2008-03

Real-time dispatching of rubber tired gantry cranes in container terminals

McNary, Bradley S.

Monterey California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

REAL-TIME DISPATCHING OF RUBBER TIRED GANTRY CRANES IN CONTAINER TERMINALS

by

Bradley S. McNary

March 2008

Thesis Advisor:	Johannes O. Royset
Second Reader:	Robert F. Dell

Approved for public release; distribution unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2008	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Real-Time Dispatching of Rubber Tired Gantry Cranes in Container Terminals			**5. FUNDING NUMBERS**
6. AUTHOR(S) Bradley S. McNary			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Within the past 50 years, containerization and globalization have driven a change from small container terminals to large container terminals that need efficient logistic models to keep up with the significant growth in container traffic. Efficiently managing rubber-tired gantry cranes and planning container placement within the terminal are two ways to increase the overall efficiency of a terminal. In this thesis, we combine these strategies in a real-time dispatching tool using an approximate dynamic programming heuristic. The heuristic re-optimizes at the rate the quay crane handles containers, incorporating endogenous and exogenous information in each solution. We formulated and solved an Integer Linear Program (ILP) to estimate the heuristic's solution quality. The heuristic finds solutions within seconds and the absolute gap between the heuristic solution and the ILP solutions remained essentially constant as the size of the problem increased.

14. SUBJECT TERMS Approximate Dynamic Programming, Optimization, Container Terminal, Yard Management Strategies, Crane Deployment, Rubber Tired Gantry Cranes, Container Grounding Strategies, Crane Dispatching			**15. NUMBER OF PAGES** 57
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

REAL-TIME DISPATCHING OF RUBBER TIRED GANTRY CRANES IN CONTAINER TERMINALS

Bradley S. McNary
Lieutenant, United States Navy
B.S., United States Naval Academy, 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 2008

Author: Bradley S. McNary

Approved by: Johannes O. Royset
Thesis Advisor

Robert F. Dell
Second Reader

James N. Eagle
Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Within the past 50 years, containerization and globalization have driven a change from small container terminals to large container terminals that need efficient logistic models to keep up with the significant growth in container traffic. Efficiently managing rubber-tired gantry cranes and planning container placement within the terminal are two ways to increase the overall efficiency of a terminal. In this thesis, we combine these strategies in a real-time dispatching tool using an approximate dynamic programming heuristic. The heuristic re-optimizes at the rate the quay crane handles containers, incorporating endogenous and exogenous information in each solution. We formulated and solved an Integer Linear Program (ILP) to estimate the heuristic's solution quality. The heuristic finds solutions within seconds and the absolute gap between the heuristic solution and the ILP solutions remained essentially constant as the size of the problem increased.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Due to containerization and globalization, the shipping industry has seen significant growth. The growth that the shipping industry is experiencing can also be seen in container terminals through an increase in the volume of containers that have to be handled and processed. With record growth and limited space, container terminals need efficient models of their operations.

The commonly accepted performance measure of a container terminal is the time a container ship is berthed. The operation within the terminal that can most impede a timely-turn around of a container ship is the movement of the rubber tired gantry cranes (RTG). Due to this fact, it is critical to plan the efficient movement of RTGs within a container terminal. Also, when an import container is removed from a container ship there are a multitude of places within the terminal that a container can be temporarily stored. Terminals have internal policies called grounding policies that determine where to place a container. Like planning the efficient movement of RTGs, container terminal wants to establish a grounding policy that will aid in reducing a containers ship's berth time. Traditionally, the grounding policy is found independently of the efficient movement of RTGs.

In this thesis we develop a model to find a grounding policy in conjunction with the efficient movement of RTGs to increase the throughput of import and export containers and to reduce a container ship's berthing time. The objective is achieved through a deterministic real-time dispatching model solved using an approximate dynamic programming (ADP) heuristic. Both the grounding policy and the movement of RTGs are optimized simultaneously in order to find an approximately optimal solution. The output of the model is a grounding policy and recommended RTG movement for a given planning horizon. The model is reoptimized in real time using a rolling planning horizon and updated forecasts of required container movement and RTG availability.

We formulate an integer linear program (ILP) to bound the optimal value of the RTG movement and ground policy optimization problem. The ILP is a relaxation of our

problem and provides a lower bound. Additionally, there is a prior formulation that models a restriction of the same problem and we use it to establish upper bounds to our problem.

The ADP based heuristic solves problem instances with a planning horizon of up to 100 containers in less than 30 seconds for six RTGs covering 36 blocks. Based on solve times, our ADP heuristic is suitable for real-time dispatching of container terminal assets. The objective function values obtained using the heuristic are well below the upper bound from the prior, restrictive model. Additionally, the objective values obtained from the heuristic follow the same trends as the lower bounds from the baseline model as the planning horizon increases.

ACKNOWLEDGMENTS

I thank Prof. Royset, Prof. Carlyle and Prof. Dell for their help and guidance in the writing of this thesis. The process and journey of completing the thesis has been a tremendous learning process. The task could not have been completed without their expert guidance and mentoring. I additionally thank Prof. Alderson for his help and expertise. I am also greatly appreciative of the foundation knowledge that the faculty has laid. Additionally, I thank the staff that has helped me with the successful completion of the degree and thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. CONTAINERIZATION

Containerization, the practice of loading goods into standardized sealable containers for overseas shipment on large container ships, was developed in the 1960s and makes it much easier to move bulk goods from port to port by reducing the time and labor required to load and offload container ships. Containerization has made the shipping industry more profitable and competitive (Levinson, 2006), and has enabled the globalization of many other industries. Due to these two factors, the shipping industry has changed significantly within the past 50 years and is still experiencing significant growth. In the United States alone, container traffic has tripled in volume from 1995 to 2006 and has grown 10% alone from 2005 to 2006. It is also estimated that 90% of the volume of world trade is handled at one point by the shipping industry (U. S. Department of Transportation, 2007).

A *container* is a large metal box used to store goods for shipment overseas. Containers come in a small number of standard sizes to make their handling easier. A *container ship* is a merchant vessel that is engineered to optimally transship containers. A *container terminal* (or simply *terminal*) is a facility in a port that manages the loading, unloading and temporary storage of shipping containers. There are hundreds of container terminals located at large ports worldwide. Shipping containers arrive at and depart from a container terminal via container ships, trucks and trains. For a container terminal to be competitive, it must keep shipping costs down by performing its operations efficiently.

In this thesis, we focus on the movement of containers to and from container ships. When an *import* container arrives on a container ship, the container will be temporarily stored in the *yard* of a container terminal. After the container is stored, it is exported via one of the above modes of transportation. All containers stored within the yard for future loading onto other container ships are *export* containers. The yard of a container terminal is made up of series of container *blocks*, each block is an area where many containers are stacked and stored together. Each block has a *buffer area* which is a

queue for containers waiting to be placed within the block. A series of adjacent blocks makes up a *lane.* The *quayside* or *berth* is the area of the terminal that a container ship is berthed and quay cranes are positioned to load and discharge containers from a container ship (Figure 1).

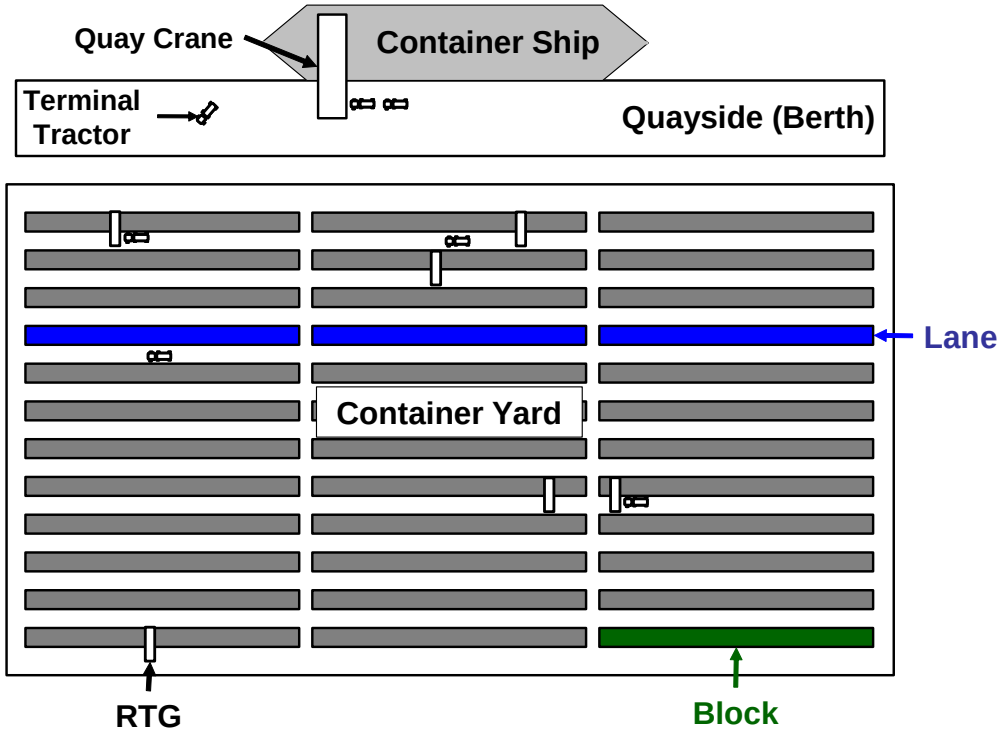

Figure 1. Container Terminal Layout: A schematic overhead view of a container terminal that has 12 lanes and 36 blocks. The berth has a container ship that has a quay crane to handle the import and export container sequence. The containers are moved between the berth and the yard on the terminal tractors. Within the yard RTGs are moving export containers from blocks to terminal tractors, and vice-versa for import containers.

1. Terminal Equipment

The equipment within a container terminal varies from facility to facility. In order to improve the efficiency of container facilities, we investigate the operations of quay cranes, rubber tired gantry cranes and terminal tractors.

The *quay crane* (Figure 2) is an essential part of operations at a terminal. The quay crane is located in the berth of a terminal and is used to load and discharge

containers from a container ship. In the *load* operations, export containers are moved from the yard to the quay crane via a terminal tractor. The quay then removes the export container from the terminal tractor to place it onboard a container ship. For the *discharge* operation, an import container is removed from a container ship by a quay crane and placed on a terminal tractor for transit. The terminal tractor moves the container to an appropriate location within the yard for temporary storage. The quay crane can vary the rate at which it loads and discharges containers. This variable rate, called the *push rate*, can be slowed to help prevent or alleviate congestion within the container terminal.

