Dynamic AGV-Container Job Deployment Strategy

Hock Chan, Sen
HPCES Programme, Singapore-MIT Alliance
E-mail: hockchan@finiq.com.sg

Abstract- Automated Guided Vehicles (AGVs) are now becoming popular in container-handling applications at seaport. Efficacy of the dispatching strategy adopted to deploy AGVs is a prime factor affecting the performance of the entire system. The objective of this project is thus to develop an efficient dispatching strategy to deploy AGVs in a container terminal.

The scenario considered was a container terminal where containers are uploaded to and discharged from ships. Discharged containers are stored at specific storage locations in the terminal yard. Containers are moved between dock and yard by a dedicated fleet of AGVs. At any point of time, each AGV carries at most two containers. This two-container load may comprise of any plausible permutation of containers for discharge or upload. To reduce congestion and increase utility level, an efficient dispatching strategy for AGVs is paramount.

At present, a variety of heuristic methods for dispatching AGVs are available, but these methods were primarily developed to work in a manufacturing context where the network structure is uncomplicated and only a small number of AGVs are required. The situation under consideration entails greater network complexity and also a large fleet of close to 80 AGVs.

In this study, the problem was modeled via network flows with constraints, which describe the disparate instances when the AGV carries one container and when it carries two. Heuristic algorithms based on this model are proposed and their performance investigated.

I. INTRODUCTION

As one of the world's leading port operators, PSA Corporation plans to automate its container transportation operations by implementing an Automated Guided Vehicle (AGV) System (AGVS) in its new container terminal premises. Typical operational and control requirements of such systems include: dispatching of AGVs to containers in the terminal, routing of AGVs and controlling of vehicular traffic in the transportation network. In this project, one particular aspect of the terminal operations is considered, that of dispatching AGVs to containers.
To design a highly efficient automated container terminal, PSA expressed the need to develop a dynamic AGV dispatching strategy to deploy AGVs to transport containers within the terminal area.

The AGV used by PSA has the capacity to carry one 40/45 feet container or one 20 feet container or two 20 feet containers. Each container job involves the loading of a container onto the AGV, the movement of the AGV to the destination of the container, and the unloading of the container from the AGV. For each container job, the following parameters are assumed:

- Exact location and impending movement route of an AGV can be accurately retrieved from the AGV Deployment System (ADS).
- Time needed to travel from each point to another point in the AGVS can be retrieved from the ADS.
- Source and destination location of all container jobs are given.
- The time for container-unloading job (container to be unloaded from a vessel) to leave the quay crane is given.
- The time for a container-loading job (container to be loaded onto a vessel) to reach the quay crane is given.
- Yard crane resources are always available.

The AGV dispatching problem entails the deployment of AGVs to serve all pending container jobs such that all imposed time constraints are met. This ensures that an AGV reaches the quay crane site before a container in time for a container to be deposited or lifted by the quay crane. If this constraint is satisfied by the deployment scheme, the terminal operates at a desired throughput rate. However, a situation whereby all AGVs queue up at the quay site and lead to traffic congestion is undesirable, hence another objective of the deployment scheme is to reduce the idle time of the AGVs at the quay site (time spent waiting for the quay crane to lift/deposit containers from/onto it).

Since an AGV can carry either one or two containers, we first consider the case whereby an AGV carries only one container at a time. This instance is termed AGV with unit capacity. Subsequently, we consider the case whereby the AGV is able to carry either one or two containers. We label this problem AGV with 2 units of capacity.

In Section II, a model developed for the problem when AGV has one unit of capacity was presented. The performance and effect on throughput are examined. In Section III the model and algorithm to solve the model for the problem when AGV has two units of capacity was discussed.

II. AGV WITH UNIT CAPACITY

In this section, it is assumed that the AGV can carry only one container at any time (i.e. it has a capacity of one).

First, the current deployment algorithm used by PSA is described. This algorithm is termed the "PMDS algorithm". Following that, the new model for this problem and its associated deployment strategy are explained. This new deployment strategy is termed the Minimum Cost Flow (MCF) algorithm. Throughputs resulting from the PMDS and MCF algorithms are then compared. Finally, simulation results are shown.

A. CURRENT DEPLOYMENT ALGORITHM (PMDS ALGORITHM)

The PMDS algorithm is an algorithm that tries to minimize the total time AGVs spend waiting to pick-up containers from their source locations.

