the apron is the area directly beside the ship. Containers are loaded and transferred between ship and apron. A quay crane (QC, see Section 2.1.3) is a machine which is assigned

to each ship for loading and unloading containers. In this subsystem, a container terminal manager needs to deal with berth allocation (i.e. allocating ships to dock positions), and quay crane scheduling (i.e. assigning quay cranes to ships). A berth has to be allocated to the ship before the arrival of the ship. The port manager has to provide enough quay cranes to serve a specific ship.

2) Transfer area: the transfer area is the area between apron and gate. In this subsystem, straddle carriers (SCs, see Section 2.1.3) pick up containers from the apron and drop them into the yard area or gate. There are a range of setups in different ports. Some ports would use Automated Guided Vehicles (AGVs), internal trains and external trucks to deliver containers to and move them from the transfer area. Other ports, such as the one we base our work on, use straddle carriers. The central problem in this subsystem is the allocation of container moves to machines (e.g. straddle carriers or AGVs).

3) Yard: the yard is the area where containers can be placed and stacked on each other in a certain pattern. The storage yard is divided into rectangular regions. Each container block has rows for storing containers in stacks. These blocks are served by yard cranes or straddle carriers which can lift containers and stack them on top of each other. The storage location is specified in terms of row, bay and tier within the block. The main problem with stacking containers on top of each other is that a container on top of another container needs to be put at another position when the bottom one is needed. The key problem in the yard is yard space allocation. This is because efficiently allocating containers to yard space can minimise the berthing times for ships and avoid double handling of containers. The yard space also can be limited which is a challenge for the port operation.

4) Truck/train: This subsystem is the interface between the port terminal and inland transportation such as trains or trucks. The containers can be loaded onto or from trains or trucks. One issue is that the arrival time of trains and trucks can be unpredictable, and that when they arrive, it is desirable to process them rapidly.

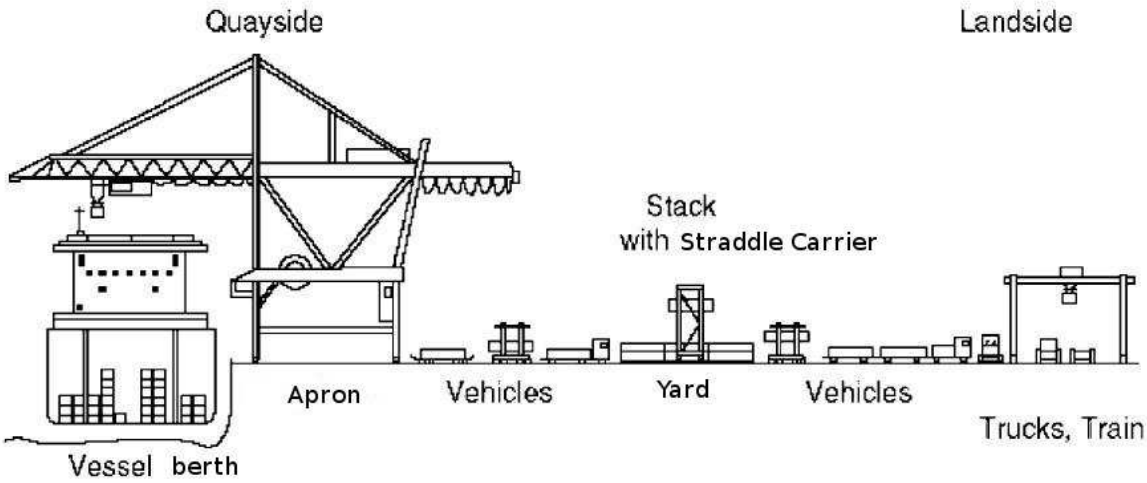

Figure 2.2 Container terminal transportation and handling system (copied and modified from Steenken et al. 2004, p 13 [7])

2.1.3 Container handling machines

Different ports have different types of machines to handle containers when ships arrive. We use the local port setup which has quay cranes and straddle carriers.

Quay cranes are allocated to a ship to discharge/load containers from/to the vessel. They are not able to move freely through the port and are limited to the crane lanes. They have a trolley which moves along the arm of cranes. Quay cranes are very large in order to handle containers from the ship. There are two types of quay cranes in the terminal operation: single-trolley cranes and dual-trolley cranes. The single trolley cranes are able to move one TEU container at a time whereas dual-trolley cranes can move up to four TEU containers. In large ports, there are normally up to 5 quay cranes to be allocated to large vessels. In the case of the local port, there are two or three quay cranes which are allocated to vessels. Port operators enforce a work rule that requires operators to maintain a safety distance between crane movements, known as clearance in order to avoid cranes clashing. Another constraint is that the apron space is limited. The time required to load and unload containers to and from ship depends on the speed of the quay crane. Figure 2.3 shows a quay crane.

Figure 2.3 An example of quay crane

(Source: http://www.flickr.com/photos/richardsinyem/1184224068/)

Straddle carriers are used to transport containers horizontally and stack containers vertically in the yard. Normally a quay crane is served by three straddle carriers when numbers of jobs are queuing up at the transfer area at local port. In the case of unloading containers, straddle carriers move the containers discharged by quay cranes from the transfer area to a target position in the yard. They then return to the transfer area and waits for the next job assignment. Figure 2.4 shows straddle carriers transferring containers within the yard area.

Figure 2.4 An example of straddle carrier

(Source: http://commons.wikimedia.org/wiki/File:Portalhubwagen.jpg)

2.1.4 Basic operation for unloading

As shown in Figure 2.5, the chain of operation for unloading containers at the local port can be described as follows: quay cranes pick up imported containers from the ship to apron, and then straddle carriers delivery them to the yard and stack them in the yards according to a stacking pattern and schedule[2]. The container loading process is the reverse of this.

[2] In some cases containers may go directly to a train or trucks, or even to another ship.

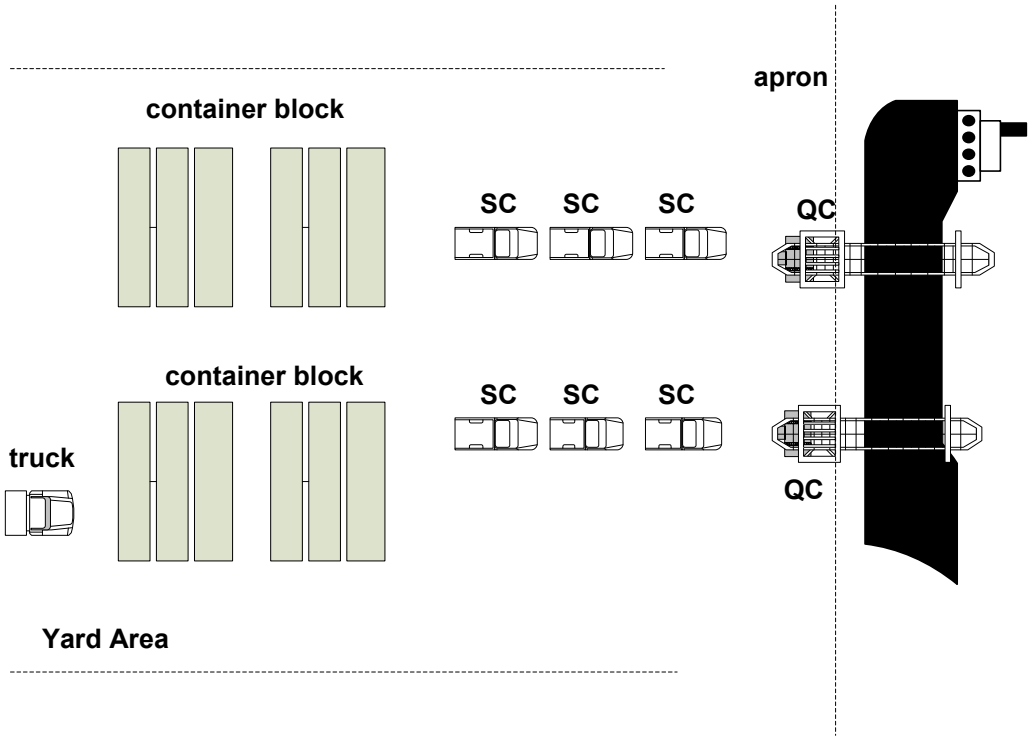

Figure 2.5 container unloading operation at the local port

2.1.5 Problem description

The performance of container terminal management is determined by a variety of inputs, outputs, and other factors and external influences [8]. Operators of container terminals are facing a big problem of how to process a set of containers in an efficient way. Port efficiency depends not only on service time, but also involves consideration of the transportation cost of containers. A port terminal tries to handle more and more containers in a short time at lower time cost. According to the interview with the operation team at the local port, the main problem of CT management is how to solve the problems of planning, scheduling and controlling. These problems have been identified as having impact on container terminal productivity. The scheduling problem is a key factor for the productivity of a container terminal, in particular, allocating container moves to machines. This is because the productivity of a container terminal is measured in terms of the productivity of quay cranes and quay cranes depend on straddle carriers to service them. Normally, an unloading and loading plan is given for quay cranes. But if straddle carriers are not managed well, then quay cranes can be idle, and waiting for containers to be provided for loading and unloading to and from ship. Hence, efficiently managing the available straddle carriers can increase the quay crane's productivity [9].

There are a range of factors and constraints that make container terminal operation complex. For instance:

- For safety reasons, humans and machines can not be in the same area at the same time (e.g. when humans are connecting reefers to power, the area cannot be accessed by straddle carriers to add or remove containers).

- The port needs to ensure that reefers are off power for no longer than 10 minutes, with a hard maximum of 60 minutes.

- Straddle carriers not only serve quay cranes, but also have to deal with trucks and trains that arrive. In the case of trucks, their arrivals are not predictable.

- In order to maintain a ship's balance, the order of loading and unloading may be constrained.

- The availability of yard space can be a serious problem for the port, especially at certain times of year.

In addition to these factors and constraints, a port terminal is a dynamic environment where things can go wrong in a range of ways. In these cases, the port manager has to deal with issues. For example, a straddle carrier may malfunction and be unable to put down a container it is carrying. The plan sometimes may fail for different reasons e.g. overstorage and equipment breakdown. Some other unexpected situations that may occur during container terminal operation include wrong placement of containers and badly stacked containers: straddle carrier operators are humans, and may make mistakes, which may lead to the yard data not being entirely reliable, i.e. containers not being where they should be.

The many decisions involved in managing ship loading and unloading are performed by humans. This is an opportunity for automated decision support.

2.2 Software agents

A software agent is a piece of software that is situated in an environment on behalf of humans to carry out different tasks. It is able to act autonomously to accomplish a task or a goal, react if the environment gets changed, interact with other agents or humans (negotiation) and be proactive in achieving its goals.

Software agents can solve problems in a dynamic environment and interact with other agents. Compared with objects, agents have a number of significant differences. Some of these are related to individual agents, such as operating autonomously, which results in agents being a good model for decentralized decision making; and being proactive, which is realized by the agent having goals. Some of these are related to multi-agent systems, such as interacting with humans and software, which have particular structure. The key problem facing an agent is which actions should be performed in order to satisfy its design objectives.

Agents' characteristics are a good match for domain of container terminal operation. Firstly, container terminal operation is distributed and autonomous, which involves interacting entities (e.g. cranes). Secondly, there are multiple goals (e.g. unloading a ship and loading a truck) and multiple constraints which make the problem being solved complex. Thirdly, the environment is dynamic, things can go wrong during container terminal operation (e.g. machines breaking down). These three characteristics make agents are candidate solution for container terminal management. We return to agents and container terminals in the next chapter.