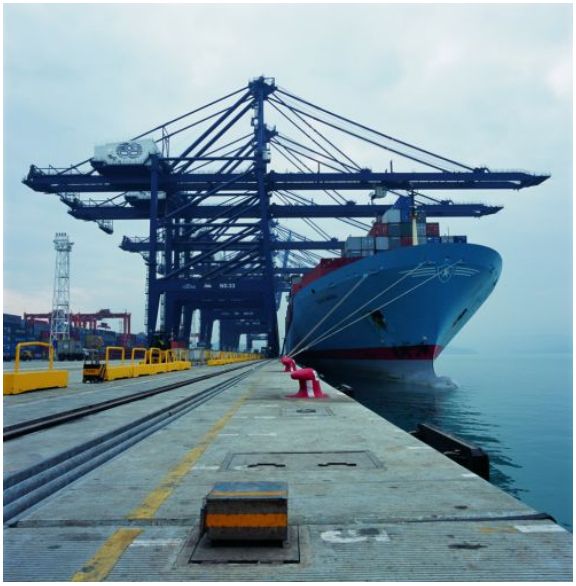

Figure 2. Quay crane: Quay cranes loading and discharging containers from a container ship (From Container Shipping Information Service, 2008).

Within the yard of the terminal the assets performing the majority of the work are the r*ubber tired gantry cranes* (RTG) and the *terminal tractors* (Figure 3). A terminal tractor moves a single container between a quay crane and a RTG. A RTG is a crane that moves from block to block working on containers. The work of a RTG is defined as moving import containers off terminal tractors to place them within a block and vice versa for export containers.

Figure 3. Rubber Tired Gantry Crane and Terminal Tractor: A terminal tractor in the buffer area waiting on a RTG. The RTG will move the containers from the terminal tractor to place the container in the block (From Container Shipping Information Service, 2008).

2. Operations

Within the terminal, minimizing the berth time of a container ship is not as simple as buying more cranes. Buying more quay cranes for a terminal so that more work can be accomplished is cost prohibitive, and placing more RTG cranes in the yard could potentially slow a terminal's operations due to traffic congestion. In the daily operations of a container terminal, RTGs are frequently the cause of delays. The efficient operation of each RTG is the key to expediting the movement of import and export containers and reducing the berthing time of a container ship.

The focus in this thesis is placed on the RTG operation because of the potential it has to slow terminal operations if their operation is not planned appropriately. A RTG is not constrained to one work block, but is free to move to other work blocks as necessary. The movement of RTGs is essential in the task of completing the work on the import and export containers, but it comes at a price. The movement is a timely operation that

obligates additional personnel, causes congestion within the terminal, and reduces the effectiveness of a RTG because it is not handling containers during the move. Figures 4 and 5 are two examples of RTG transitions. The more complex the move that a RTG makes, the greater the potential for a reduction in terminal efficiency.

Container terminals have different policies and procedures for determining where an import container is temporarily stored in the yard. The rule for determining this container placement is referred to as the *grounding policy*. The grounding policy typically groups containers with similar characteristics such as content, origin, and destination. A grounding policy that is enacted in conjunction with the dispatching or movement of the RTGs will result in a more efficient planning process.

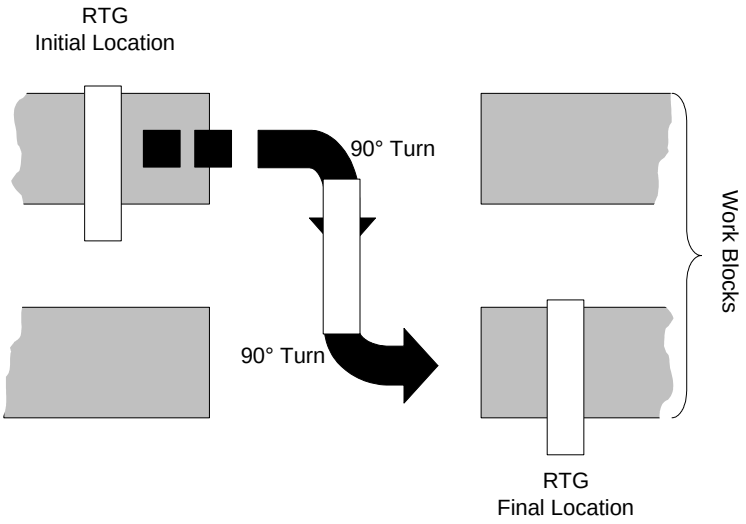

Figure 4. Complex RTG Move: A RTG transitioning to a block in an adjacent lane requires a RTG to turn 90º on its wheels twice.

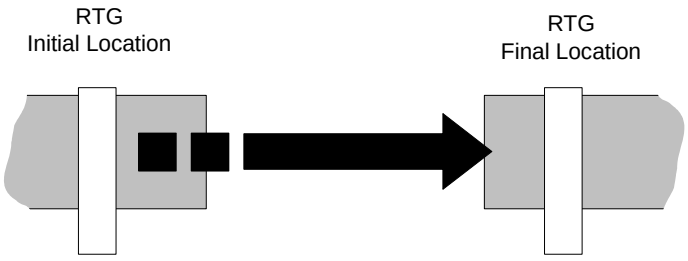

Figure 5. Simple RTG Move: A RTG transitions to an adjacent block within its lane. The move does not require a RTG to turn 90º on its wheels and is the most efficient move between blocks.

B. RESEARCH GOAL AND APPROACH

In order to improve the efficiency of RTGs in a terminal, we have developed a mathematical model of the terminal operations that directly involve container ship cargo. We present this model and a real-time dispatching tool that uses *approximate dynamic programming* (ADP) to find efficient solutions. Our model helps improve the operations of the RTGs in a terminal, thereby increasing the throughput of import and export containers and reducing each container ship's berthing time. The real-time dispatching tool will take as input a *container sequence* (i.e., a known sequence of "jobs" for the quay crane) and will provide as output a grounding policy for all import containers and recommended RTG moves. The tool will ensure that all export containers are moved out of the blocks to terminal tractors that will take them to the quay crane for loading onboard a container ship. The tool will also ensure that all import containers are placed into the blocks by a RTG. The objective of the real-time dispatching tool is to complete the work associated with the given container sequence in minimum time.

The decision points in the dynamic program (DP) are regularly spaced in time, with the spacing determined by the quay crane push rate, which is fixed at 60 seconds in our examples. At each decision point, the tool will determine a grounding policy (destination of containers on terminal tractors) for the next container that is being discharged from a container ship. In addition to the grounding policy, the tool will also determine the movement and handling sequence of the RTGs for import and export containers within a container terminal. The DP is deterministic; all input data is assumed to be known. However, to handle changes that occur in real-time, the tool will be used with re-optimization at each decision point, incorporating any new data at that time. Some examples of the exogenous processes that could have an impact on operations are changes to a container sequence, delays in the transit of RTGs or terminal tractors due to congestion, or mechanical failures that remove assets from operation.

We solve the DP using a heuristic algorithm based on *approximate dynamic programming* (ADP). At every decision point, the DP will provide solutions for a given *planning horizon,* which is the number of containers in the container sequence the DP

will solve for. The resulting solution provides a rule, or *policy*, for efficient movement of containers via terminal tractors (a grounding policy) in conjunction with the efficient movement of RTGs. Overall the tool will result in reduced berthing time for container ship, reduced traffic congestion in a container terminal and reduced work preformed within a container terminal.

Since the DP is solved using a heuristic, a comparison is needed to verify solution quality. We formulate an integer linear program (ILP) for baseline comparison. Additionally, the optimal values and computational times for ADP are compared with results from a previous ILP developed by Akel (2007). The two integer linear programs will give a measurement of performance for the ADP algorithm.

C. STRUCTURE OF THESIS AND CHAPTER OUTLINE

This thesis is divided into five chapters, including the Introduction. Chapter II gives information on developing events and studies in the area of container terminal optimization. The under lying decision problem is discussed and developed in Chapter III and then the DP and ILP are presented. In Chapter IV, we present a scenario with a discussion of the data set which was provided for the use in numerical tests. Following the scenario development is a comparative analysis of the numerical results. Chapter V gives conclusions based on the findings presented in this thesis in and recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. DEVELOPMENTS IN CONTAINER TERMINAL OPTIMIZATION

A. INTRODUCTION

A large degree of coordination and planning is needed for operations to run efficiently at container terminals. Optimizing operations within the container terminal yard is considered one of the most complex problems in the industry (Vis et al., 2003). In this chapter, we review some analytic approaches to container terminal optimization.

B. CURRENT DEVELOPMENTS

Kim et al. (2003) examines the problem of scheduling quay cranes within a container terminal. They present a heuristic to identify a load and discharge sequence for a quay crane. When identifying the sequence of containers to load and discharge, the objective is to minimize the container ship berth time. The optimal solution of the load and discharge sequence is found in isolation of other events within a container terminal.

In the area of yard resource planning, there are two topics of interest: (i) identifying the destination locations of the import containers or the grounding policy and (ii) the scheduling of the yard assets such as RTGs (Taleb-Ibrahimi et al., 1993). Taleb-Ibrahimi et al. (1993) discusses different container storage strategies and the handling requirement that is coupled with the strategies.

Zhang et al. (2002) explores RTG deployment based on workload requirements. Their approach formulates the problem as a mixed integer program; however, the problem is solved heuristically by Lagrangean relaxation in order to improve solution times. The study was limited to 20 blocks, and solutions were found for four-hour planning horizons, with the remaining unhandled work to the next time window. Linn and Zang (2003) developed a similar RTG deployment heuristic based on workload forecast. The scope of the problem was limited to 10 blocks and 10 RTGs to keep the

solve time of the mixed integer program reasonable. Their heuristic produces near-optimal solutions in seconds. In both of the above studies, the number of RTGs and blocks are kept equal.

Murty et al. (2005) discusses different decision support tools to be used in all aspects of container terminal operations. One performance measure that is of particular interest is a qualitative measure of congestion. The strategy is to monitor the flow of the traffic on the routes traveled. When the flow reaches an upper boundary, this indicates an increase in potential traffic congestion.

Ng (2005) developed a dynamic programming heuristic to schedule multiple yard cranes within a single zone. A zone is defined as the set of blocks that a RTG is constrained to. The focus was to find the optimal movement of yard cranes while avoiding the interference of other cranes to reduce terminal tractor waiting time. The scope of the study was limited to two yard cranes and two blocks.

Linn et al. (2003) developed a RTG deployment algorithm for an integer linear program to find the optimal deployment of RTG. A unique feature in the program allowed RTGs to temporarily store a container and then return to the container for optimal placement within the yard. The scenarios implemented had at least 68 blocks and 68 RTGs. The algorithm found solutions for four-hour windows cascading the remainder work to the re-optimization for the next four-hour time window.