This algorithm is best illustrated via an example. Suppose there are n AGVs and m container jobs in the container terminal. Ready times for the container jobs are displayed in Table 1.

Container job	Ready Time at the source location
1	00:30
2	00:31
3	00:36
.	.
.	.
.	.
m	00:58

Table 1: Ready time of container job 1 to m.

For a container which is to be unloaded from a vessel, the ready time is the time it is deposited by a quay crane at the quay site. For a container, which is to be loaded onto a vessel, the ready time is the time it needs to leave the yard site. This is extrapolated from the time it has to reach the quay site in order to be served by the quay crane.

From the container job list, the earliest available job, container job 1, is designated to be served first. To job 1, we then assign the AGV that is able serve whilst incurring minimal waiting time.

In Figure 1, we show all the AGVs that are able to reach the source location of container job 1 before 00:30. tr,agvi, denotes the ready time of AGV i after it serves its last job. ttr,agvi is the traveling time from location of AGV i to the source location of container job 1. tw,agvi is the waiting time that AGV i incurs if we deploy AGV i to container job 1. As shown in Figure 1, AGV 25 is able to serve container job 1 with the shortest waiting time. Hence AGV 25 is deployed to serve container job 1.

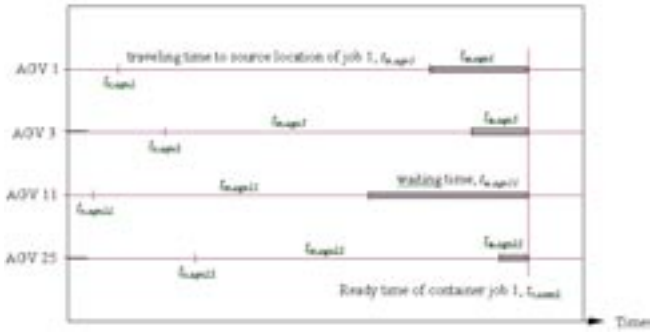

Figure 1: Chart gives the waiting of AGVs that are able to serve container job 1.

The next available job is then identified from the job list, and the AGV with minimum waiting time is deployed. The process iterates. The job list expands with time as the arrival of new vessels to the terminal necessitates the transportation of more containers or as the planner decides to schedule for more containers.

It is assumed that if an AGV reaches the source location of a container job after its ready time, it will be unable to cope with the high throughput of the quay crane and the PMDS algorithm gives an infeasible deployment solution (not all container jobs are served). However, in practical implementation, the AGV that first reaches the source location of a pending container job will be selected to serve that job.

B. NETWORK MODEL OF THE PROBLEM

In this paper, a new model that formulates the deployment situation as a network flow problem was developed. A minimum cost flow algorithm is then used to solve the problem [1].

1) Formulation of the Network Model

We assume that the problem involve n AGVs and m container jobs. A network G(N,A), with N denoting nodes and A signifying arcs, is constructed.

a) Nodes in the Network

Each container job to be served is represented by a node in network G. For each AGV, a node with a supply of 1 is inserted. The network G hence comprises n+m+1 nodes, where the $(n+m+1)^{th}$ node is a sink node with a demand equivalent to the total number of AGVs considered.

b) Arcs connect container nodes

Nodes *i* and *j* are connected by an arc a_{ij} if a single AGV can serve container jobs *i* and *j* in that sequence without violating the constraints imposed by the ready times of the jobs. This means $t_j - (t_i + t_{travel,ij})$ is greater than zero. t_i and t_j is ready time for container i and j. $t_{travelij}$ is the time needed to travel from location after serving container *i* to the location to serve container *j*, for example, if container i

is to be unloaded from the vessel and container j is to be loaded onto the vessel, $t_{travel,ij}$ is equal to the sum of the traveling time from the quay site to the destination location of container job *i*, and the traveling time from destination location of container job *i* to the quay crane that serves container job *j*.

Ready time for a container to be unloaded from a vessel is the time at which the container will be deposited by its assigned quay crane to the quay site. And the ready time for a container to be loaded onto the vessel is the time at which the container is to be picked up by its assigned quay crane.