2.3 Tabu Search

Tabu search is a heuristics method for solving optimisation problems. The idea of Tabu Search is that it applies moves to the current solution, and keeps applying moves until a termination condition is met. Each solution has a neighbourhood which is used to identify adjacent solution for Tabu Search movement. When a move is made from a current solution to a new solution, it uses the neighbourhood structure of the new solution. The Tabu Search always chooses the best move. In order to understand the algorithm of Tabu Search, the basic ingredients of Tabu Search are briefly defined as follows [10]:

- **A move** is an operation that generates a new (related) solution from the current solution.

- **The tabu list** records forbidden moves, which are referred to as Tabu moves.

- **An aspiration criterion** is a condition which can override Tabu restrictions.

The algorithm of Tabu Search is that it starts from initial solution, then creates a candidate list of neighbouring solutions and chooses the best admissible solution. If the Tabu Search is satisfied with the current solution, it will choose it as the final solution, otherwise it will update the Tabu aspiration conditions, which are a set of rules that override the Tabu conditions and ensure that the best admissible solution is accepted. Figures 2.6 and 2.7 show the algorithm of Tabu Search.

1. Set S to the initial solution;
2. Generate the neighbourhood of S, N(S);
3. Obtain the non-tabu neighbourhood;
4. Update current solution to best neighbour;
5. if current solution is good enough or termination conditions met, then return current solution;
6. Otherwise, repeat from step 2 and update S to the new solution;

Figure 2.6 the pseudocode of Tabu Search algorithm process

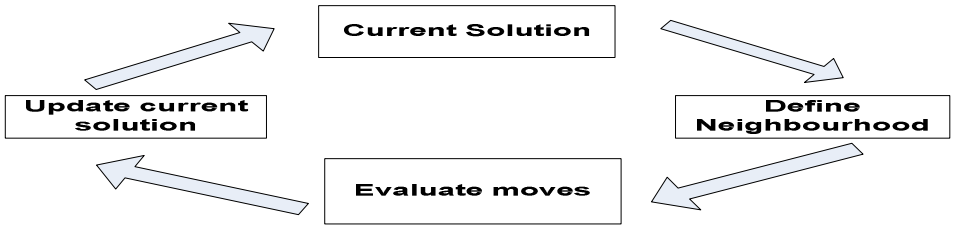

Figure 2.7 Tabu Search algorithm process

Chapter 3: Literature Review

Container terminal operations are becoming more and more important. The increasing complexity of container terminal operations requires the management to improve the performance. Container terminal operation managers try to develop different strategies that can solve the problems and improve the efficiency of port operation. Since this is an important problem, it has attracted increasing interest from researchers. This chapter provides an overview of relevant literature, which mostly focuses on the problems of loading and unloading containers to and from ships, berth allocation, machine scheduling and yard management.

This thesis focuses on agent-based container terminal management so the next Section (3.1) reviews how previous research has used agent-based technology to optimise container terminal operation. Section 3.2 reviews how non-agent-based approaches have been used to tackle various aspects of the management and optimisation of container terminal. Section 3.3 discusses the application of agents to the related domain of logistics. Finally, Section 3.4 describes more details of the Tabu Search algorithm of Chen et al. [32] upon which our work is based.

3.1 Agent-based container terminal optimisation

Chapter 2 has introduced the agent's definition and explained why agents are suitable for container terminal operation. This section reviews recent research about how to apply multi-agent systems for port terminal management.

We begin by considering a number of papers that aim to address all aspects of port management. This work is quite ambitious in scope, and as a result is often incomplete.

Thurston and Hu [11] present a distributed agent-based system that aims to automate container terminal operation. The system is controlled by four different types of agents: quay crane agent, straddle carrier agent, traffic agent and area manager agent. They focus on the

loading operation only. For a job assignment, each agent is communicating and the agents solve tasks together, but they do not provide details of mechanism which show how agents are able to communicate. They use a simulation to evaluate their system, but they also do not provide details on how their simulation method works. The evaluation is based on a job assignment for straddle carriers and how their routing can be plotted. The evaluation results are not provided on their work. The strength of this paper is that the authors designed a distributed multi-agent system which can provide a solution to improve the performance of quay cranes. The system can be evaluated by agent simulation which was developed by them. However, their work only outlines an early Java prototype. They have not done any subsequent work (from personal contact with authors).

Degano and Pellegrino [12] also develop an agent-based system that aims to control container terminal operation. The system is composed of several agent-based subsystems and a Remote Logical Data Base (RLDB). The RLDB is able to provide operation information for the system and interacts with each subsystem. Each subsystem has three functions: 1) a detection function which detects any unexpected actions 2) a diagnosis function which can identify problems 3) a decision function which can help port manager make a final decision. However, while they define these subsystems, they do not provide details on how these subsystems work. They also provide a sketch of an algorithm to help subsystems to decide which action should be performed. The algorithm provides the steps that the subsystem has to perform based on Effective Allocated Resource Set. Their paper lacks details on the algorithm. They evaluate their system based on input data from the Italian Voltri Terminal, but details are not given in the paper.

Peng and Junqing [13] design a distributed agent-based system that aims to optimise container terminal operation. There are two main agent types in their system: operation agent and resources agent. Their system uses a market based approach: it makes use of auctions where each agent can bid for a job assignment. Their research is an initial approach. An experiment evaluation is not provided. Also there are no details about how each agent can bid for a job assignment, e.g. what auction protocols are they using.

Yan et al. [14] propose to automate container terminal using a multi-agent production dispatching system. The main objectives of the system are data integration, subsystem interaction, decision making and users interface support. Their agent system has a user agent,

core agent, integrating agent and transfer agent. The user agent provides an interface between users and the system. The integrating agent is acting as the decision-maker. The core agent stores information of all agents. It collects related information from users and provides information about results. The transfer agent manages activities between yard area and quay cranes. Also they designed coordination and collaboration strategies for this system so that subsystems are able to co-operate, but the paper lacks a meaningful and useful description of this. No experimental evaluation is reported from their research.

Carrascosa et al. [15] also propose to use a multi-agent system to automate container terminal operation. They also design a multi-agent system architecture. Their architecture divides the problem into sub problems. Each sub problem can be solved by a specific agent. Agent communication is based on the FIPA standard [16]. But this paper only provides a system architecture. They do not provide details for how individual agents would operate, and there are no evaluation results.

It is difficult and ambitious to address all of problems of a container terminal. Kefi et al. [17] is an example of work that solves a part of the problem. The aim of their research is to improve the efficiency of container handling for ship departures. They developed a multi-agent architecture. There is a container agent in this architecture, which can cooperate and negotiate within a solving process. They use the JADE (Java Agent DEvelopment Framework) platform [18] to implement this proposed model with centralized and distributed version. The evaluation results show that the proposed multi-agent model can reduce unproductive movements compared with the centralized version. Their proposed model only takes into account container departures. We have not found any subsequent papers describing containers arrivals. Another potential issue is that their evaluation only considered up to 26 containers, which is quite limited compared with the number of containers that are unloaded in a real port.

Lokuge and Alahakoon [19, 20] investigate the possibilities of using intelligent software to solve the berth allocation problem. The aim of their research is to use intelligent agent software to improve decisions on berth assignments so that berth productivity can be increased. They realise that the common agent architecture of Beliefs, Desires, and Intention (BDI) is inappropriate for complex problems that must learn and adapt their behaviours. Therefore, they design a new hybrid BDI agent model that extends the BDI model with

learning and adaptability features. The feature of new hybrid BDI agent model is able to enhance the decision making process of BDI agent. Their architecture involves a berth agent, vessels agent and scheduling agent. The scheduling agent is the main part of their architecture. There are three main components in scheduling agent, which are event-handler, plan-selecter and goal evaluator. These three components are able to assist port manager to select appropriate plan for dynamic change during port terminal operation and predict the degree of success of final goal. Their evaluation is based upon data from Colombo's Jaya terminal (Sri-Lanka). The evaluation results show that the new BDI agent system framework is able to autonomously adapt to a changing environment. But they do not provide more details about their systems and experimental results. The strength of this paper is that they develop a new BDI agent framework and show the details about evaluation. However, there are some difference between our work and their research. Firstly, we focus on straddle carriers, not berth allocation. Secondly, our ultimate aim is to improve decision making in a dynamic environment without using learning agents.

Sun et al. [21] also developed a multi-agent berth allocation management system to solve the berth allocation problem. There are three main agents in their system: berth allocation agent, berth agent and vessel agent. The berth allocation agent is responsible for planning loading/unloading containers and allocation of berths. The berth agent is responsible to report the free or busy status of berths before vessels arrive and minimizing the conflict of machines. The interaction and collaboration of agents are using the Contract Net Protocol (CNP). The CNP is a high level protocol for negotiation among agents in the distributed system. It involves the initiator sending out a call for proposals. Each agent then reviews proposals and bids on the feasible ones. The initiator then chooses the best bid, gives the contract to the respective agent and rejects the other bids. They implemented their system with C# as the development language. They do not provide an experimental evaluation for their system. Their research is only in the preliminary stage.

Some other researchers focus on simulation to evaluate fixed policies but not to control a port terminal. An example is Henesey et al. [22], who develop an agent based simulator ("SimPort") for evaluating operational policies in container terminal operation. The aim of this research is not to automate the operation of container terminal operation, but instead to investigate how the simulation tool can be used to analyse the impact of different policies for sequencing arriving ships, berthing and container stacking on the performance of container

terminal operation. The simulation is based on real data from a container terminal and evaluates eight trans-shipment policies. The simulation results indicate that good choices of yard stacking and berthing position policies can improve ship turn-around time. The strength of this paper is that it developed an agent based simulation tool which can analyze which container terminal management policies can improve performance in difference scenarios. This research also found that some policies can improve ship turn-around time and lower the distance travelled by straddle carriers. Their work provides motivations for our research and evidence that scheduling matters. However, they do not provide details about their simulation tool, and we are interested in using dynamic policies in our work.

3.2 Non-Agent-based approaches

There are non-agent-based approaches that aim to solve various aspects of the container terminal optimisation problem. In general, these kinds of approaches do not address the dynamic nature of the environment. They apply different operation research methods to container terminal optimisation.

Dai et al. [23] propose a local search algorithm to solve the static berth allocation problem. Their approach can allow for determining the vessel allocation and minimizing a ship's waiting time. They have done evaluation with a moderate load and a heavy load scenario. The result from the moderate load scenario shows that this approach is able to allocate space to over 90% of vessels upon arrival. The result also shows that delaying berthing of vessels is a good way to achieve higher throughput. However, there are some limitations of their approach. They do not consider some berthing constraints while allocating space to vessels. In our research, we do not focus on berthing, because it is not really an issue for the local port (low number of ships).

Kim and Park [24] tackle the quay crane allocation problem using a heuristic search algorithm. They propose an inefficient branch & bound method to solve the QC scheduling problem, and then improve it using a Greedy Randomized Adaptive Search procedure (GRASP). They do some experiments to compare the branch and bound method to GRASP. The experiment results show that GRASP has better performance than branch & bound method. The GRASP is quite similar to Tabu Search and our negotiation process, but their work only focuses on QC scheduling, not for straddle carriers.

Canonaco et al. [25] not only tackle the QCs allocation problem, but also consider how to allocate straddle carriers to quay cranes. They propose to use a queuing network model to maximize the productivity of quay cranes and support dynamic allocation of straddle carriers to a specific quay crane. This is done by developing a queuing network model of the arrival-departure processes at container terminal. They also develop a simulation tool for tracking completion of loading and unloading operation and scheduling of QCs. They evaluate the simulation tool with different scheduling QC policies based on input data from the Gioia Tauro terminal. Their input data has 182 containers and 3 quay cranes. The evaluation result shows that the overall vessel completion time can be improved by using different policies. In one scenario, completion time is improved by 8.6% by changing the allocation of quay cranes.