C. FUTURE OF CONTAINER TERMINAL OPTIMIZATION

There have been significant developments in optimization of container logistics. Some of the developments range from optimizing the quay crane, managing the stowage plan for a container ship and optimizing the movement of RTGs in a terminal. There has been no specific focus on interrelating the different yard operations and optimizing multiple yard operations simultaneously. For example, it may be optimal for a quay crane to discharge a set of containers before handling the subsequent set. However, if the destination of all containers in the first set is the same, then terminal congestion may be unavoidable. Integrated operations need to be considered in the high performance environment of container terminals (Steenken et al., 2004). This thesis is a first attempt to optimize simultaneously both the grounding policy and the RTG movement policy.

III. MODEL DEVELOPMENT

A. MODEL DEVELOPMENT

There are many interrelated aspects of a container terminal that can have an impact on performance (*i.e.,* terminal layout, types of equipment used, external truck operations, and terminal tractor operations). All operations within a terminal are important and interrelated, but the computational complexity of a model that considers the multiple objectives of the different operations prevents us from proceeding in this direction. Since RTGs are considered the bottleneck of container terminal operations, we choose to limit the scope of the problem by looking specifically at a grounding policy in conjunction with the movement of RTGs. The model will increase the throughput of import and export containers and reduce a container ship's berthing time.

In order to sequence the movement of the RTGs with the model, a container sequence is put together by the container terminal personnel. Before a container ship pulls into a port, terminal personnel know the contents, destination and position in the discharge sequence of each import container. Based on the terminals internal rules on grounding a container within a terminal, personnel build a list of different areas where a container can be stored in the terminal. Additionally, the personnel know the location of the containers that are to be exported onto an incoming container ship. With the above information they build a sequenced list of containers to be discharged and loaded onto this ship by the quay crane

The data for the model is put together before a container ship arrives. With the sequenced list of containers and list of potential grounding areas for each import container, our objective is to find the most efficient sequencing of RTG work and grounding policy to satisfy the constraint of the container sequence. However, in the development of a model to meet this objective, we made some assumptions in order to simplify the problem.

For the terminal tractors, we specifically choose not to model their movement but instead assume that they are a freely available resource. With this choice, we do not have to consider the scheduling of terminal tractors and we can assume that the time that it takes a terminal tractor to traverse a route is deterministic, and based on the distance traveled. We acknowledge that a terminal tractor can be delayed due to congestion within the yard or due to exogenous processes, however, the time of delay is negligible compared to the time it takes to position a RTG within the yard to handle a container on a terminal tractor.

Within the model, RTGs are allowed to transit to any block within the yard. The RTGs are not limited to move to specific set of blocks. However, RTGs are not allowed to occupy the same block at the same time.

Each block in a container terminal has an associated *buffer*, a first-in-first-out queue that contains import containers waiting for a RTG to place them in the block. The buffer also contains empty terminal tractors that are waiting for an export container to transit to the quay cane.

Our final major assumption is made on the quay crane push rate. The push rate is the rate that a quay crane is moving containers to and from a container ship. The quay crane push rate can be varied in order to alleviate congestion within the yard, however, we assume the rate is constant at one unit of time (one minute).

B. DYNAMIC PROGRAM FORMULATION

We first formulate the RTG Operations Problem (RTGOP) as a dynamic program (DP). The DP seeks to minimize the total amount of time until a given sequence of containers have been moved to their respective destinations within the terminal or onboard a ship. The work is composed of import and export containers. Work on an import container consists of moving a container from a container ship to a block and work on an export container consists of moving a container from a block to a container ship.

The formulation of the DP is as follows:

Indices/Sets

$t \in T = \{1, 2, ..., \infty\}$	Discrete time step which represents the decision points in a continuous time process.
s	Container location representing onboard a container ship.
p	Container location representing at a quay crane.
$w, w' \in W = \{1, 2, ..., \lvert W \rvert\}$	Blocks that RTGs can occupy.
$b, b' \in B = \{1, 2, ..., \lvert B \rvert\}$	Buffer areas for containers.
$l, l' \in L = W \cup B \cup \{p\} \cup \{s\}$	Locations for containers.
$g, g' \in G = \{1, 2, ..., \lvert G \rvert\}$	RTGs.
$a \in A = \{0,1\}$	Type of shipping container (0 = import and 1 = export).
$r \in R = W \times [0, \infty)$	RTG attribute vectors.
$q \in Q = L \times [0, \infty) \times A$	Container attribute vectors.
$W(b) \subset W$	Block *w* that corresponds with buffer *b*.
$B(w) \subset B$	Buffer *b* that corresponds with block *w*.
$c_I, c_I' \in C_I = \{1, 2, ..., \lvert C_I \rvert\}$	Import containers.
$c_E, c_E' \in C_E = \{1, 2, ... \lvert C_E \rvert\}$	Export containers.
$c, c' \in C = C_I \cup C_E$	Container sequence to be imported or exported.

Data

t^{HAND}	Time for a RTG to move a container from a terminal tractor to a block of containers or vice-versa.
$t^{RTG}_{w,w'}$	Time to move a RTG from block *w* to block *w'*.
$t^{CON}_{l,l'}$	Time to move a container between locations *l* and *l'*.
$t^{AVAIL}_{c_I}$	Time of the day when import container c_I is scheduled to be available at the quay crane for transport to the yard.

$t_{c_E}^{AVAIL}$	Time of the day when export container c_E is scheduled to be loaded onto the container ship by the quay crane.
$\Delta(t)$	Length of time interval t (in our example $\Delta(t)$ =1 minute for all t)

The following variables, states, and functions are all defined at any time t in the planning horizon.

Decision Variables

$X_{t,g} \in W$	Selected block for RTG g.
$H_{t,g,c} \in \{0,1\}$	1 if RTG g assigned to start work on container c.
$Y_{t,c} \in L$	Selected location for container c.
$Z_t = \begin{pmatrix} \left(X_{t,g}\right)_{g \in G} \\ \left(H_{t,g,c}\right)_{g \in G, c \in C} \\ \left(Y_{t,c}\right)_{c \in C} \end{pmatrix}$	Vector of decisions.

States

$r_{t,g} = \begin{pmatrix} w_{t,g} \\ \tau_{t,g} \end{pmatrix} \in R$	RTG attribute vector where w is the current block or destination block of RTG g and τ is the time until RTG g becomes available. τ > 0 indicates that the RTG is unavailable and τ = 0 indicates that the RTG is available
$q_{t,c} = \begin{pmatrix} l_{t,c} \\ \sigma_{t,c} \\ a_c \end{pmatrix} \in Q$	Container attribute vector where l is the current location or destination location of container c and σ is the time until container c becomes available. σ > 0 indicates the container is unavailable and σ = 0 indicates the container is available. The container type is indicated by a, where 1 is an export and 0 is an import.

$$S_t = \begin{pmatrix} \left(r_{t,g}\right)_{g \in G} \\ \left(q_{t,c}\right)_{c \in C} \end{pmatrix}$$ State vector

Constraints

(1) $X_{t,g} \neq X_{t,g'}$

(2) $X_{t,g} \in \begin{cases} N_R(S_t) & \text{if } \sum_c H_{t,g,c} = 0 \text{ and } \tau_{t,g} = 0 \\ \{w_{t,g}\} & \text{otherwise} \end{cases}$ $\forall g$

Where $N_R(S_t) \subset W$ is a *neighborhood function* for a RTG; it is the set of blocks an RTG can be sent to given state vector S_t.

(3) $Y_{t,c_I'} \in \begin{cases} \{s\} & \text{if } t < t_{c_I'}^{AVAIL} \\ N_C(S_t) & \text{if } t \geq t_{c_I'}^{AVAIL},\ l_{t,c_I'} \in \{s\}, l_{t,c_I} \notin \{s\} \text{ and } l_{t,c_E} \in \{s\}\ \forall c_I \text{ and } c_E < c_I' \\ W(l_{t,c_I'}) & \text{if } \sum_g H_{t,g,c_I'} = 1 \\ \{l_{t,c_I'}\} & \text{otherwise} \end{cases}$ $\forall c_I'$

Where $N_C(S_t) \subset L$ is a *neighborhood function* for a container; it is the set of blocks a container can be sent to given state vector S_t.

(4) $Y_{t,c_E'} \in \begin{cases} \{s\} & \text{if } l_{t,c_E'} \in \{p\},\ \sigma_{t,c_E'}=0,\ l_{t,c_I} \notin \{s\} \text{ and } l_{t,c_E'} \in \{s\}\ \forall c_I \text{ and } c_E < c_E' \\ \{p\} & \text{if } l_{t,c_E'} \in B \text{ and } \sigma_{t,c_E'}=0 \\ B(l_{t,c_E'}) & \text{if } \sum_g H_{t,g,c_E'} = 1 \\ \{l_{t,c_E'}\} & \text{otherwise} \end{cases}$ $\forall c_E'$

(5) $\sum_c H_{t,g,c} \leq 1$ $\forall g$

(6) $\sum_g H_{t,g,c} \leq 1$ $\forall c$

(7) $H_{t,g,c_I} = 0$ if $w_{t,g} \neq W(l_{t,c_I})$ or $\sigma_{t,c_I} \neq 0$ or $\tau_{t,g} \neq 0$ $\forall g, c_I$

(8) $H_{t,g,c_E} = 0$ if $w_{tg} \neq l_{t,c_E}$ or $\sigma_{t,c_E} \neq 0$ or $\tau_{t,g} \neq 0$ $\forall g, c_E$

Transitions

(9) $w_{t+1,g} = X_{t,g}$ $\forall g$

(10) $l_{t+1,c} = Y_{t,c}$ $\forall c$

(11) $\tau_{t+1,g} = \max\{\tau_{t,g} + \sum_{c} H_{t,c,g} t^{HAND} + t^{RTG}_{w_{t,g}, X_{t,g}} - \Delta(t),\ 0\}$ $\forall g$

(12) $\sigma_{t+1,c} = \max\{\sigma_{t,c} + \sum_{g} H_{t,c,g} t^{HAND} + t^{CON}_{l_{t,c}, Y_{t,c}} - \Delta(t),\ 0\}$ $\forall c$

Contribution Function

(13) $F(S_t, Z_t) = \begin{cases} 0 & \text{if } Y_{t,c_I} \in W \ \ \forall c_I, \ \ Y_{t,c_E} \in \{s\} \ \ \forall c_E \ \text{ and } \sigma_{t+1,c} = 0 \ \ \forall c \\ \Delta(t) & \text{otherwise} \end{cases}$