The cost c_{ij} of arc a_{ij} is equal to $t_j - (t_i + t_{travel,ij})$, that is, the waiting time the AGV needs to wait between serving container jobs *j* and *i*.

c) Arcs connect AGV nodes to container nodes

The arc a_{ij}, where *i* is an AGV node and *j* is a container node exist, if AGV *i* can serve container job *j* on time when start from its original location. This again means $t_j - (t_i + t_{travel,ij})$ is greater than zero. t_j is the ready time of container *j*, and t_i is the time when AGV i is freed.

For container to be unloaded from the vessel, $t_{travel,ij}$ is the traveling time from the start location of AGV *i* to the quay crane that serves container job *j*. For container to be loaded onto the vessel, $t_{travel,ij}$ traveling time from the source location of AGV *i* to the source location of container job *j* plus the traveling time from source location of container job *j* to the quay crane that serves container job *j*.

The cost c_{ij} of arc a_{ij} is equal to $t_j - (t_i + t_{travel,ij})$; that is the waiting time the AGV needs to wait before it can serve container job *j* after finish serving its last container job.

d) Arcs connect all AGV nodes and container nodes to sink node

An arc links each AGV/container node with the sink node. These arcs signify that an AGV can remain idle after having served any number of container jobs or not having served at all. These arcs have zero cost.

e) Additional constraints to the network model

If AGVs are to be deployed to container jobs so as to minimize the total waiting time, the described network flow problem may be solved using a minimum cost flow algorithm. However, there is a need to include additional constraints to ensure that all container nodes are visited exactly once. These additional constraints are shown in mathematical form in section **g**.

These additional constraints can be eliminated by splitting each container node *i* into $i^{'}$ and $i^{''}$, and add an arc $(i^{'}, i^{''})$, set the upper bound and lower bound on flow traversing each arc $(i^{'}, i^{''})$ equal to 1 so that exactly one unit of flow passes through this arc, also set the cost of each arc $(i^{'}, i^{''})$ to zero.

A minimum cost flow algorithm can then be applied to the transformed network to obtain an AGV deployment with minimum total waiting time.

f) Obtain Deployment Solution From The Solved Network

Solving the transformed network model generates n paths, each of which commences from an AGV node and terminates at the sink node. The n paths visit all nodes in the network. Each path describes the container job sequence of the AGV whose node denotes the origin of the path. This deployment strategy is referred to as the MCF algorithm.

g) The Mathematical Model

Suppose there are m container jobs and n AGVs. The original network model formulated in the previous section can be expressed in mathematical form as follows,

$$\min \sum_{i,j} c_{ij} x_{ij}$$

s.t

$$\sum_j x_{ij} - \sum_j x_{ji} = 0 \qquad \forall i \in container\ nodes$$

$$\sum_j x_{ij} - \sum_j x_{ji} = 1 \qquad \forall i \in AGV\ nodes$$

$$\sum_j x_{ij} - \sum_j x_{ji} = n \qquad for\ i = sink\ node$$

$$\sum_j x_{ij} = 1 \qquad \forall i \in container\ nodes$$

$$x_{ij} \geq 0 \qquad \forall i, j$$

Where x_{ij} is the flow traversing arc a_{ij}, and *n* is the total number of AGVs in the AGVS. The first three constraints are flow balance constraints for the network, and the fourth constraint decrees that all the container nodes are served

C. UTILITY OF PMDS AND MCF ALGORITHM (SINGLE CRANE MODEL)

In this section, we assume that there is only one quay crane in the terminal serving one vessel; hence, there is only a single sequence of container jobs. All AGVs travel between the quay crane location and yard area to transport containers. The following theorem can be proved.

Let υ_{PMDS} be the utility level (throughput) of the container terminal resulting from the use of the PMDS algorithm, and let υ_{MCF} be the utility level of the container terminal resulting from using the MCF algorithm.

Theorem 1: For any loading container job sequence or unloading container job sequence, the utility levels of the container terminal obtained through the PMDS algorithm and MCF algorithm are the same, $\upsilon_{PMDS} = \upsilon_{MCF}$ [19].

This theorem holds if all container jobs in the job sequence are to be loaded onto the vessel (loading container job sequence), or all container jobs in the job sequence are to be unloaded from the vessel (unloading container job sequence), i.e. there are no combination of loading and unloading jobs in the job sequence. We first prove $\upsilon_{PMDS} \leq \upsilon_{MCF}$ and then $\upsilon_{PMDS} \geq \upsilon_{MCF}$ to arrive at the conclusion that $\upsilon_{PMDS} = \upsilon_{MCF}$.