Lee et al. [26] propose an optimisation model to tackle yard storage allocation problems. The aim of their research is to minimize the reshuffling and traffic congestion between the ship and the storage area. They formulate the storage allocation problems as a mixed-integer linear programming model. In order to test the performance of their model, they develop two heuristic methods which are sequential method and column generation method, and they do an evaluation to compare the two methods. The evaluation results show that the sequential method is effective at finding good solutions for low levels of utilisation, but is not able to find solutions for high utilisation. The column generation method is also able to find effective solutions for low utilisation levels, and finds solutions (typically low quality) for high utilisation levels. However, it is not clear whether they use real data, or to what extent their data is realistic.

Balev et al. [27] propose an Ant Colony Optimization (ACO) method to tackle allocation of straddle carriers. The aim of their research is develop an ant colony system which is able to provide a schedule for straddle carriers during the terminal operation. In order to test the performance of this system and compare with other current systems, they develop a simulator. There are two main parts in their simulator, which are terminal simulator and ACO system. The terminal simulator contains terminal structure. The ACO system deals with handling the dynamic events during terminal operation. They do experiments to evaluate their simulator based on 12 containers, which is quite limited compared with the number of containers that are unloaded in a real port. The simulation results show that their ant colony system can provide scheduling information to straddle carriers. However, they only have preliminary

results and have not tested all of the parameters of this system. They are collecting real input data from the seaport of Le Havre in France in order to compare the performance of their system with the current scheduling method used in port terminal.

3.3 Agent-based technology in logistics domain

We now consider the application of agent-based technology in the logistics domain. We do this because logistics is a closely related (but different) domain to container terminal operation. Logistics problems are complex and have specific properties which make them difficult to solve. There are lots of constraints that need to be fulfilled in order to carry out logistics activities. According to Davidson and Kowalczyk [28], one of the key properties of the logistics problem is uncertainty, which means that unexpected events may occur during work processing (the other key property is complexity). Pokahr et al. [29] have argued that agents can provide a good solution for logistics problems. They have examined properties of the logistics domain and properties of agents.

There are some works that have successfully applied agent to logistics domain. Dorer and Calisti [30] is an example of work that uses agent-based solution to solve dynamic transportation problems. They design a problem model which includes constraints specification and cost function definition. They also report on a real case logistics scenario in order to analysis the performance of their proposed solution. Their experiment results show that the proposed solution has better performance in transport optimisation compared with professional (human) dispatchers.

Zeddini et al. [31] develop a model based on self-organisation and multi-agent system to solve the dynamic vehicle routing problem. They develop an agent-based negotiation mechanism to allow each agent to bid for a job task by sending messages to other agents. The negotiation principle broadcasts the viability agents, accepts the bids on the agent and commits the agent to the vehicle with the lowest cost. Each client agent requests and receives from the bidder agent a cost of its insertion. After that, it enters in interaction with vehicle agent. They implemented an agent based system where agents can interact and make decision via an action negotiation process. The results show that their multi agent model can solve the dynamic vehicle routing problem though agent negotiation process. The strength of this research is that brings agents negotiation concept to solve real dynamic problems. It also

designed agents' auction mechanism to deal with bidding a job assignment. However, their research only focuses on general logistics problems, not for the vehicle routing problems of container terminal operation.

3.4 The Tabu Search algorithm of Chen et al.

In this section, we describe the Tabu Search algorithm of Chen et al. [32], since our work extends theirs. They propose a Tabu Search algorithm to deal with allocating container moves to machines. The objective of their research is to minimize the makespan (the time it takes to serve a given set of ships) in order to increase the productivity of the terminal. They develop a Tabu Search based algorithm to solve the container scheduling problem, represented as a Hybrid Flow Shop Scheduling problem with precedence and blocking constraints.

3.4.1 Assumptions

Their formulation was the starting point of our work, and our approach to optimising container moves is in some ways quite close to their work. However, their work does not deal with dynamic events during terminal operations. They also make a number of assumptions that are unreasonable in practice, such as:

- They assume that the unloading operations begin only after all the loading operations are completed, and yard blocks do not mix to-be-loaded and to-be-unloaded containers. We allow yard blocks to mix blocks that are to be loaded and blocks that are to be unloaded (however, since we only model unloading, we are not able to relax the first constraint).

- They do not provide for "buffering" where, for example, a quay crane can unload a second or third container even through the first container has not been taken to the yard (as long as space remains in the apron).

Another difference is that there are three types of machines in their model, which are quay cranes, yard cranes and yard vehicles (whereas we only have two machine types: quay cranes and straddle carriers).

3.4.2 Tabu Search algorithm

1. Solution representation

Firstly, they design a solution representation (i.e. allocation of containers to machines) in order to define their neighbourhood structure. Their solution representation is effectively a collection of machines, each with an ordered list of allocated container moves. This is different to our solution representation, in that we explicitly model and update the start and end times of each 'action' (a single container move). This is also means that adding (or removing) a container move to a machine in our approach involves updating the start and end times of subsequent container moves.

In their solution representation, they calculate the makespan based on container processing times on each machine and setup time between each container. On the other hand, since we store and update the start and end times for container moves, we calculate the cost differently (see section 4.2 for details).

2. Initial solution generation

Their overall approach has two phases: initial solution generation, and optimisation. We have the same overall approach, with initial solution generation and optimisation.

In their initial solution phase, they calculate the extra time for each container insertion when inserting containers into possible positions in the machine. The container selects a machine that has the minimum extra time.

3. Neighbourhood structure

In their optimisation phase, the neighbourhood structure is defined by two types of moves: a) moving a container from one machine to another, and b) moving a container within a machine, and these moves are filtered based on constraints. In our work, we impose some additional conditions in our optimisation phase (see section 4.2). Constraints are considered in the neighbourhood structure. For example, constraints must guarantee that each operation is processed by one machine, and ensure that the order of operation for each container is sensible (e.g. the container is unloaded from the ship before being moved to the yard).

4. Evaluation moves

There are some different between how Chen et al calculate costs and how we calculate costs. Firstly, they calculate the cost of each job during the evaluation of possible moves. Their cost calculation is based on quay cranes and approximates the cost of subsequent machines (yard vehicles and yard cranes). Secondly, they calculate the lower bound of the makespan to obtain a faster evaluation. However, our calculation of costs is not just a simplification of their calculation: because we have an unloading plan for quay cranes, we do not optimize quay cranes, only straddle carriers, so we base our calculation on the cost of straddle carriers, not quay cranes. Thus our cost calculation is different to theirs.

3.4.3 Evaluation

Their experimental evaluation is based on input data from port terminal. But it is not clear what data is provided, and how their evaluation makes use of the provided data. For example, the setup times appear to be generated rather than provided, but no information is given on the generation process, or how the times are ensured to be realistic (other than noting that they are generated in a way that satisfies the triangle inequality).

In order to evaluate their proposed solution, they develop two heuristic methods and compare them with each other: TA1 is the proposed Tabu Search with MIH_ND (schedule all containers with non-delay) as initial solution generator: TA2 is the one with MIN_MET (schedule all containers with minimum extra time) as initial solution generator. The computational results indicate that TA1 is better than TA2 for quality and efficiency of solution. The results show that a good initial solution is quite important for the container allocation problem. Their Tabu Search algorithm successfully deals with containers blocking constraints problem and provides a good solution to solve it.

Chapter 4: Agent-Based Container Management

The problems of container terminal operation have been discussed in Chapter 2. One sub-problem that affects ship turnaround time is straddle carrier management. For example, if straddle carriers are not managed well, then quay cranes have to wait for apron space to unload containers, or for containers to be provided for loading which delays ship turnaround time. In this chapter we describe an agent-based mechanism for allocating container moves to straddle carriers, specifically we focus on the unloading operation. The high-level idea is that agents can propose moves and bid for jobs during negotiation.

Section 4.2 describes the model and solution process. The model focuses on dealing with each machine's schedule along with associated time. The optimisation solution involves allocating container moves and swapping container moves to improve efficiency. This negotiation process is implemented using a Tabu Search framework (OpenTS) which is described in Section 4.3.

4.1 Solution concepts

The essence of a solution is a list of containers for each machine, where each container on a given machine has a time window. First the model is presented. Suppose there is a list of containers that need to be processed by doing number of moves (i.e. these containers must be allocated to machines). Each container has a container ID.

Let M be the set of machines that are processing a number of loading and unloading containers to and from ship. The machines have machine types which are quay cranes and straddle carriers, and a list of allowed moves which is specified in terms of location types that they can reach (e.g. yard, shore). Each machine also maintains a schedule, which is list of actions where each action in the schedule has a start and end time.

An action represents a single container move (e.g. ship to shore) and contains source and destination locations, and container ID. Figure 4.1 shows how these concepts are related.

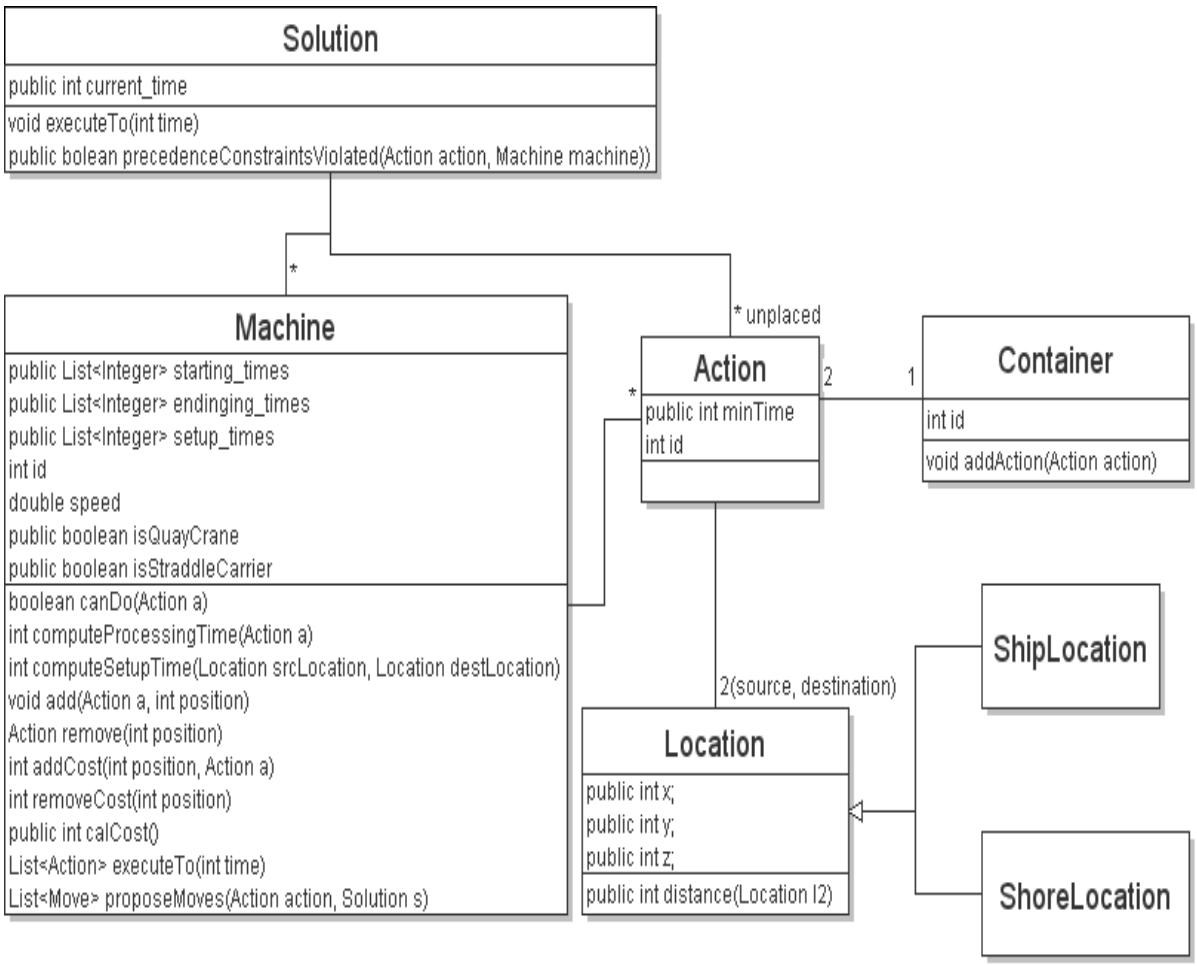

Figure 4.1 UML data model

All container operations have an associated time cost, which is derived from the distances involved. We design the time cost based on the notions of setup time and processing time. Processing time means how long to move a container from source location to destination location. Setup time is how long to get to the machine's next action. These times are depicted in Figure 4.2.