Value Function

(14) $V_t(S_t) = \min_{Z_t} \left(F(S_t, Z_t) + V_{t+1}(S_{t+1}) \right)$

The cost function (13) describes the amount of additional time added in period *t* based on the current state. Note that when all work is complete $F(S_t, Z_t) = 0$. The value function (14) seeks to minimize the total amount of time that is spent working on the containers. Constraint set (1) ensures that all RTGs *g*,*g'* are assigned to different blocks. Constraint set (2) allows a RTG *g* to only move to blocks within its neighborhood if it is not scheduled to handle a container and if it is available at time *t*. Otherwise RTG *g* will remain in the same block. Constraint set (3) restricts the movement of import container c_I. The quay crane does not move container *c* from a container ship until the container is available in the handling sequence t_c^{AVAIL}. When container *c* is the next container in sequence, the quay crane moves the container from the container ship to a terminal tractor and the terminal tractor then transports the container to a buffer zone to await a RTG. RTG *g* moves a container *c* from the buffer area into the corresponding work block when indicated by the decision $H_{t,g,c}$. Otherwise, the container *c* remains in the same location until the next decision point. Constraint set (4) restricts the movement of a

export container c_E. RTG *g* moves a container *c* from a block *w* to a terminal tractor in buffer area *b* when indicated by decision $H_{t,g,c}$. A terminal tractor located in the buffer area *b* moves container *c* to the quay crane *p* when a container is in position to be moved. The quay crane moves container *c*, when the container is next in sequence t_c^{AVAIL} and the container is in position to be moved. Otherwise, container *c* remains in the same location until the next decision point. Constraint set (5) allows at most one container *c* to be assigned to be moved by a RTG *g* at time *t*. Constraint set (6) allows at most one RTG *g* to be assigned to move a container *c* at time *t*. Constraint set (7) does not allow a RTG *g* to handle an import container *c* if it is not collocated with the RTG or if either is unavailable at time *t*. Constraint set (8) does not allow a RTG *g* to handle an export container *c* if they are not collocated or if either is unavailable at time *t*. Transition set (9) moves RTG *g* to a block $X_{t,g}$ at time *t*. Transition set (10) moves a container *c* to location $Y_{t,c}$ at time *t*. Transition sets (11) and (12) assign work time to a RTG *g* and a container *c* respectively. The transition sets (11) and (12) are decremented by $\Delta(t)$ during the transition from *t* to the next time period *t*+1. The max function in these sets does not allow the respective τ_{t+1} or σ_{t+1} to fall below zero.

C. POLICY DEVELOPMENT

In RTGOP, solving Bellman's equation (14) with backward recursion is intractable since there are an infinite number of states and (theoretically, although not practically) an infinite time horizon. So, we develop a heuristic policy founded on rule-based policies.

A rule-based policy is derived from common operational practices. For example, "move an available RTG to the nearest container that needs to be handled", or "move a container to where a RTG is positioned," are both examples of common practices that can be implemented as a rule-based policy for handling containers and RTGs. The rules that are established determine a decision based on the endogenous information that is available at that time period.

1. Neighborhood Function Policies

Our policies are intended to restrict the set of destination areas and container choices in (3) and (4) in our DP formulation. The set-functions $N_R(S_t)$ and $N_C(S_t)$ are each called a *neighborhood function*, because they indicate a set of feasible, typically "nearby," choices that can be made for the next steps involving an RTG or a container, respectively. We try to develop policies to constrain the neighborhood functions that minimize the time to complete all work; our policies reflect this by trying to minimize the travel time of each RTG (Figure 6). We made this choice to focus on RTG travel time rather than container travel time because the travel time of a container is directly dependent on how efficiently a RTG transits to handle that container (Figure 7).

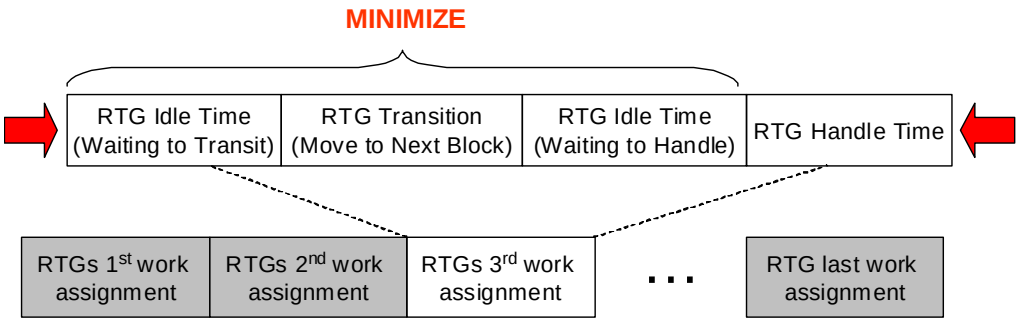

Figure 6. RTG Timeline: The sequence of jobs or block assignments for a RTG and the timeline of the assignments.

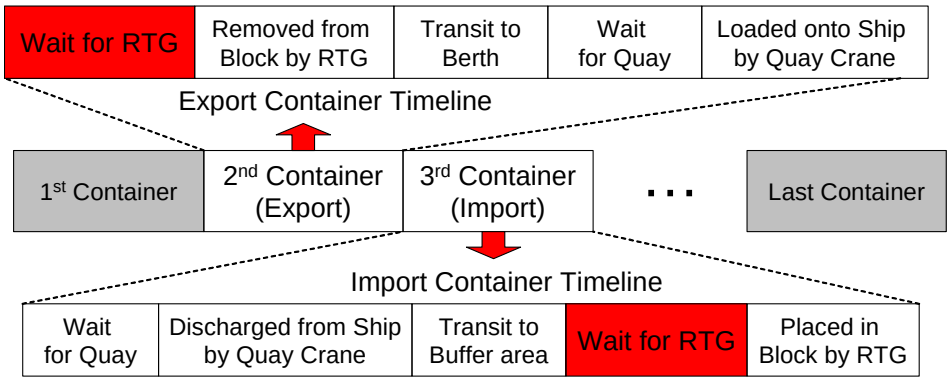

Figure 7. Container Timeline: The container sequence and the timeline of an export container and a timeline of an import container. The event in red is the containers dependency on a RTG.

a. RTGOP Heuristic Algorithm

The policies that constrain the blocks a RTG can move to are defined in the neighborhood function, $N_R(S_t) \subset W$. We define $N_R(S_t)$ as the blocks that have containers that are waiting in the buffer to be handled and do not have a RTG assigned to them. We then define feasible policies as those assignments of available RTGs to blocks in $N_R(S_t)$ that do not assign multiple RTGs to the same block during the same time period. We nominate three separate measures of performance to evaluate policies at each decision epoch *t* to enable a sequential, myopic heuristic for solving RTGOP, and then we define a modified heuristic with limited look-ahead that can hold work for a worthy unavailable RTG.

Our heuristic algorithm sequentially solves RTGOP by choosing a destination for each available RTG, one time period at a time. The heuristic starts at the first time period t=1, and, in each time period, it examines each RTG in a fixed sequence. For each available RTG *g*, the algorithm considers each block $b \in R_G(S_t)$ as a potential destination, calculates the travel time for *g* to each *b* and it saves the six closest blocks for each RTG. It then enumerates all possible selections of one block (out of the six saved blocks) per available RTG. If all six RTGs are available, this leads to 46,656 policies. The algorithm considers each of these policies and chooses the feasible policy (*i.e.*, one that does not have multiple RTGs in the same block in the next time period) that minimizes our selected measure of performance. This policy is implemented and fixed for the current period, and then the next decision epoch *t* is considered.

We introduce three separate measures of performance that guide our choice of a single policy based on the state S_t:

(1) Minimize the Sum of RTG Travel Time. For each feasible policy, the RTG travel times are summed, and we choose a feasible policy that results in the minimum sum of travel times.

(2) Minimize the Maximum RTG Travel Time. We choose a feasible policy that minimizes the maximum, over all RTGs, of their travel time.

(3) Minimize the Maximum RTG Travel Time Plus a Reward. This measure of performance has the same definition outlined in Paragraph (2) with the addition of a time reward that is subtracted from a RTG's travel time if its destination block has an export container. The export container that determines the reward is the export container that closest to becoming available t_c^{AVAIL} in the container sequence *C*. The reward is scaled according to the difference in the of the terminal's current time *t* and the time export container will become available t_c^{AVAIL}. The reward makes blocks that have an export container that is approaching its availability time look more appealing.

b. Container Neighborhood Function

The container neighborhood function $N_C(S_t) \subset L$ finds a grounding policy if the next container to become available t_c^{AVAIL} is an import container. All of the potential grounding locations of an import container are compared with each of the RTGs to find a grounding location that adds minimal travel time. The grounding location is compared to RTG's that are in two different states; a RTG that is preparing to move to a new block and a RTG that has a block assignment.

A RTG that is preparing to move has a destination block and a travel time assigned by the RTG neighborhood function. The grounding location that adds the least amount of travel time to a RTGs current travel time is the grounding location used for comparison to all other RTGs. The change in travel time is determined from comparing the original travel time to the RTG traveling to the grounding location before and after traveling to its destination block.

The change in travel time for a RTG that has a block assignment is simply the amount of time it takes the RTG to travel to the grounding location from its current block. The grounding location that has the smallest change in travel is the used for comparison to all other RTGs.

Across all the RTGs, the grounding location that has the smallest change in travel time to a RTG determines the grounding policy. If a grounding policy changes

the sequence of travel for a RTG preparing to travel, then the RTG's destination block is changed to the grounding policy block.

2. Look-Ahead Policy

In constraint set (2) of the DP, the neighborhood function does not assign a destination block to a RTG that is unavailable. However, we define a look-ahead policy that identifies how desirable it is to reserve a destination block for an unavailable RTG, *g*. When calculating *g's* travel time to a destination block, the travel time is penalized the amount of time until it becomes available, $\tau_{t,g}$. How far the policy looks into the future is the *availability window*. If the availability window is greater than a RTG unavailable time $\tau_{t,g}$, then g is allowed to participate in the performance measure using the travel time calculations described above. The destination block that is assigned to an unavailable RTG by the neighborhood function is then reserved until the next decision epoch. The availability window is defined at runtime of the ADP.

To find an appropriately sized availability window, we implemented the ADP look-ahead heuristic multiple times, solving for an availability window of zero, then one, until an availability window of thirty is reached. We then choose the best solution from this list of thirty.