D. UTILITY OF PMDS AND MCF ALGORITHM (MULTIPLE CRANES MODEL)

In this section, we assume that there is more than one quay crane in the terminal and thus more than one sequence of container jobs. The following theorem can be proved under the above condition,

> ***Theorem 2*** *For any container job sequence, the utility levels of the container terminal obtained through the PMDS algorithm is less than that obtained through the MCF algorithm,* $\upsilon_{PMDS} \leq \upsilon_{MCF}$ [19].

In facts, any deployment solution given by the PMDS algorithm is an instance in the solution space of the MCF algorithm for different sets of arc costs. Hence, as long as the PMDS algorithm provides a feasible deployment solution for a prescribed set of container jobs with n AGVs, the MCF algorithm is able to do the same. This suggests $\upsilon_{PMDS} \leq \upsilon_{MCF}$.

E. SIMULATION RESULT

This section is to compare the performances of the PMDS and MCF algorithms on a multiple-crane AGV dispatching problem with discharging job sequence. We first compare the disparity in total waiting times for these two algorithms for different numbers of randomly generated container jobs. Subsequently, the quay crane rate is varied, and the consequent effect on the solutions of PMDS and MCF algorithms is investigated.

1) Comparison Of Total Waiting Time

The total waiting for all AGVs given by the PMDS algorithm and the MCF algorithm is shown in Table 2. The quay crane rate is set to 30 containers per hours, and the yard crane rate is set to 24 containers per hours in the simulation. Each AGV is assumed to travel with uniform speed.

Number of Quay Crane	Number of AGV	Total Number of Container Jobs	Total Waiting Time/min	
			PMDS Algorithm	MCF Algorithm
1	4	50	111.34	71.78
1	5	50	108.39	64.57
1	4	100	265.16	175.20
1	5	100	262.00	163.19
2	8	100	189.01	88.84
2	9	100	186.66	84.92
2	8	200	418.60	250.70
2	9	200	414.76	243.27
3	12	100	123.34	69.62
3	13	100	122.06	65.92
3	12	200	329.12	157.76
3	13	200	327.68	152.79
4	16	100	127.22	53.49
4	17	100	124.41	50.47
4	16	200	278.68	134.58
4	17	200	275.85	130.83

Table 2: Comparison of total waiting time for PMDS algorithm and MCF algorithm.

From Table 2, it is observed that the solution given by the PMDS algorithm, which tries to minimize total waiting time for all AGVs, is far from optimal. On the other hand, the MCF algorithm gives the minimum total waiting time when deploying AGVs to serve container jobs. With shorter waiting imposed on the AGVs when they serve container jobs, there will be less congestion and throughput of the terminal will increase.

2) Effect Of Quay Crane Rate

In this section, we raise the quay crane rate from 30 containers per minute to 75 containers per minute, and observe the effect on the number of container jobs served later than its specified time (henceforth termed "late jobs"). The simulation result is shown in Table 3. In the simulation, there were 200 containers, 20 AGVs and 4 quay cranes.

It is observed that when there quay crane rate increases, late jobs exist for both algorithms. However, the MCF algorithm generates a deployment solution with fewer late jobs than the PMDS algorithm.

Quay Crane Rate (Containers/hour)	Number of Container Jobs Served Late over 200 Container Jobs	
	PMDS Algorithm	MCF Algorithm
30.00	0	0
33.33	0	0
40.00	0	0
50.00	0	0
54.54	6	2
60.00	17	5
66.67	30	6
75.00	45	8

Table 3: Comparison of the number of late jobs for PMDS and MCF algorithm under different quay crane rate.

III. AGV WITH 2 UNITS OF CAPACITY

In this section, it is assumed that each AGV has a capacity of two units – it is able to carry one or two containers at any time.

In current practice, two container jobs were paired and treated as a single job to which the PMDS algorithm described in the previous section was applied. Here, we propose a new model, drawing insight from section II, for this problem.

A. MODEL OF THE PROBLEM

This model consists of a network which represents all container jobs, AGVs, time constraints and constraints that ensure all container jobs are served at least once.