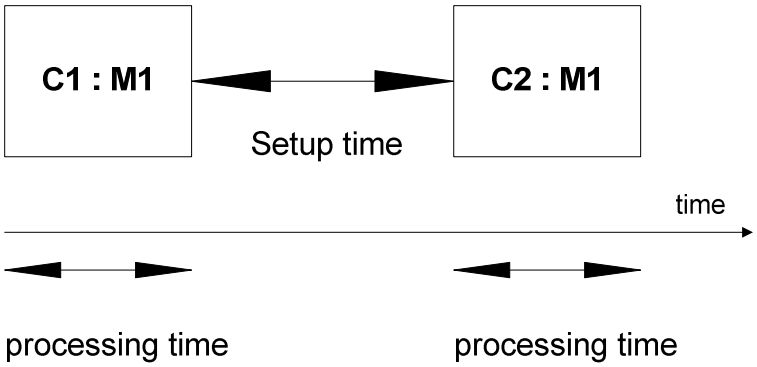

Figure 4.2 processing time and setup time

For example, in Figure 4.3, there are three actions (A1, A2 and A3) which are allocated to a machine M1 with time schedule. The setup time between each action is 3. Each action also has a start time and end time. Except for the first action, each action's start time is the sum of the previous action's end time and the setup time the between actions (however, as discussed later, this may be affected by constraints). The overall cost for the machine is the end time of the last action (A3) which is 18.

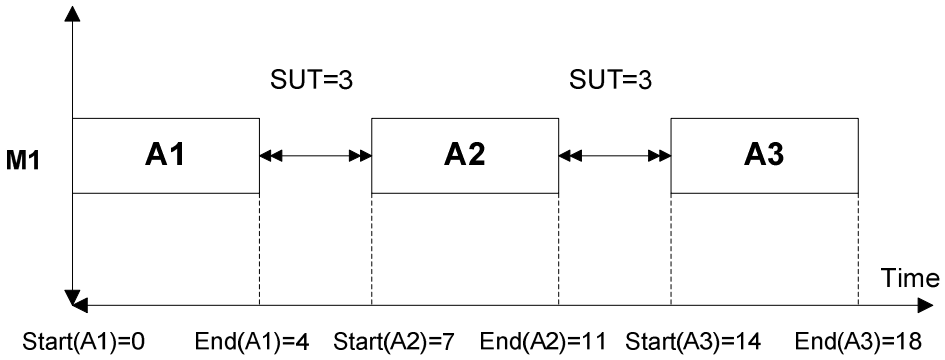

Figure 4.3 schedule time of action

In order to reduce the ship turnaround time, the overall quality of a solution is given by its cost. This is simply the completion time of the last machine to complete (i.e. the maximum of the finishing time over the machines).

We define the following notation:

C set of containers needing to be processed e.g. C1 represents the first container

M set of machines for processing containers e.g. M1 represents the first machine

A set of actions e.g. A1 represents the first action

Start (A) start time of each action

End (A) end time of each action

setupTime (A1, A2) setup time between A1 and A2

processingTime (A) processing time of action A

The container allocation problem is how to allocate the needed container moves to machines (i.e. quay cranes and straddle carriers) in a way that meets all constraints. The moves proposed must respect the various constraints, while adding/removing container moves to/from machines.

We now look at how add a container move to a machine (see Figures 4.4, 4.5 and 4.6). Suppose there are two actions (A1 and A2) which have already been allocated to machine M1. A new action (A3) is being allocated to machine M1. There are three cases for calculating the cost after the addition of A3.

Case 1: add A3 at start of machine M1 list. The new overall cost of machine is defined as: New Cost = Old Cost + processingTime (A3) + setup time (A3, A1). Setup time between A3 and A1 is added since the new action (A3) is inserted into the position of before A1.

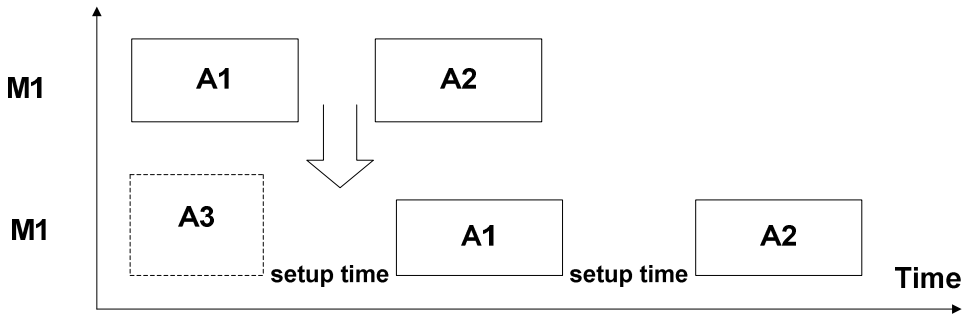

Figure 4.4 add A3 at front of machine M1

Case 2: add A3 between A1 and A2 on machine M1. The new overall cost of machine M1 is defined as: New Cost = Old Cost + processingTime (A3) + setup time (A1, A3) + setup time (A3, A2) – setup time (A1, A2). We need to add setup time (A1, A3) and setup time (A3, A2),

and subtract setup time (A1, A2) since new action (A3) is inserted into between A1 and A2. Here we subtract setup time (A1, A2) because the setup time between A1 and A2 is no longer applicable, since A2 is no longer immediately after A1.

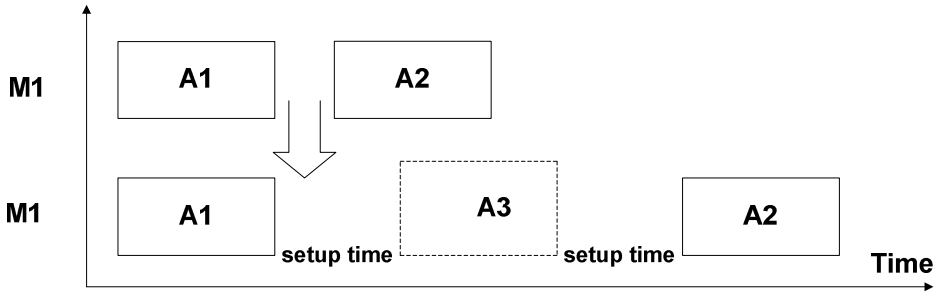

Figure 4.5 add A3 between A1 and A2 at machine M1

Case 3: add A3 at end of machine M1's list. The new overall cost of machine M1 is defined as: New Cost = Old Cost + processingTime (A3) + setup time (A2, A3). Setup time between A3 and A2 is added since new action (A3) is inserted after A2,

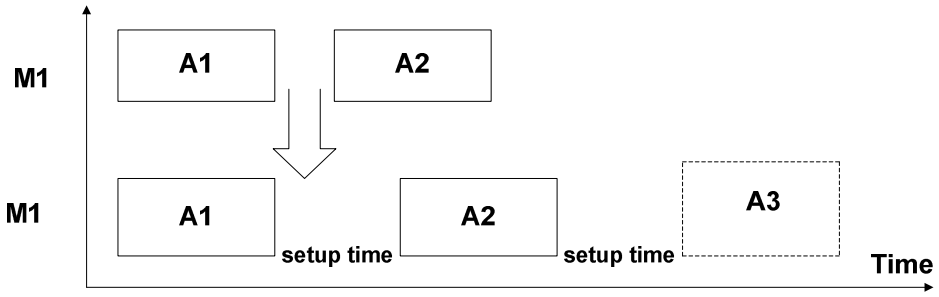

Figure 4.6 add A3 at end of machine M1

Similarly, there are also three cases for removing a container move from a machine (see Figures 4.7, 4.8 and 4.9). Suppose there are three actions (A1, A2 and A3) which are allocated to machine M1.

Case 1: remove A1 from the front of machine M1. The new overall cost of machine is defined as: New Cost = Old Cost - processingTime (A1) - setup time (A1, A2). Setup time between A1 and A2 is subtracted since we remove A1 from its position before A2.

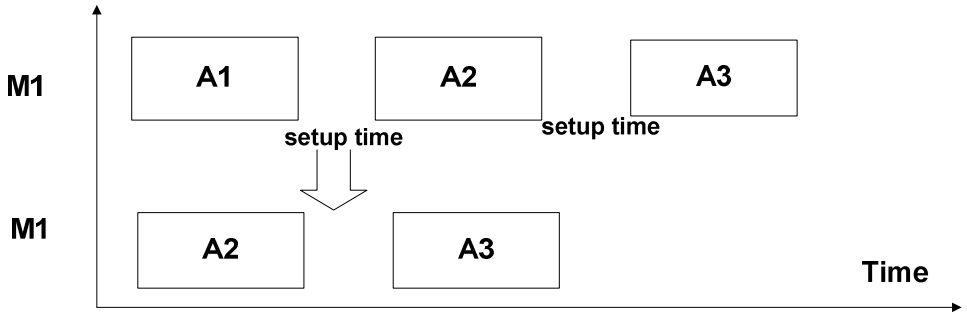

Figure 4.7 remove A1 from the front of machine M1

Case 2: remove A2 (between A1 and A3) from machine M1. The new overall cost of machine is defined as: New Cost = Old Cost- processingTime(A2) - setup time (A1, A2) - setup time (A2, A3) + setup time (A1, A3). We need to subtract setup time (A1, A2) and setup time (A2, A3), since we remove A2 from machine M1 between A1 and A3. On the other hand, because A3 now follows A1, we need to add the setup time from A1 to A3.

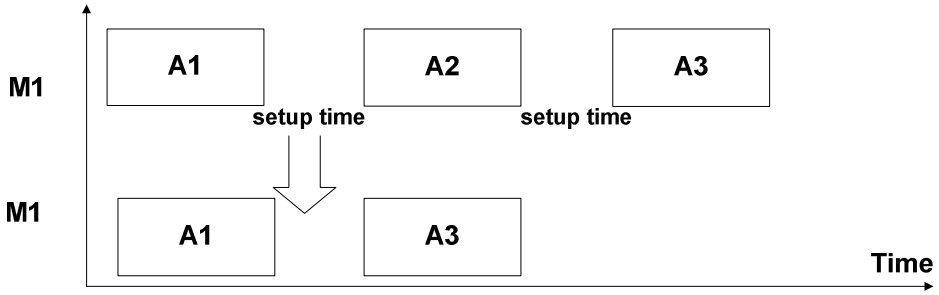

Figure 4.8 remove A2 from machine M1

Case 3: remove A3 from the end of machine M1 list. The new overall cost of machine is defined as: New Cost = Old Cost – processingTime(A3) - setup time (A2, A3). Setup time between A2 and A3 is subtracted since we remove A3 from its position after A2.