3. Container Handling Policy

A RTG is not allowed to transit from a block until all work on available containers in its block or in its buffer has been completed. When handling the containers, export containers take precedence over import containers. An import container in the buffer has been discharged from the container ship by the quay crane satisfying a piece of the container sequence. An export container in the block still has a requirement to be loaded onto a container ship by the quay crane. Logically, if there are containers in a block or buffer to be handled, the export containers take precedence over the import containers due to the fact export containers still have a sequencing constraint.

D. INTEGER LINEAR PROGRAM FORMULATION

The baseline problem is formulated as an integer linear program (ILP). Like RTGOP, the ILP seeks to minimize the total amount of time until a given sequence of containers have been moved to their respective destinations within the container terminal or onboard a ship. To accomplish the task of the container sequence, the model sequences RTG movement and container handling.

The model is different from the DP in that it does not distinguish between a block and a buffer and any RTG can simultaneously occupy the same block as another RTG. We choose not to constrain the RTGs because the constraint would add another level of complexity to the model and the solution of this relaxed formulation will adequately provide a lower bound for comparison.

The formulation of the ILP is as follows, with sizes of sets given for the numerical example in Chapter IV:

Indices/Sets

$c \in C$	Ordered set of containers to be handled.	($\|C\|$=108)
$c \in C_I \subset C$	Import containers.	($\|C_I\|$=54)
$c \in C_E \subset C$	Export containers.	($\|C_E\|$=54)

We require that $C_I \cap C_E = \varnothing$ and $C_I \cup C_E = C$ in the ordered sequence *C*. We use the notation *ord(c)* to indicate the ordinal position of container *c* in the sequence *C,* and the notation *c*+1 to indicate the next container in the sequence *C* after container *c,* as long as *ord(c)*<$|C|$.

$w, w' \in W = \{1, 2, ..., \|W\|\}$	Blocks for RTGs.	($\|W\|$=36)
$g \in G = \{1, 2, ..., \|G\|\}$	RTGs.	($\|G\|$=6)
$j \in J = \{1, 2, ..., \|J\|\}$	RTG job on ordered job list.	($\|J\|$=30)

Data

$conTransitTime_{c,w}$	Time to move container *c* between the quay and block *w*.
$rtgTransitTime_{w,w'}$	Time to move a RTG between block *w* and *w'*.
$initial_{g,w}$	Binary data where 1 indicates the initial block *w* of RTG g.
quayTime	Amount of time a quay crane takes to handle a container.
handleTime	Amount of time a RTG takes to handle a container.
upperBound	A large number to bound the nonnegative variables.

Nonnegative Variables

$HS_{g,j,c}$	Time RTG *g* on job *j* starts handling container *c*.
$CQS_{g,j,c}$	Time quay crane starts to handle container *c*, on job *j* of RTG *g*.
$RTS_{g,j,c}$	Time RTG *g* on job *j* starts transit to container *c*.
CTC_c	Time when work on container *c* is completed.
TC	Time when work on all containers is completed.

Binary Variables

$X_{g,j,c}$	1 if container *c* is moved with RTG *g* on job *j*.
$P_{g,j,w,w'}$	1 if RTG *g* on job *j* is moved from block *w* to *w'*.
$G_{g,j,c,w}$	1 if container *c* is grounded at block *w* for RTG *g* on job *j*.

Objective Function

(1) min TC

s.t

(2) $\sum_{g,r} X_{g,j,c} = 1$ $\forall c \in C$

(3) $\sum_{c} X_{g,j,c} \le 1$ $\forall g, j$

(4) $\sum_{c} X_{g,j+1,c} \le \sum_{c} X_{g,j,c}$ $\forall g, j, ord(j) < |J|$

(5) $\sum_{w} G_{g,j,c,w} = X_{g,j,c}$ $\forall g, j, c \in C$

(6) $\sum_{g,j,w,w'} P_{g,j,w,w'} = |C|$

(7) $\sum_{w'} P_{g,j,w,w'} \le initial_{g,w}$ $\forall g, w$ and for $j = 1$

(8) $\sum_{w'} P_{g,j+1,w,w'} \le \sum_{w'} P_{g,j,w',w}$ $\forall g, j, w, ord(j) < |J|$

(9) $\sum_{w} P_{g,j,w,w'} = \sum_{c} G_{g,j,c,w'}$ $\forall g, j, w'$

(10) $HS_{g,j,c} \le X_{g,j,c} upperBound$ $\forall g, j, c \in C$

(11) $\sum_{g,j} HS_{g,j,c} \ge \sum_{g,j} CQS_{g,j,c} + quayTime + \sum_{g,j,w} G_{g,j,c,w} conTransitTime_{c,w}$ $\forall c \in C_I$

(12) $\sum_{c} HS_{g,j,c} \ge \sum_{c} RTS_{g,j,c} + \sum_{w,w'} P_{g,j,w,w'} rtgTransitTime_{w,w'}$ $\forall g, j$

(13) $CQS_{g,j,c} \le X_{g,j,c} upperBound$ $\forall g, j, c \in C$

(14) $\sum_{g,j} CQS_{g,j,c} \ge \sum_{g,j} HS_{g,j,c} + handleTime + \sum_{g,j,w} G_{g,j,c,w} conTransitTime_{c,w}$ $\forall c \in C_E$

(15) $\sum_{g,j} CQS_{g,j,c+1} \ge \sum_{g,j} CQS_{g,j,c} + quayTime$ $\forall c \in C, ord(c) < |C|$

(16) $\sum_{g,c} RTS_{g,j,c} = 0$ for $j = 1$

(17) $RTS_{g,j,c} \le X_{g,j,c} upperBound$ $\forall g, j, c \in C$

(18) $\sum_{c} RTS_{g,j+1,c} \ge \sum_{c} \left(HS_{g,j,c} + X_{g,j,c} handleTime \right) -$

$(1 - \sum_{c} X_{g,j,c}) upperBound$ $\forall g, j, ord(j) < |J|$

(19) $CTC_c = \sum_{g,j} HS_{g,j,c} + handleTime$ $\forall c \in C_I$

(20) $CTC_c = \sum_{g,j} CQS_{g,j,c} + quayTime$ $\forall c \in C_E$

(21) $TC \ge CTC_c$ $\forall c \in C$

The objective function (1) expresses the total amount of time that is spent working on the containers within a container terminal. Hence, the model seeks the minimum time it takes a container terminal complete all work on containers by sequencing RTG movement and the handling of containers in order to satisfy the sequence in which a quay crane must handle the containers. Constraint set (2) ensures that each container *c* is scheduled to be handled a RTG *g* on job *j*. Constraint set (3) allows at most one container *c* to be scheduled for a RTG *g* on job *j*. Constraint set (4) ensures all decisions to handle containers *c* with RTG *g* on job *j* precede jobs *j* that have no decision. Constraint set (5) ensures that the grounding decision $G_{g,j,c,w}$ for container $c \in C$ is linked with the decision $X_{g,j,c}$ to handle the container with RTG *g* on job *j*. Constraint set (6) ensures that there is a decision to move $P_{g,j,w,w'}$ for every container in the container sequence. Constraint set (7) assigns the starting block *w* of RTG *g* on its first job *j*=1. Constraint set (8) assigns the destination block w' for RTG *g* on job *j* in $P_{g,j,w,w'}$ as the starting position *w* in the next job *j*+1. Constraint set (9) assigns the ending block w' of RTG *g* on job *j* in $P_{g,j,w,w'}$ based on the decision $G_{g,j,c,w'}$ to ground a container $c \in C$ at w'. Constraint sets (10), (13) and (17) bound the positive variables $HS_{g,j,c}$, $CQS_{g,j,c}$ and $RTS_{g,j,c}$ to be zero or less than a defined upper boundary based on the value of the binary variable $X_{g,j,c}$. Constraint sets (11) and (12) coordinate the handle times for a containers $c \in C$ and a RTG *g* on job *j*. In constraint set (11), a import container's $c \in C_I$ handle start time is greater than its quay crane start time plus the quay handle time and the container's transit time. Constraint set (12) ensures that all RTGs *g* on job *j* have a handle start time greater than there transit start time plus the time to transit to the container $c \in C$. Constraint sets (14) and (15) set the lower bound for the time that a quay crane can start work on container *c*. More specifically, constraint set (14) ensures that a quay crane does not start work on an export container $c \in C_E$ until it has been handled and moved to the quay crane. Constraint set (15) ensures that timing of the container sequence is preserved. Constraint set (16) sets the initial transit start time for RTG *g* on job *j* to zero. Constraint set (18) ensures that when a RTG *g* transit start time to handle a container $c \in C$ on job *j+1* is greater than its handle start time plus its handle time on job *j*. Otherwise, the transit start time is forced to zero for all containers *c* not on

the RTG's job list *j*. Constraint sets (19) and (20) accumulate the total time that it took to complete work on an import or export container. Constraint set (21) sets the lower bound for the objective function which is seeking to minimize the total time to complete all containers.

IV. COMPUTATIONAL STUDY

A. SCENARIO DEVELOPMENT

Since RTGOP is solved with a heuristic algorithm that sacrifices accuracy for computational speed over a planning horizon, two alternate measures of performance were employed for comparison. The first measure is the baseline ILP developed in Chapter III and the second is a model developed by Akel (2007). Data and container terminal layout were chosen to remain consistent with Akel's thesis (Figure 1). The container terminal layout consists of twelve lanes which are made up of three blocks. The lanes run parallel to the ships berth and within the blocks of containers, six RTGs are in operation. The data was provided by Navis LLC which consists of over 6,000 import and export containers and the sample of containers used for numerical tests was 54 import and 54 export containers.

B. IMPLEMENTATION AND RUN TIME ANALYSIS

With the scenario developed, we implement the ADP and the ILP that were developed in Chapter III, in addition to the ILP developed by Akel (2007), for analysis. All three models were run on a Dell Precision PWS690 Intel® Xeon™ CPU 3.73GHz processor, with 3.00GB of RAM.

1. Approximate Dynamic Program

The ADP was implemented and run on Java Platform Version 1.6.0 with Eclipse: Version 3.3.1.1 as the integrated development environment.

Since we made the choice to focus on RTG travel within the yard of a container terminal, we will first look at the RTGOP heuristic output that defines the movement of RTGs. When looking at the movement of the RTGs compared to the timeline, we can see how an RTG moves around the terminal from block to block and lane to lane. Initially, the heuristic does well at keeping RTGs within a single lane or moves RTGs to nearby lanes. However, just after the 75-minute mark, all export containers have been moved out

of their blocks and are waiting to be loaded onto the container ship by the quay crane. The result of the event is that the RTGs are forced to move to different locations in the yard to handle the import containers. After the quay crane stops work, the movement of RTGs in the yard settles and RTGs finish work on the remaining import containers in the yard.