Any container jobs or combinations of any two container jobs (if allowed by imposed time constraints) are represented as one node in the network model. There could be as many as $O(n^2)$ number of nodes and $O(n^4)$ number of arcs in the network model, where n is the total number of container jobs. The model would be very complicated if a container could be carried from quay to yard and back to quay again and again. To simplified the network model, we would want to ensure that a container job to be loaded onto a vessel will be sent to the assigned quay crane and that once it is brought to the quay, it will never again be transported back to the yard. Containers to be discharged from vessels are treated similarly.

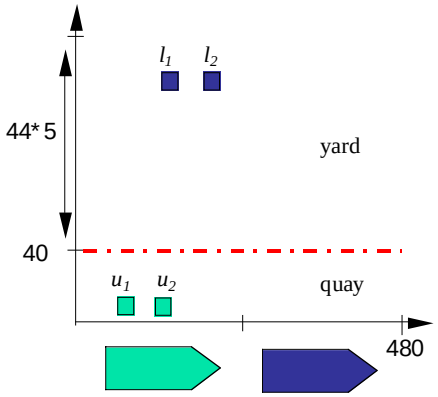

Figure 2: Virtual line separating the quay and yard

In the formulation of the model, we assume the existence of a line, the dotted line shown in Figure 2, separating the quay area from the yard area, and each container can cross this line only once. For example, once the container job u1, as shown in Figure 2, being carried by an AGV from the quay to the yard, the AGV can never return to the quay again unless container job u_1 is dropped at its destination location in the yard.

In this section, u denotes container jobs to be unloaded from vessels and l denotes container jobs to be loaded.

1) Formulation of the Network Model

Given n AGVs and m container jobs, we formulate a network to represent all container jobs, AGVs and time constraints imposed. Let $G(N,A)$ be the network, where N denotes the nodes, and A the arcs. Container jobs to be unloaded from vessels are denoted u and container job to be loaded onto vessels are denoted l.

a) Nodes in the Network

Suppose there are k u container jobs and (m-k) l container jobs (m container jobs in total), and n AGVs. There are then seven types of nodes in the network. These represent:

- AGVs; the AGV nodes.
- Containers to be loaded onto vessel; the u_i nodes.
- Containers to be unloaded from vessel; the l_i nodes.
- Groups of two containers to be loaded onto vessel; the u_iu_j nodes.
- Groups of two containers to be unloaded from vessel; the l_il_j nodes.
- Pairs of u and l-type containers; the u_il_j nodes.

When an AGV is deployed to serve a u_il_j node, it signifies that the AGV currently has container u_i on it and is waiting at the designated pick-up area for container l_j to be deposited on it. Conversely, a l_iu_j node indicates that the AGV is loaded with containers l_i and u_j and is waiting at the appropriate quay crane location for container u_j to be picked up by the crane. Similar interpretations apply to nodes u_iu_j and l_il_j.

The network has at most m^2 container nodes, n AGV nodes and 1 sink node. The number of container nodes can be reduced through preprocessing by considering the containers' time constraints. For instance, node u_iu_j cannot exist if u_j is to be served before u_i.

b) Arcs connect container nodes

There are 21 different arc types connect the container nodes in the network [19]. For example, there are arcs connecting node u_i to u_m, this means AGV send u_i to Du_i, and travel to Su_m. And there are arc connecting node u_i to l_ml_n, this means AGV will send u_i to Du_i, pick up l_m, l_n, and travel to Dl_n in shortest possible time. And there could be arc connecting node u_iu_j to l_ml_n, this means AGV will send u_i, u_j to their destination, pick up l_m, l_n and travel to Dl_n in shortest possible time.

Su_m is the source location of container job u_m, and Du_m is the destination location of container job u_m. Similar notations are used for container job l_m.

The cost c_{ab} of arc a_{ab} is the waiting time an AGV incurs at node b, having moved from node a to b, before it is allowed to proceed to the next node. AGVs typically wait for the quay cranes to deposit or remove containers. Each container has a specified time at which it is to be served by the quay crane. This depends on the crane rate.

c) Arcs connect AGV nodes to container nodes

There are 4 types of arcs connecting AGV nodes to the container nodes. There are from AGV nodes to u_m, l_m, $l_m l_n$, and $l_m u_n$ nodes. For example, arc from AGV node to $l_m u_n$ node means travel from AGV's start location to pick up l_m, and travel to Su_n in shortest possible time, and arc from AGV node to l_m node means travel from AGV's start location to pick up l_m and transport it to Dl_m.