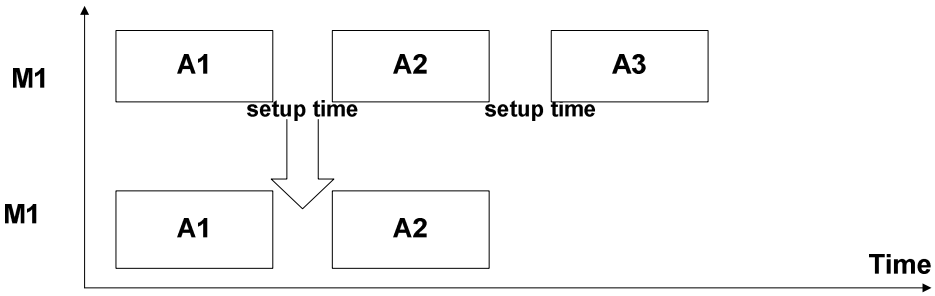

Figure 4.9 remove A3 from the end of machine M1

We need to consider propagation when we reflect a change. The propagation process is performed whenever an action is added or deleted. The key idea of propagation is that we need to adjust times for moving. Any adjustments to a container's start and end times may affect other container moves allocated to different machines. For example, when inserting A3 between A1 and A2, the starting and ending times of A2 (and anything after it) need to be recomputed. Because of constraints also need to propagate these delays to other machines. Note that the cost calculation is approximate since it does not do propagation.

4.2 Process

The solution process we propose has two phases: initial allocation and optimisation. In the initial allocation phase, containers are allocated to the cheapest bidding machine. The initial allocation is done according to the following process:

1) Consider all machines and check which machine is available to allocate unallocated moves. If a machine is available, then it bids.

2) The machine's bid is based on the incremental cost difference. When allocating the unallocated moves, we check the cost difference of each move. Containers are allocated to the cheapest bidding machine.

Constraints need to be considered during allocation process. We need to ensure that each container is unloaded from the ship before it is scheduled to be moved to the yard. We track the time at which a container reaches the apron (its "minTime"). The straddle carriers cannot start moving the container earlier than its minTime, this is ensured by calculating the start time to not just be the ending time of the previous action plus the setup time, but to also take

the minTime into account. Specifically, the start time of an action is the larger of its minTime and its computed start time. However, we do not need to consider the constraint that quay cranes'schedules may be affected by the straddle carriers' schedules because there are buffers, and in practice there are enough straddle carriers so that the buffer space is sufficient.

The optimisation phase aims to improve on the initial solution by swapping container moves between (and within) machines. The possible reallocations are evaluated based on an approximation of the cost difference. In order to obtain a new solution, the cheapest one is chosen and applied to the current solution. When swapping container moves, it considers all possible values for source machine, destination machine and positions within two machines. It then rules out moves where any of following hold:

1) The source and destination machine are the same, and source and destination positions are the same (i.e. a move with no effect).
2) The moves changes the "past" (discussed later in this chapter).
3) The move involves a quay crane (This is because we take the allocation of containers to quay crane as given – since a container is in a certain bay on the ship, and each bay is handled by a single quay crane, we cannot re-allocate a container to a different quay crane.)
4) The move would violate precedence constraints (described below).

During proposing and applying container move reallocations, we need to ensure that we do not violate the "unload before move" constraint. We also need to ensure that we do not propose to deal with containers in a non-viable sequence (e.g. if container A is on top of container B in the ship, then we must deal with container A first).

When reallocating container move to machines, the optimisation phase needs to consider a) moving it to a different position in its machine's schedule; or b) moving it to a different machine. Now we show two cases how the actions are constrained on a same machine or different machines.

Case1: actions which are constrained are on the same machine. Figure 4.10 shows two actions (A1 and A2) which are allocated to the same machine M1. Suppose that these two actions are constrained so that A1 must occur before A2. It is simple when we check the

constraint for this case. The constraint is true if A1 is before A2 on M1, otherwise the constraint is false.

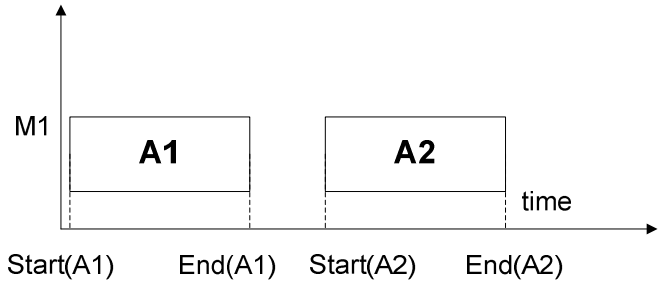

Figure 4.10 constrained actions are on the same machine

Case2: actions which are constrained are on different machines. Figure 4.11 shows two actions (A1 and A2) which are respectively allocated to machine M1 and M2, and which have the constraints that A1 must occur before A2. In this case we need check whether the constraint holds by looking at the times. The constraint that A1 must be before A2 is true if A1's end time on M1 is less than A2's start time on M2, otherwise it is false.

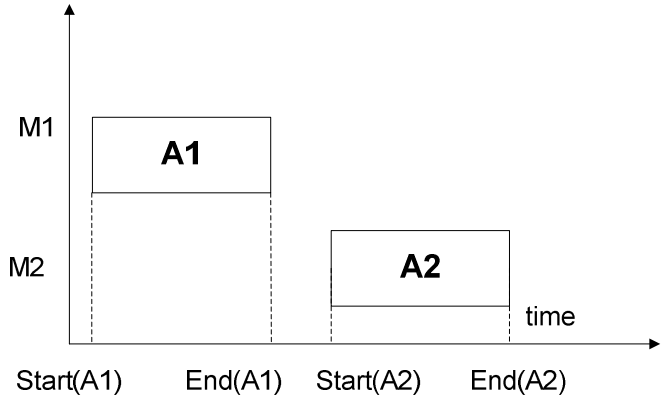

Figure 4.11 constrained actions are on the different machines

Figure 4.12 gives a method for checking whether any constraint would be violated by allocating a new action to a given machine. Note that when dealing with constraints, we ignore those that involve unplaced actions since they cannot be determined yet and will be checked when the relevant actions are allocated. Also note that in fact the algorithm in Figure 4.12 can stop if at any point both 'Before' and 'After' are empty.

Precedence Constraints Checking Process

precedenceConstraintsCheck(newAction NA, Machine M, Position P)

1. Initialize set of actions

2.Let 'Before' be the set of actions that are constrained to be before new action (NA)

3. Let 'After' be the set of actions that are constrained to be after new action (NA)

4. Compute the start time and end time of new action(NA) based on its position P on machine M

5. **For** all actions are before new action (NA) on machine M
   Remove the actions from 'Before'
   Similar, remove the actions after new action(NA) from 'After'

6. **For** each machine M'
   For each action A in machine M'
   If A's end time is equal or less than new action(NA)'s start time
   Then remove A from 'Before'
   If A's start time is equal or greater than new action(NA)'s end time
   Then remove A from 'After'

7. **If** 'Before' and 'After' are both empty
   Then all constraints are satisfied;
   Else there are constraints that are violated

Figure 4.12 Pseudo-Code for checking precedence constraints

Finally, the algorithm presented develops a schedule before execution. However, we need to take into account that the schedule will be executed over time in a dynamic and error prone environment. The schedule can be updated to reflect changes, and the allocation process re-run in order to deal with changes. We define a variable "currentTime" which represents the current time, i.e. all action moves with time less than the current time have already been done. Then during the process of container allocation and optimisation, we ensure that container moves which have already been done (i.e. in the past) do not get changed. For example (Figure 4.13), suppose there are three actions (A1, A2 and A3) which are allocated to machine M1, each action has start time and end time, and the currentTime is 18. We cannot add a new action between A1 and A2 since this would change the past. Similarly, A2 cannot be removed since it is something the machine is currently doing. A1 is also not able to be removed since it has already been done.

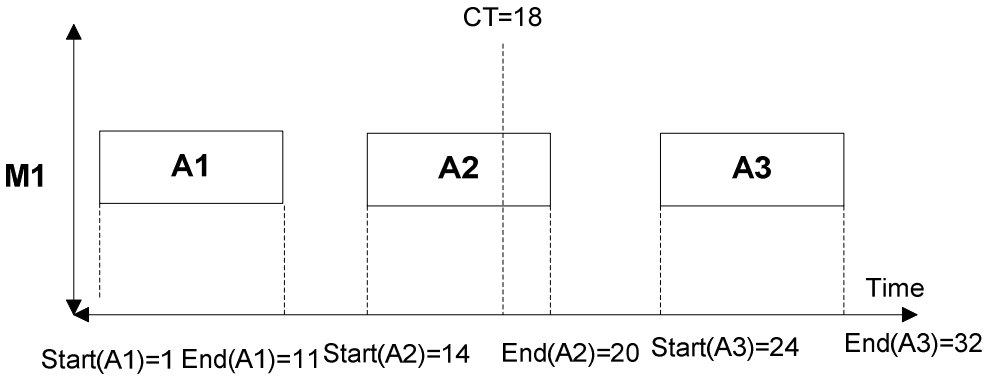

Figure 4.13 the currentTime of machine

4.3 Tabu Search implementation

Chapter 2 has described the algorithm of Tabu Search. This section describes how the negotiation process described in this chapter is implemented as a Tabu Search. Tabu Search searches through a solution space by making repeated changes to an initial solution while being guided by cost. The Tabu Search implementation that we use is provided by the OpenTS library. OpenTS is a Java Tabu Search framework that implements the Tabu Search meta-heuristics.

We map our negotiation process to the Tabu Search framework by defining two different types of Tabu Search moves: one for initial allocation, and one for optimisation.

4.3.1 Algorithm

As shown Figure 4.14, the Tabu Search implementation algorithm can be summarised as:

Step 1: Initialising containers & machines.
Step 2: Allocating container moves to machines (repeated for each action).
Step 3: To improve solution, swap moves (repeated).