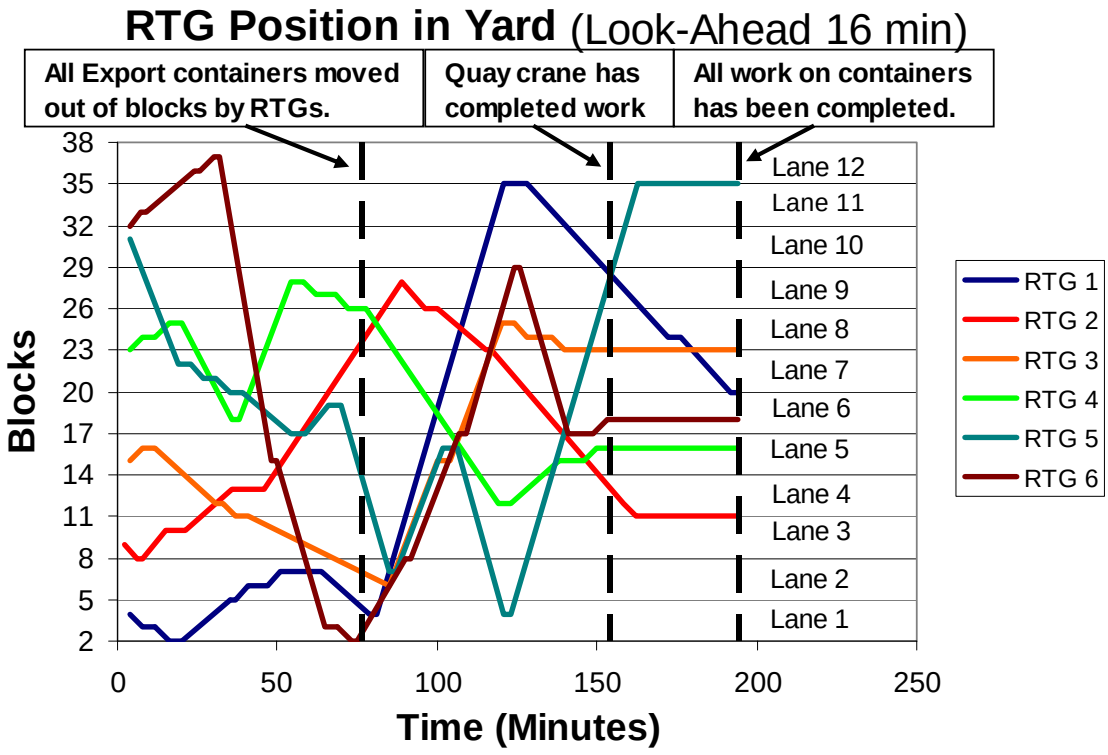

Figure 8. RTG Movement in Time (16 Minute Look-ahead): The movement of the RTGs that resulted from planning moves for RTGs that had less than 16 minutes of work.

Comparing a solution with no look-ahead to a solution with a 16 minute look ahead, we see that the RTGs are making larger moves more often with no look-ahead. The time that the RTGs finish work on the export containers is about the same, however, all the RTGs converge to lane three at about the 120 minutes in the no look-ahead solution. This convergence of the RTGs combined with the longer moves slows the quay crane such that it finishes later (Figure 9). The solutions in both Figure 8 and 9 demonstrate the importance of a look-ahead heuristic.

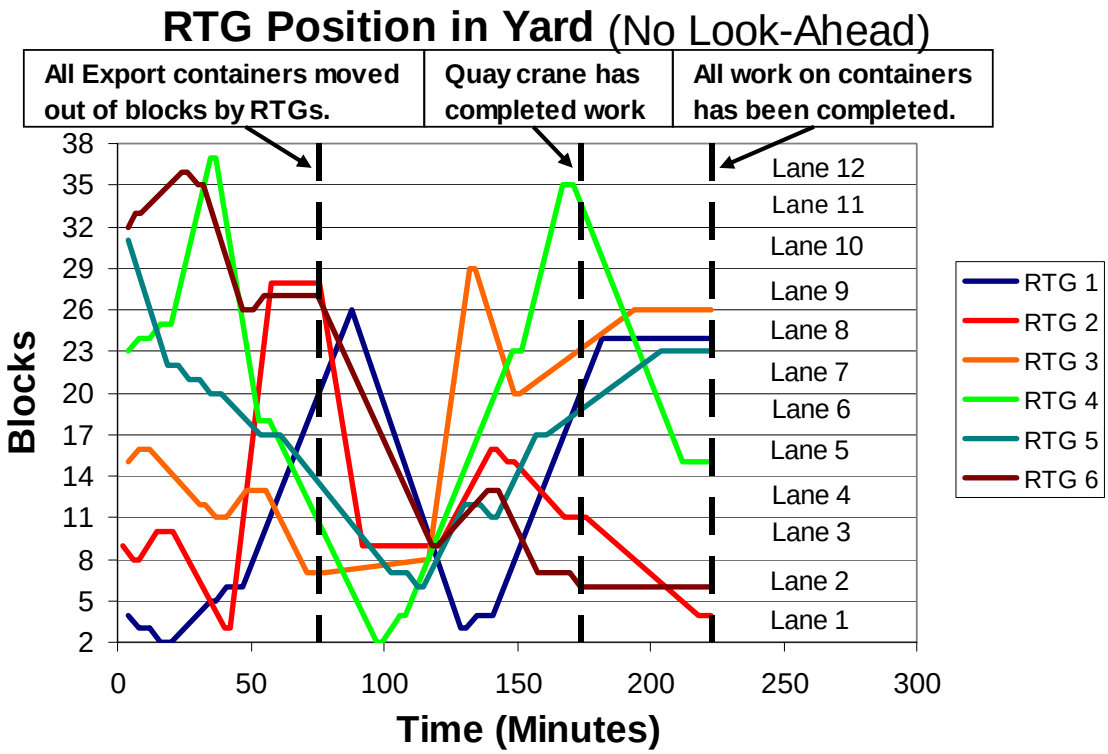

Figure 9. RTG Movement in Time (No Look-Ahead): This Chart represents no future planning of RTGs.

Within the ADP, the time to solve is largely dependant on the size of the approximate optimal value (t) and yields a linear run time of $O(t)$. With each unit of increase in the optimal value the number of computations required in the ADP is increased by approximately 47,000. The majority of the computations per iteration is attributed to the RTG neighborhood policy, making the neighborhood function the most computationally expensive piece of the ADP algorithm.

In the developed scenario for numerical trials, the number of RTGs remains at six. If the number of RTGs used in the scenario is not constant, then the run time becomes dependant on the number of RTGs. The dependency between run time and the number of RTGs is found in the neighborhood policy. The approximate number of computations in the neighborhood policy is a constant 46,700. When the number of RTGs is allowed to vary, the run time of the combinatorial search for a policy's RTG travel times becomes

$O(n^n)$ where n is the number of RTGs. Varying RTGs within the scenario was not considered and will have dramatic effects on run time.

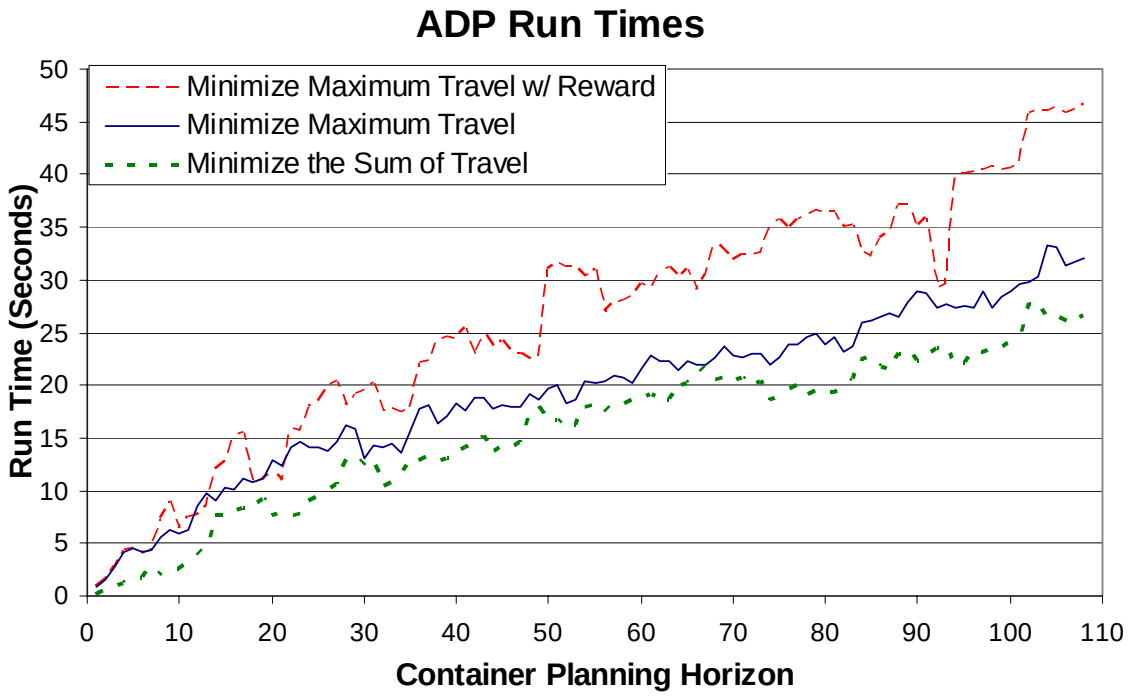

Figure 10. RTGOP ADP Run Times: For the scenario, runtimes of the neighborhood policies are compared across different planning horizons.

The run time of the ADP is extremely important when looking at the objective of real-time dispatching of RTGs. As in the formulation of RTGOP, the frequency of the actual decision epoch within a terminal is determined by the quay crane push rate. RTGOP decisions have to be determined in the time between a terminal's decision epochs. When comparing the neighborhood policies, all policies in the neighborhood function combined with a sizable planning horizon would meet the requirement of even the most aggressive quay crane push of 30 seconds (Figure 10). As we can see from Figure 10, the minimum maximum path with a penalty is just under a 30 second solve time for an 50 container planning horizon and the other two destination selection criteria solve for a 100 container planning horizon under 30 seconds.

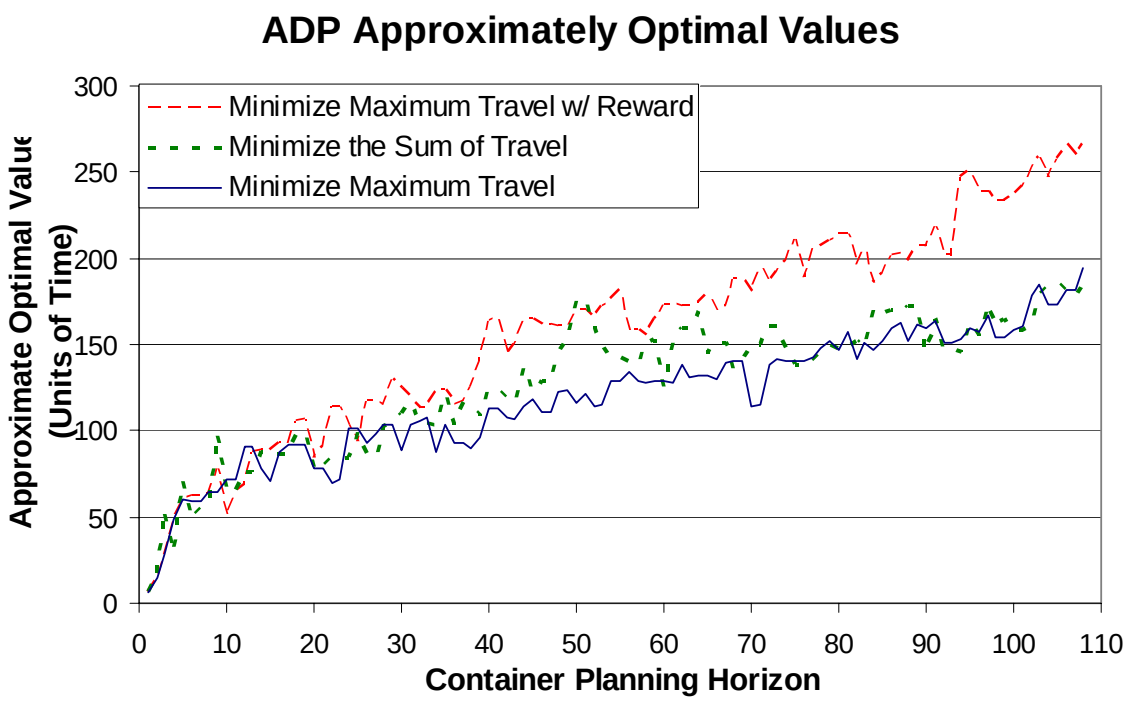

Figure 11. RTGOP ADP Approximate Optimal Values: The approximately optimal values for the different neighborhood policies are computed for the different planning horizons for comparison.

Since the run times of the different neighborhood policies are fairly comparable, the optimal values are used to distinguish their performance. We use regression analysis to identify the neighborhood policy that on average yielded the lowest optimal values. The policy that preformed the best for the given scenario and data was the minimum maximum shortest path (Figure 11). Later in this chapter, RTGOP with the minimum maximum shortest paths is used for comparison to the baseline models.

2. ILP Analysis

Both integer linear program models were implemented in GAMS and found optimal or near-optimal solutions using the CPLEX solver version 10.0.1(ILOG, 2007).

a. Baseline ILP from Chapter III

Since the computational complexity of the baseline ILP for the movement of RTGs within a container terminal is Nondeterministic Polynomial-time hard, the

number of containers and RTG moves that could be practically calculated for comparison is small. The number of exact solutions we were able produce on our timeline was nine. The solve time for a nine container planning horizon was over 19 hours (Figure 12). For a ten container planning horizon, the CPLEX algorithm ran for 48 hours resulting in a solution with a 15.79% relative gap. Based on the fact that run times were getting progressively worse, it was not practical to proceed further in this direction.

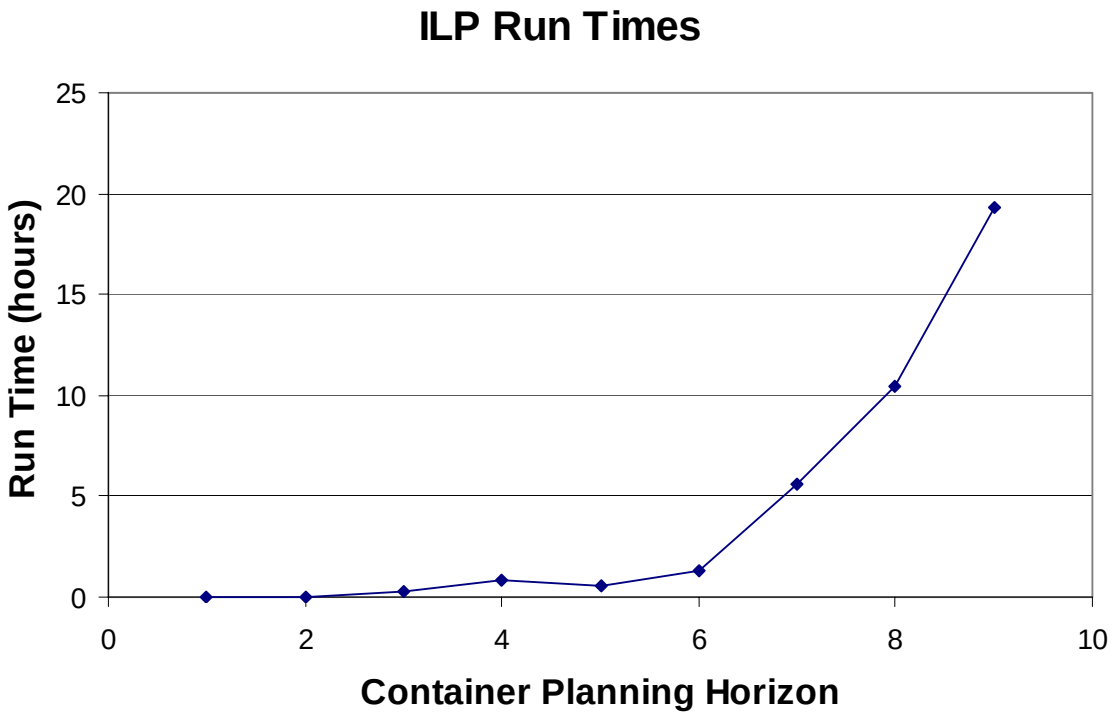

Figure 12. Baseline ILP Run Times: Run times to find an exact solution of the different planning horizons are compared. The graph shows that run time gets progressively worse as the planning horizon is increased.

The small set of optimal values in the solution of the ILP provided us with some grounds for comparison to the RTGOP ADP solutions but the quantity of solutions is insufficient. The run time behavior may prevent us from finding solutions for the entire 108 container horizon, but solution to the relaxation of the ILP model to a linear program (LP) provided us with an optimistic lower bound for comparison to the RTGOP ADP. We solved for 58 planning horizons which had a combined total solve time of over 150 hours (Figure 13). The worst case run time was twenty hours for a 56 container

planning horizon. When comparing the exact solutions of the ILP to the LP relaxation, the optimal values were exactly equal. The absolute best case would be if the exact solution continues to equal the LP relaxation for future values.

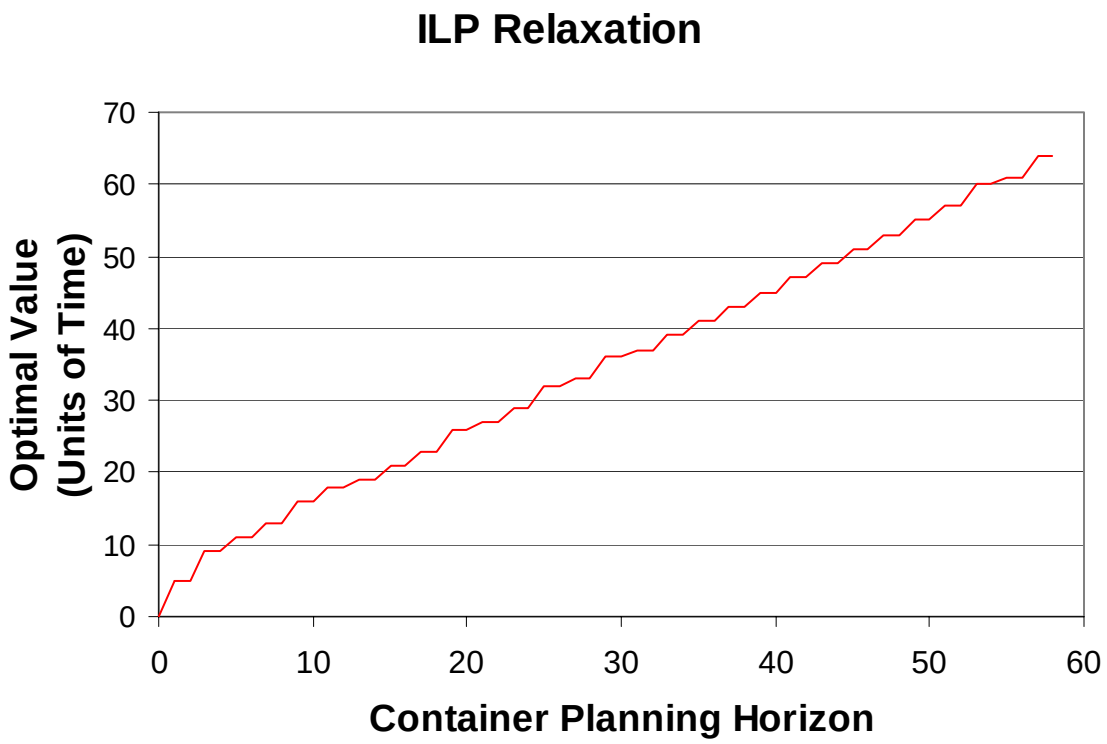

Figure 13. Optimal Values of the Baseline ILP Relaxation: The optimal values computed for the different planning horizons provide an absolute lower bound to our problem.

b. Previously Developed ILP (Akel, 2007)

In the implementation of the Akel's model, five different planning horizons were chosen for points of comparison to the ADP. The Cascade Method was the optimization based heuristic method used to solve Akel's formulation (Baker et al. 1998). In the cascade, the problem was divided into equal sub problems. The optimal solutions from the first sub problem are fixed and the subsequent sub problems are re-optimized including the fixed solutions to the prior sub problems. The process is continued until all the sub problems are solved. Each sub problem has an equally complex portion of the original problem, giving solution technique a linear run time.

The heuristic solution methodology allowed the run time to be significantly faster than our baseline ILP, finding a solution for a 60-container planning horizon within three minutes (Figure 14). However, when considering the real-time dispatching of RTGs, the model will not be able to keep up pace with a slow quay crane push rate of 90 seconds. The run time of CPLEX at a planning horizon of 10 containers is 100 seconds. The output of Akel's model provides an additional benchmark for the RGTOP ADP optimal values.