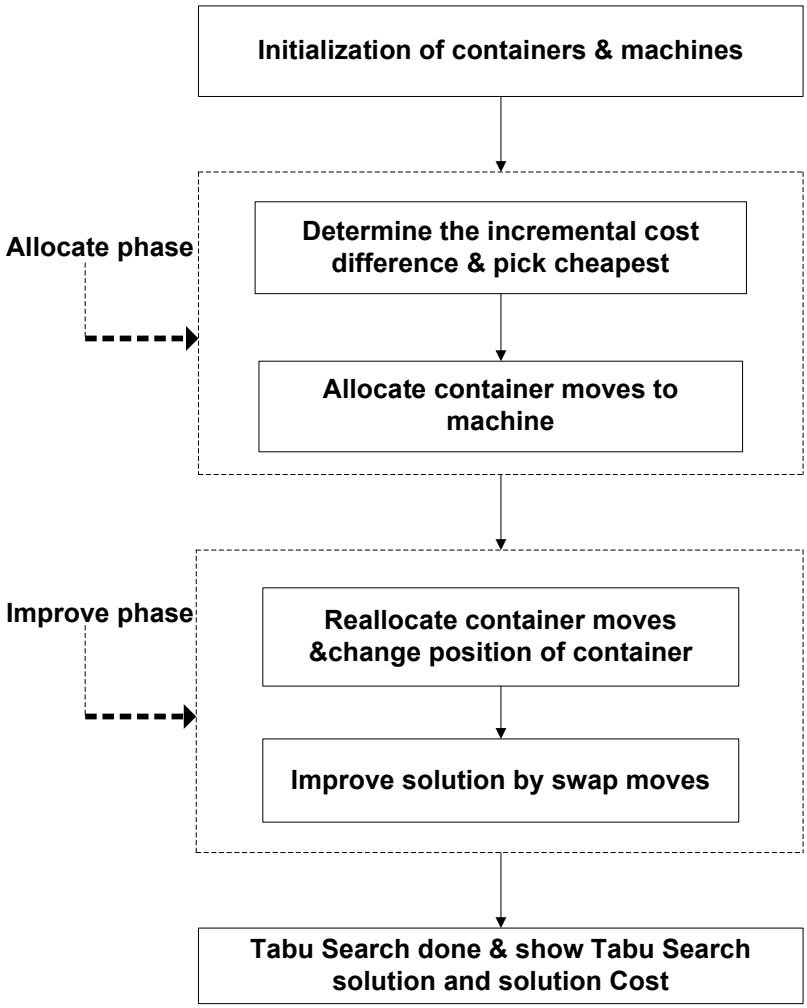

Figure 4.14 Tabu Search implementation algorithm

4.3.2 Initialisation

The input data from local port is written in Extensible Markup Language (XML)[3] format. In order to create empty solution, we use XStream[4] to read XML. In the initial step, we initialise different types of machines. Figure 4.15 shows that a quay crane and a straddle carrier are initialised. The initial solution contains machines with empty schedules and a queue of unassigned actions.

[3] http://en.wikipedia.org/wiki/XML

[4] http://xstream.codehaus.org/

```
Initialisation of Containers & Machines

CPSU1821385
... initialising new Quay Crane name=C4 with ID 0
... initialising new Straddle Carrier name=12 with ID 1
... initialising container CPSU1821385 =
   shipLocation[SHIP361240] -> apronLocation[C4TA0] ->
yardLocation[B19101]

Solution has 60 actions and 60 unplaced actions
```

Figure 4.15 Tabu Search output for initialisation

During the initialisation we also extract constraints from ship positions. For example, container A and container B in the ship positions respectively are (30, 8, 41) and (30, 8, 40). Firstly, we look at container's X and Y coordinates in the ship. For the same X and Y coordinates in the ship, the container with a higher Z coordinate needs to be unloaded first. For example (Figure 4.16), container A needs to be unloaded before container B since container A has a higher Z coordinate, and so we create a precedence constraints (i.e. A before B).

```
x is ReceiveShip, craneName is C3, itemId is Container A,
source(X,Y,Z)=(30,8,41)

Adding action(1, shipLocation[SHIP30,8,41] ->
shoreLocation[C3TA0]; Container A; craneName=C3) to x,y =
30,8

x is ReceiveShip, craneName is C3, itemId is Container B,
source(X,Y,Z)=(30,8,40)

Adding Action( 2, shipLocation[SHIP30,8,40] ->
shoreLocation[C3TA0]; Container B; craneName=C3) to x,y =
30,8
```

Figure 4.16 Tabu Search output for extracting constraints from ship positions

4.3.3 Initial Allocation Phase

In the Tabu Search, a move is defined from a solution to a neighbour solution. The two phases (allocate and improve) are modelled by two different types of moves: allocating container moves, and swaping moves.

Allocating container moves: this type of move is to allocate a container move ('Action'[5]) to a machine. A move is defined by a source machine[6] and position, and a destination machine and position. As shown Figure 4.17, Tabu Search shows that allocate container moves to machines from ship location to apron location.

```
Allocating Container Moves to Machines

>>> machine 0/C4 proposing move to position 0
>>> machine 2/C3 proposing move to position 0
>>> MoveManager returned 2 moves for Move( 0,
shipLocation[SHIP361240] -> apronLocation[C4TA0];
CPSU1821385)
Inserting move on Machine 0 at position 0
```

Figure 4.17 Tabu Search output for allocating container moves

4.3.4 Optimisation Phase

Optimisation phase is done by swap moves: container operation through this type of moves to improve solution. Move can reallocate moves from position of one machine to position of another machine or within a machine. The move manager is a controller which is able to generate possible moves. It also can determine which move is valid for each insertion move. The move manager deals with modifications to solutions ("moves"), whereas a quay crane deals with container moves ("actions"), As shown Figure 4.18, the solution is improved by swapping moves, and move manager generates 230 swap moves between machines.

[5] In order to avoid confusion between container moves and Tabu Search moves we have used the term 'Action' to denote a container move.
[6] src=-1 is used to denote a move from the queue of unallocated actions.

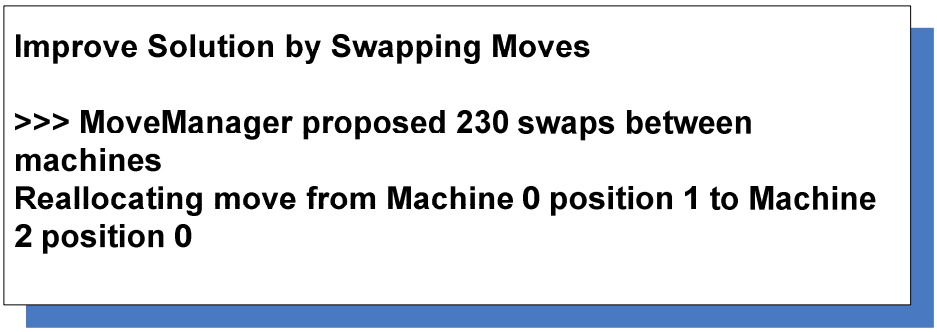

Figure 4.18 Tabu Search output for swap moves

It is worth emphasising that the process described in Section 4.2 is a distributed and parallel negotiation process. On the other hand, our implementation using Tabu Search is centralised: each container is considered in turn, and we wait for each machine to bid. This also assumes that each machine bids if appropriate.

Chapter 5: Evaluation

5.1 Aim of Evaluation

The previous chapter has proposed a solution for optimising allocation of moves to straddle carriers. In this chapter, we will evaluate our solution approach. The aim of evaluation is to: 1) assess the quality/effectiveness of our approach 2) understand the tradeoff between time and quality 3) assess to what extent our approach is robust (i.e. is able to deal with something going wrong).

5.2 Design of Evaluation

5.2.1 Input Data

The experiment input data is from a local port JMT (Jade Master Terminal) system which stores all the real-time information about container terminal activities. The data contains all recorded activities by machines which move containers around the local port terminal. The actions prefixed with "ReceiveShip", "SetDown", "PickUp" or "ReleaseShip" contain a summary of the journey of a container. The "ReceiveShip" and "ReleaseShip" activities include associated machines' types such as straddle carriers and quay cranes. The example of unloading a container from source location to destination location at the local port terminal is presented as follows. Note that the input information includes the actual straddle carriers that performed each container move. However, our evaluation ignored this information.

Container: FRLU8768454;
Date: 2009-01-24
Time: 14:30:24
Source Location (on a ship location): 543211;
Destination Location: B2131;
Machines: C2 which is a quay crane;
25 which is a straddle carrier;
Actions: 1) A move from location 543211 to location C3TA2 which is crane lane

2) A pick up by machine 25 at this crane lane

3) A set down at B2131

The input data was supplied as an XML file. In order to fit our simplified scenario (unloading operation only, using one area in the yard, and with no housekeeping moves considered), we select container moves which contains machine activity (e.g. “SetDown”, “pickup”) as well as a record of containers arriving and leaving port (e.g. “ReceiveShip”, “ReleaseShip”). We focus on ship to yard moves.

Our experiments used two data sets: a small data set and a larger data set. Both data sets involve a single ship. The smaller data set has 30 containers, 1 quay crane and 4 straddle carriers. The larger data set contains 82 containers, 3 quay cranes and 8 straddle carriers. An unloading container contains three transactions which describe information about this unloading operation.

5.2.2 Experimental Setup

There are no initial values input apart from the data. In order to understand the tradeoff between time and quality, we apply a number of optimisation steps. The number of iterations can be formulated as follows since initial allocation involves one move for each unassigned action.

Total Iterations = optimisation Level + number of unassigned actions

Quality and CPU time are the two variables that need to be measured. We measure the quality based on the cost of the best solution. We measure the CPU using user time plus system time. We use the Java 1.5 (Java.lang.management) method to report CPU time and user time per thread. Figure 5.1 shows how a Java method can report CPU time. It calls **ManagementFactory.getThreadMXBean ()** to describe current JVM threads. The **getCurrentThreadCpuTime ()** method returns the CPU time for the current thread.

```
Public static long getCpuTime(){
   ThreadMXBean bean = ManagementFactory.getThreadMXBean();
   Return bean. getCurrentThreadCpuTime();
}
```

Figure 5.1 Java method reports CPU time

Experiments were run on a personal computer with an Intel 2.16 GHz processor and 2046 MB RAM, running on Windows (XP) with JVM (JDK 1.5). For CPU time we take the average value of ten runs.

5.3 Results and Discussion

Figure 5.2 shows that for the small data set, the solution quality (i.e. its cost: smaller is better) is changed by the first few optimisation steps. The horizontal axis of figure 5.1 is the number of optimisation steps (0-50). The vertical axis of figure 5.1 is the solution cost. As can be seen, the initial solution resulting from the initial allocation phase is improved from optimisation one to optimisation five. But there is no further improvement after optimisation five.

Similarly, figure 5.3 shows that for the larger data set, the initial solution is also improved by the first few optimisation steps. As can be seen, there is also no further improvement after optimisation five.

As Figures 5.4 and 5.5 show, it can be observed that there is a linear increase between CPU time and iterations after number of optimisation steps. Note that we are measuring the time taken for a centralised implementation of what is really a distributed algorithm. A proper distributed implementation (see Chapter 6) would yield different CPU time measurements.

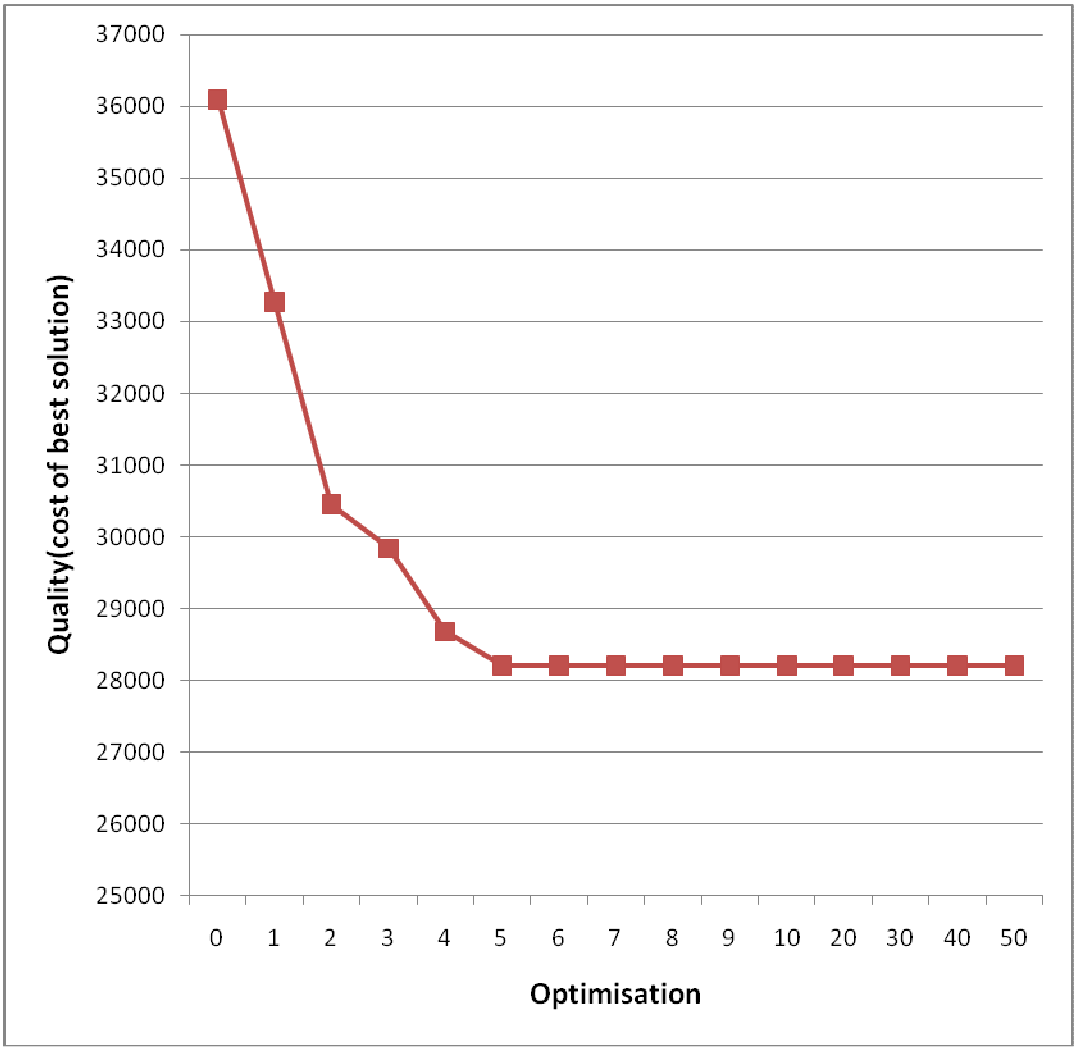

Figure 5.2 Experimental Results (Small data set)

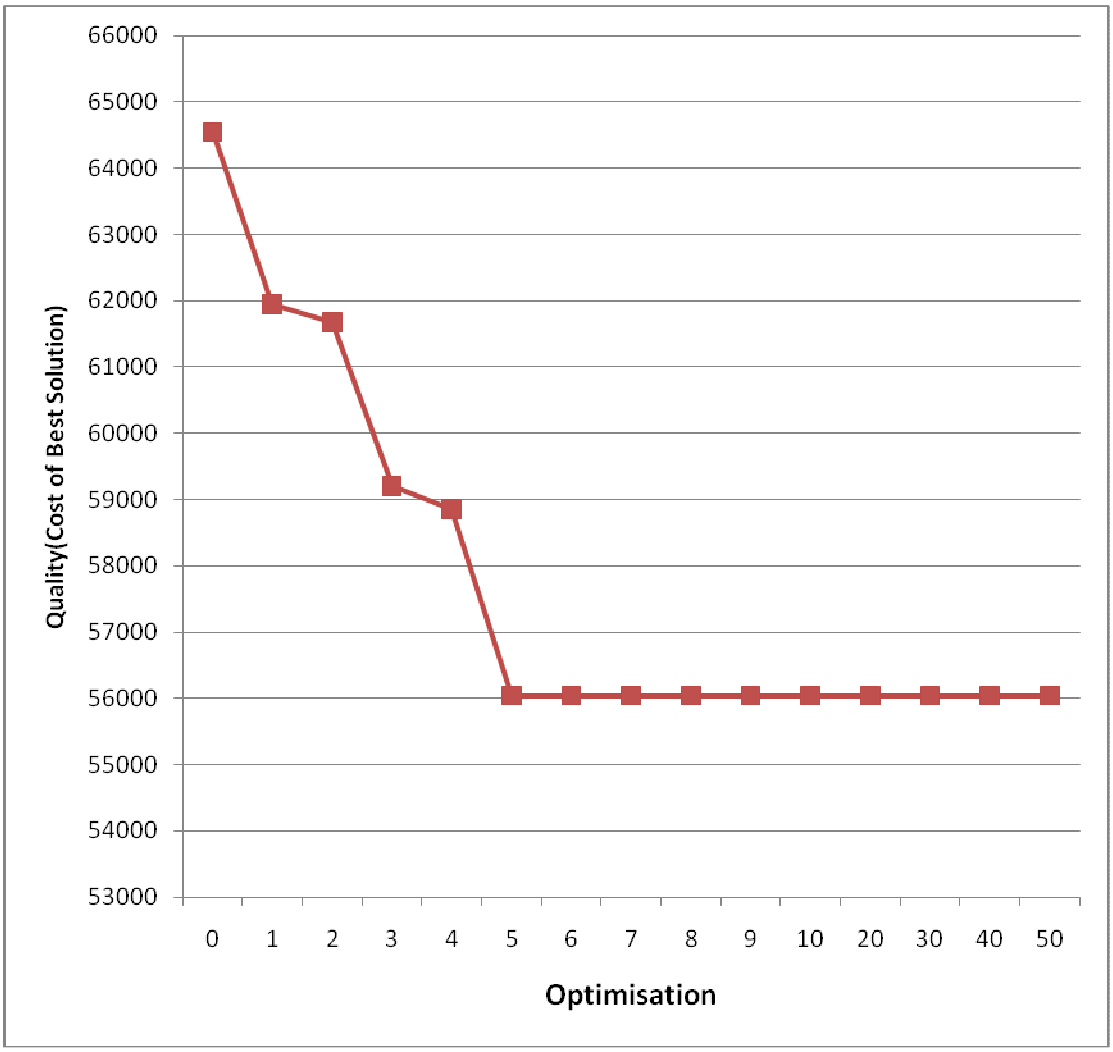

Figure 5.3 Experimental Results (Large data set)

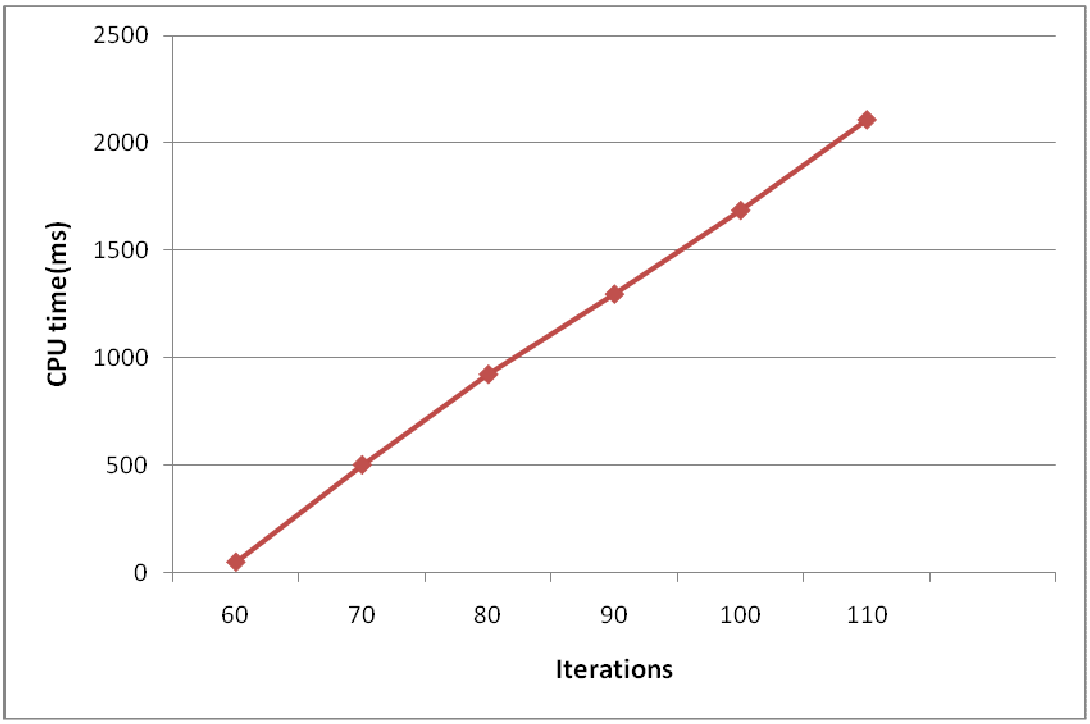

Figure 5.4 Iterations vs. CPU Time (Small data set)

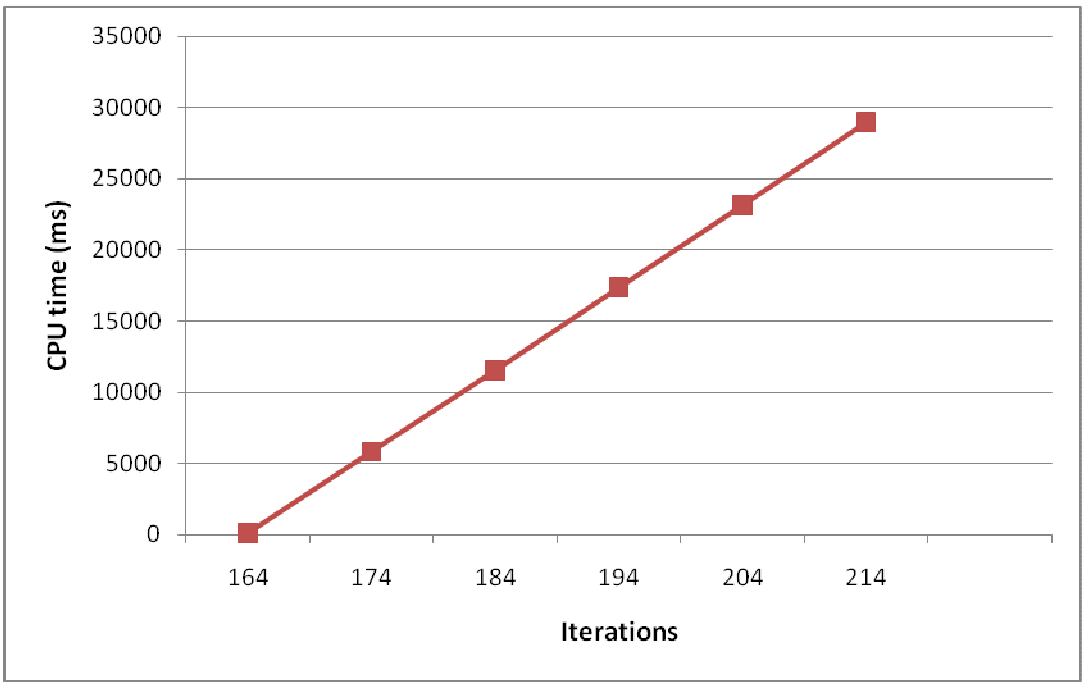

Figure 5.5 Iterations vs. CPU time (Large data set)

One question is whether the solution we have found is actually a good solution. Specifically, how close it is to the best possible solution? We did a preliminary assessment of this issue[7] using a small data set. We systematically generated all possible solutions, and assessed their quality. The small data set contains 10 containers, 7 straddle carriers and 2 quay cranes (C3 and C4).

We selected all containers which are allocated to C3 (which gave 4 containers and 3 straddle carriers) and all containers which are allocated with C4 (which gave 6 containers and 4 straddle carriers). We generated all possible assignments to straddle carriers, and then ran them through the system to check precedence constraints and calculate cost.

The experiment results show that there are 360 options for all possible allocations for the 4 containers unloaded by C3, and that the solution returned by Tabu Search after optimisation was indeed the best possible solution. However, we were not able to access the containers unloaded by C4, which had 60,480 possible options, due to Java limitations.