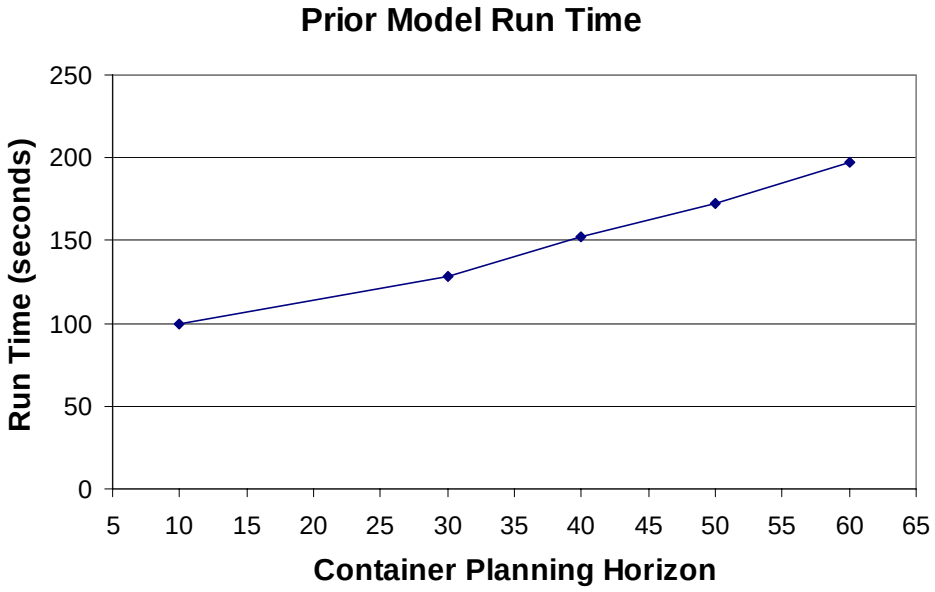

Figure 14. Prior Model Run Time (After Akel, 2007): Prior model runtime increases linearly as the planning horizon increases

C. COMPARATIVE ANALYSIS OF OPTIMAL VALUES

In the comparison of the optimal values, the three models are using the same container terminal layout, number of RTGs, and sequenced container data. Where the models are different are the constraints that are placed on container terminal operations.

The baseline model, the ILP formulated in Chapter III, is the least restrictive implementation of container terminal operations. The formulation simply establishes a grounding policy and sequences the RTG movement and handling of containers to satisfy the given container sequence. The formulation of the DP accomplishes the same task

with an addition of one restriction. RTGs are not allowed to occupy the same block. In Akel's model (2007), the formulation is much more restrictive. The RTGs were restricted in the number of moves that they could make in handling the containers, RTGs were assigned to a specific set of work blocks, all of the RTGs performed an equal measure of work and RTGs are not allowed to handle the next container in sequence if they are currently working on a container.

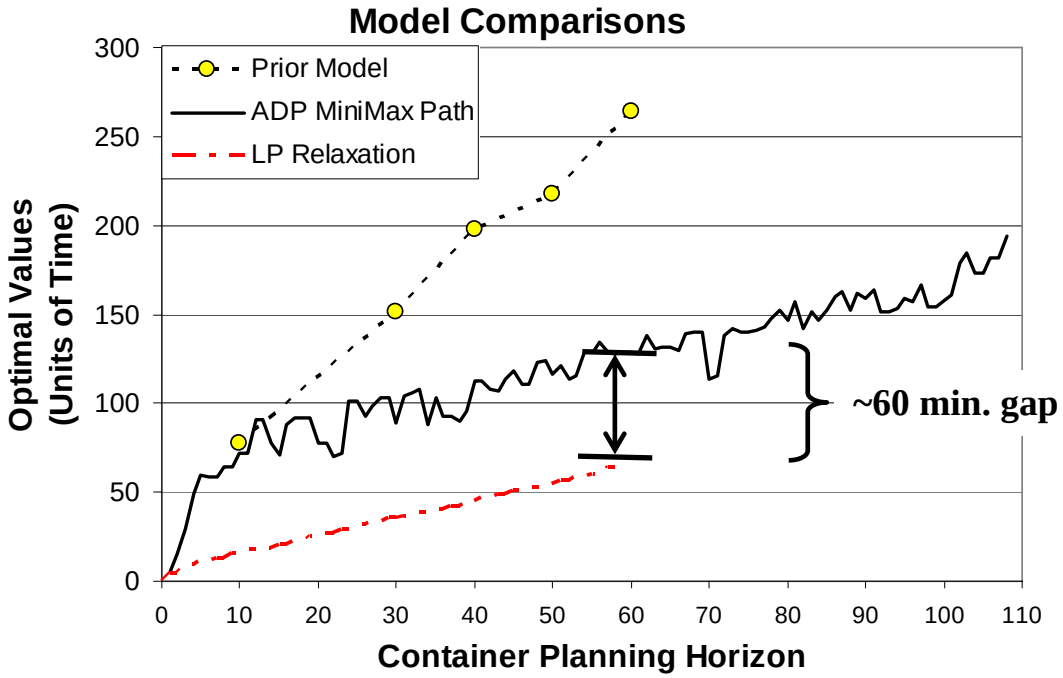

Figure 15. Comparison of Objective Values: The graph compares the optimal values and trends of the three models across planning horizons.

Being the most restrictive, Akel's model provides an upper bound on the optimal values for the ADP. On the contrary, LP relaxation provides an absolute lower bound on the optimal values for the RTGOP ADP (Figure 15). To our satisfaction, the values produced by the ADP fall in between the bounds set by the other models.

The gap between the ADP and the ILP solutions appears to be consistent, at approximately 60 minutes. For 10 hours of work, if the gap remains consistent, then our implementation provides a very effective and efficient solution to this complex problem.

Finally, we can surmise that the real value of the optimal solution is than that provided by the ILP, since the ILP is a relaxation of our problem and allows multiple RTGs to occupy the same block at the same time.

V. CONCLUSION AND RECOMMENDATIONS

A. CONCLUSIONS

The thesis is tackling the problem of real-time dispatching of assets in a container terminal as well as the grounding policy for import containers. As demonstrated in Chapter IV, the problem of optimally sequencing RTG movement and its handling of containers to meet the constraint of a container sequence is computationally expensive. Even if the problem could be solved optimally in a timely manner, an unexpected delay or machine failure can change the optimal sequence which leads to suboptimal performance. The problem would need to be resolved.

In this thesis, we develop a deterministic dynamic program for the real-time dispatching problem that can quickly be solved using approximate dynamic programming (ADP). Additionally, the ADP heuristic can be quickly resolved in the presence of data changes. Based on runtime analysis alone, it is clear that the APD based heuristic is suited for the task of real-time dispatching of RTGs in conjunction with a grounding policy. Not only do the rules within the policy of the ADP allow the program to converge to a solution set quickly, but the algorithm combined with a sizable container planning horizon has the ability to keep up with an aggressive decision making process where the quay crane push rate is 30 seconds.

As shown at the end of Chapter IV, trends of the approximately optimal values of the ADP based heuristic are closely related to the optimal values found in by the linear programming relaxation of a bench-mark ILP. This is important because it shows that as the planning horizon increases, the optimal values from the ADP heuristic and the ILP head in the same direction with very little divergence. Additionally, we surmise that the growth of the gap remain relatively consistent as the container horizon increases which gives a very optimistic outlook for large container sequences.

B. FUTURE WORK

In this thesis, we develop an ADP based heuristic algorithm that has the ability to produce real-time solutions. Additionally, we combine the grounding policy with yard asset optimization when these two have traditionally been solved separately. With the accomplishments made in this thesis, development of multiple scenarios is identified further study.

During the numerical trial of the ADP heuristic, a single scenario was developed for analyzing feasibility of real-time dispatching and initial implementations of terminal policies. Several different scenarios with unplanned delays and mechanical failures are needed for more advanced numerical trials. The multiple scenarios will aid in improving the performance of the heuristic in different terminal layouts, changing environments and large container sequences.

LIST OF REFERENCES

Akel, K., (2007). An integer linear program to combine container handling and yard crane deployment. Master's Thesis in Operations Research. The Naval Postgraduate School, Monterey, CA.

Baker, S.F. and Rosenthal, R.E., (1998). A cascade approach for staircase linear programs. July 1998. The Naval Postgraduate School Technical Report, NPS-OR-98-004.

Container Shipping Information Service, (2008). Retrieved January 28, 2008, from http://www.shipsandboxes.com/media/images/.

GAMS, (2008). Retrieved January 28, 2008 from http://www.gams.com/.

Kim, K. and Park, Y., (2004). A crane scheduling method for port container terminals. European journal of operational research, 156 (3), 752-768.

ILOG, (2007). Retrieved October 1, 2007, from http://www.ilog.com/products/optimization/archive.cfm.

Levinson, M., (2006). The Box: how the shipping container made the world smaller and the world economy bigger. Princeton, N.J.: Princeton University Press.

Linn, R. J. and Zhang, C., (2003). A heuristic for dynamic yard crane deployment in a container terminal. IIE transactions, 35 (2), 161-174.

Linn, R., Liu, J., Wan, Y., Zhang, C. and Murty, K. G., (2003). Rubber tired gantry crane deployment for container yard operation. Computers industrial engineering, 45 (3), 429-442.

Murty, K., Liu, J., Wan, Y. and Linn, R., (2005). A decision support system for operations in a container terminal. Decision support systems, 39 (3), 309-332.

Ng, W., (2005). Crane scheduling in container yards with inter-crane interference. European Journal of Operational Research, 164 (1), 64-78.

Steenken, D., Vob, S. and Stahlbock, R., (2004). Container terminal operation and operations research-a classification and literature review. OR Spektrum, 26 (1), 3-49.

Taleb-Ibrahimi, M., De Castilho, B. and Daganzo, C.F., (1993). Storage space vs. handling work in container terminals. Transportation Research B 27 (1), 13–32.

U.S. Department of Transportation. (2007). America's Container Ports: Delivering the Goods. Retrieved December 1, 2007 from http://www.bts.gov/publications/americas_container_ports/.

Vis, I. and Koster, R., (2003). Transshipment of containers at a container terminal: An overview. European journal of operational research, 147 (1), 1-16.

Zhang, C., Wan, Y., Liu, J. and Linn, R., (2002). Dynamic crane deployment in container storage yards. Transportation research. Part B, Methodological, 36 (6), 537-555.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. NAVIS LLC.
   Oakland, California

4. Assistant Professor Johannes O. Royset
   Naval Postgraduate School
   Monterey, California

5. Professor Robert F. Dell
   Naval Postgraduate School
   Monterey, California