In summary, our results show that our solution approach is able to provide a quality solution for container terminal operation. The solution quality is improved after the optimisation step is repeatedly applied. This answers the research question by showing that an agent-based approach can be effectively used in a container terminal, specifically, to allocate container moves to straddle carriers. However, the local port only has 1-2 ships at once, and 3 quay cranes and a small number of straddle carriers. The scalability will be a topic for our further research. We now turn to the issue of how well our agent-based approach is able to deal with a dynamic environment.

5.4 Robustness

A challenge in ports is that the environment is dynamic. The strength of our approach is that it can deal with the unexpected. Our solution schedule is able to be updated to reflect changes and the allocation process re-run to deal with changes. In this section, we have selected a subset of the data from the local port, and then introduced errors in operation which are

[7] This assessment was done by the first supervisor

motivated by real errors from discussion with port terminal staff.. We also showed how we can use our solution to deal with these errors in operation.

Case 1: machine breaks down. For example, straddle carrier (e.g. 25) breaks down and cannot drop container, or cannot load containers.

Solution: Our solution can solve this case as follows: 1) the solution is updated by removing the straddle carrier (25) from the machine list 2) the allocated container moves are shifted into the list of moves to be allocated 3) the allocation process is re-run to allocate the broken straddle carrier's moves to other straddle carriers. Figure 5.5 shows how the allocation process allocates the broken straddle carrier's moves to other straddle carriers.

```
TABU Best Solution cost = 56040.0
METRIC, Diff in cpu time  = 2250

TABU searcher: Straddle Carrier(25) is broken,
removing Straddle Carrier(25)from machine list

>>> machine 0/23 proposing move to position 0
>>> machine 2/26 proposing move to position 0
>>> MoveManager returned 2 moves for Move( 0,
apronLocation[C4TA0]->yardLocation[B3451];
```

Figure 5.5 output of Tabu Search

Case 2: ships arrive out of order. For example, ship A is not coming, so ship B has to take some of its containers. These will be overstowed with ship B's other containers. So extra moves are needed within the yard area to access the containers, which adds to container operational cost.

Solution: Our solution can solve this case as follows: 1) given a yard model (we don't have yard model in our solution, but the JMT system does) we can derive the needed container moves 2) then we schedule these moves.

Case 3: cannot perform move (e.g. misaligned container or container not found).

Solution: Our solution can solve this case as follows: 1) proposed action has to be removed from action list which is done by humans 2) the machine's schedule needs to be adjusted.

In this chapter we have designed the experimental evaluation based on input data from the local port. The initial solution resulting from the initial allocation phase is improved by our optimisation process. We also evaluate how our solution can deal with something going wrong during container terminal operation based on a few cases (machine breaks down, container not found and ships arrive out of order). The results show that our solution is able to deal with these cases.

Chapter 6: Summary and directions for future research

In the last chapter of this thesis, we will summarize the findings of this research and recommend directions for future research.

Container terminals play a critical role in international shipping. The pressure is adding to container terminal to improve efficiency to cope with growing volume of containers. We have described the problem of container terminal operation in Chapter 2. One sub-problem is how we can manage straddle carriers well so that can decrease ship turnaround time. In this thesis, we have investigated how to optimise of allocation of container moves to straddle carriers.

In Chapter 4, we have developed an agent-based mechanism for allocating container moves to straddle carriers. We proposed a solution which is able to deal with container moves allocation problems. In order to derive this solution, we have designed a process based on two phases: initial allocation and optimisation. Initial allocation means that containers are assigned to the cheapest machine. The precedence constraints also will be considered. The optimization phase aims to improve terminal efficiency by swapping container moves between (and within) machines, while respecting the constraints. We implemented this negotiation-based approach for container management using a Tabu Search framework (OpenTS).

In Chapter 5, we have conducted evaluations using real data from a local port. We have done two experiments based on two data sets: a small data set and a larger data set. The experiment results have shown that our approach is able to provide a feasible optimisation solution to the complex problem of container terminal operation. The experiment results also have shown that our solution is able to deal with robustness (i.e. machine breaks down and misaligned containers). Our solution schedule can be updated to reflect changes and the allocation process re-run to deal with changes.

In summary, the strengths of this research are that we developed an agent-based mechanism, which effectively optimised the allocation of container moves to straddle carriers and was able to handle things going wrong. Our conclusion is that an agent-based approach is a good choice to deal with complex problems of container terminal operation.

However, there are some limitations of my work:

1) We focus on straddle carrier's allocation only. We have not done research on quay crane allocation. We also have not done research on other port machine types such as Automated Guided Vehicle (AGV) since the local port has a particular setup.

2) There is no yard model in our solution. We do not focus on dealing with yard allocation problems. We may have a yard model in our solution later since the JMT system does.

3) We only deal with unloading operations. The container terminal operation contains unloading and loading process. In order to easily design our solution approach, we only focus on unloading operation. We do not consider loading operation.

4) Somewhat limited evaluation. We only do two experimental evaluations based on input smaller and larger data. In order to get better result, we should do more experiments to evaluate our solution.

There are a range of directions for future work:

1) Developing a distributed version of the container optimisation algorithm, including heuristics to avoid looking at all possible swaps. The future research can investigate how to implement optimisation in terms of distributed negotiation. An issue that will be of great importance is to avoid too many communications between agents, which will result in extra cost and poor performance.

2) Exploring the use of the optimisation system as human decision support. Humans still play a critical role in container terminal operation. Operation of container terminal is

still controlled by the experience of human planners. Our optimisation system aims to support them to improve efficiency of container terminal operation rather than replacing them.

3) There can be an error in the predicted cost as our cost calculations do not propagate the impact of swapping moves. We will consider the accumulation of errors in our future research.

References

[1] http://en.wikipedia.org/wiki/Containerization

[2]IHS Global Insight's World Trade Service (WTS). http://www.ihsglobalinsight.com/ Accessed 2010-08-01

[3] http://www.unescap.org/ttdw/publications/tfs_pubs/pub_2398/pub_2398_ch3.pdf

[4] M. Wooldridge and N. R. Jennings. Intelligent agents: theory and practice. The Knowledge Engineering Review, 10(2), 115-152, 1995.

[5] P. McBurneyand M. Luck. The agents are all busy doing stuff! IEEE Intelligent Systems, 22(4):6-7, July/August 2007

[6] S. Munroe, T. Miller, R. A. Belecheanu, M. Pechoucek, P. McBurney, and M. Luck.Crossing the agent technology chasm: Experiences and challenges in commercial applications of agents. Knowledge Engineering Review, 21(4):345-392, 2006

[7] D. Steenken, S. VoB, R. Stahlbock, Container terminal operation and operations research-a classification and literature review. OR Spectrum 26:3-49,2004

[8] F. Persyn, Exploring Port Productivity Measurement, in Institute of Transportation and Maritime Management. University of Antwerp: Antwerp, Belgium,1998.

[9] L. Moccia, J-F. Cordeau, M. Gaudioso, and G. Laporte. A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. Naval ResearchLogistics 53 (1), pp. 45-59, 2006.

[10] F. Glover, "A User's Guide to Tabu Search", Annals of Oper. Res., VOL.41,PP.3-28, 1993

[11] T. Thurston, H. Hu , Distributed agent architecture for port automation. In: Proceedings of the 26th annual international computer software and applications conference (COMPSAC'02), Oxford, August 26-29. IEEE Computer Society, Los Alamitos, pp81-87, 2002

[12] C. Degano and A. Pellegrino. Multi-agent coordination and collaboration for control and optimization strategies in an intermodal container terminal. Engineering Management Conference, pages 590-595, 2002 IEEE.

[13] Y. Peng, J. Sun, "Agent Based Container Terminal Optimization," case, pp.607-609, 2009 IITA International Conference on Control, Automation and Systems Engineering , 2009

[14] N. Yan, G. zhong and Z. Xi "A Multi-Agent-Based Production Dispatching Information Integration System for an Automated Container Terminal" Proceedings of the 2008 IEEE, International Conference on Information and Automation, June20-23, Zhangjiajie, China, 2008

[15] C. Carrascosa, M. Rebollo, V. Julian, V. Botti. A MAS approach for port container terminal management: the transtainer agent. In: Actas de SCI'01, pp 1–5. International Institute of Informatics and Systemics, Orlando, FL, 2001

[16] FIPA Specification Repository web page. http://www.fipa.org/repository/

[17] M. Kefi, O. Korbaa, K. Ghedira, and P. Yim. Container handling using multi-agent architecture. In N.T. Nguyen et al., editor, KES-AMSTA, volume LNAI4496, pages 685-693, Springer, 2007.

[18] JADE Development Environment User's Guide.
http://www.jadeworld.com/downloads/jade/manuals/UserGuide.pdf

[19] P. Lokuge, D. Alahakoon, "Hybrid BDI Agents with ANFIS in Identifying Goal Success Factor in a Container Terminal Application". International Conference on Autonomous Agents and Multiagent Systems (AAMAS): 1222-1223, 2004

[20] P. Lokuge, D. Alahakoon, " Improving the adaptability in automated vessel scheduling in container ports using intelligent software agents". European Journal of Operational Research 177, 1985-2015,2005

[21] B. Sun, J. Sun and P. Yang, “The design and implementation of berth allocation management system based on MAS,” The Fifth International Conference on Natural Computation (ICNC), vol.5, PP.593-597, 2009

[22] L. Henesey, P. Davidsson and J. A. Person, “Agent Based Simulation Architecture for Evaluating Operational Policies in Transhipping Containers” published in the Fourth German Conference on Multiagent System Technologies, Berlin and Heidelberg, Germany, PP 73-85, 2006

[23] J. Dai, W. Lin, R. Moorthy, C-P. Teo, Berth allocation planning optimisation in container terminal. Working paper, Georgia Institute of Technology, Atlanta; National University of Singapore, 2004

[24] K. H. Kim and Y-M. Park. A crane scheduling method for port container terminals. European Journal of Operational Rsearch, 156: 752-768,2004.

[25] P. Canonaco, P. Legato, R. M. Mazza, R. Musmanno. A queuing network model for the management of berth crane operations. Computer & Operations Research 35,2432-2446, 2008.

[26] L. H. Lee, E. P. Chew, K. C. Tan and Y. Han. An optimisation model for storage yard management in transhipment hubs. OR Spectrum 28:539-561, 2006

[27] S. Balev, F. Guinand, G. Lesauvage and D. Olivier. Dynamic handling of straddle carriers activities on a container terminal in uncertain environment-a swarm intelligence approach-, The Third International Conference on Complex Systems and Applications, Le Havre : France, 2009

[28] I. Davidson. R. Kowalczyk, Towards Better Approaches to Decision Support in Logistics Problems, Industrial Logistics, Feb 1997.

[29] A. Pokahr, L. Braubach, J. Sudeikat, W. Renz, and W. Lamersdorf. Simulation and implementation of logistics systems based on agent technology. In Hamburg International Conference on Logistics: Logistics Networks and Nodes, 2008.

[30] K. Dorer and M. Calisti, An adaptive solution to dynamic transport optimisation. In Autonomous Agents and Multiagent Systems (AAMAS), pages 45-51. ACM Press, July 2005.

[31] B. Zeddini, M. Temani, A. Yassine, K. Ghedira, "An agent-oriented approach for the Dynamic Vehicle Routing Problem", 2008 International Workshop on Advanced Information System for Enterprises, 0:70-76, 2008.

[32] L. Chen, N. Bostel, P. Dejax, J. Cai, L. Xi, " A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal", European Journal of Operational Research 181, 40-58,2007