The arc cost is the waiting period undergone by an AGV between the time it arrives at the container node and the time the quay crane deposits to or removes a container from it.

d) Arcs connect all AGV nodes and container nodes to sink node

There are arcs connecting all AGV nodes and all container node types ui, li, uilj, and uiuj to the sink node. These arcs have zero cost.

e) Additional constraints to the network model

Additional constraints have to be included to ensure that all container jobs are served at least once.

2) The Mathematical Model

Suppose there are p *u* container jobs and m-p *l* container jobs (m container jobs in total) and n AGVs. The model described in the previous section can be expressed in mathematical form as follows,

$$\min \sum_{a \in A} c_a x_a$$

s.t

$$\sum_{a \in A_i^+} x_a - \sum_{a \in A_i^-} x_a = 0 \qquad \forall i \in \textit{container nodes}$$

$$\sum_{a \in A_i^+} x_a - \sum_{a \in A_i^-} x_a = -1 \qquad \forall i \in \textit{AGV nodes}$$

$$\sum_{a \in A_i^+} x_a - \sum_{a \in A_i^-} x_a = n \qquad \textit{for } i = \textit{sink node}$$

$$\sum_{a \in B_k^+} x_a \geq 1 \qquad \forall k = 1,2,...,m$$

$$x_a \geq 0 \qquad \forall a \in A$$

A is the set of all arcs in the network. A_i^+ is the set of arcs that enter node i. A_i^- is the set of arcs that leave node i. B_k^+ is the set of arcs that enter all the nodes $x_i y_j$, where j is equal to k. x_a is the flow traversing arc a_a, and n is the total number of AGVs in the AGVS.

The first three constraints are the flow balance constraints for the network, and the fourth constraint ensures that each container job is served by at least one AGV.

B. NETWORK-BASED GREEDY (NBG) HEURISTIC

Here, a Network-Based Greedy Heuristic, motivated by the PMDS algorithm used by PSA, is proposed to generate a solution for the network model with additional constraints included.

The NBG Heuristic attempts to serve each container job while incurring minimal cost. This myopic heuristic may result in a situation whereby an AGV is compelled to serve container jobs with much higher cost giving an overall high cost.

Since there was no algorithm for deploying AGVs with two units of capacity, this Greedy Heuristic was developed to serve as a yardstick for another heuristic which is described in the following section.

C. MINIMUM COST FLOW (MCF) HEURISTIC

In this section, Langrangian Relaxation and sub-gradient method is used to find an initial solution for the network model described in above section. This initial solution is infeasible because not all container jobs are served. Working on this initial solution, the NBG Heuristic is applied to modify the solution and make it feasible.

If the fourth set of constraints shown in the mathematical formulation in the above section is relaxed via Lagrangian Relaxation, the model is transformed into a minimum cost flow problem. Suppose there are m container jobs and n AGVs, the relaxed model is shown below,

$$\min \sum_{a \in A} c_a x_a + \sum_{k=1}^{m} \lambda_k (1 - \sum_{a \in B_k^+} x_a) = \sum_{k=1}^{m} \sum_{a \in B_k^+} (c_a - \lambda_k) x_a + \sum_{k=1}^{m} \lambda_k$$

s.t

$$\sum_{a \in A_i^+} x_a - \sum_{a \in A_i^-} x_a = 0 \qquad \forall i \in \textit{container nodes}$$

$$\sum_{a \in A_i^+} x_a - \sum_{a \in A_i^-} x_a = -1 \qquad \forall i \in \textit{AGV nodes}$$

$$\sum_{a \in A_i^+} x_a - \sum_{a \in A_i^-} x_a = n \qquad \textit{for } i = \textit{sink node}$$

$$x_a \geq 0 \qquad \forall a \in A$$

$$\lambda_k \geq 0 \qquad \forall k = 1,2,...,m$$

λ_k is the dual variable for the relaxed constraint and can be interpreted as the dual price for each container job.

We use an iterative sub-gradient method to solve the above problem. During each iteration we to solve a set of shortest path problems (SPP) to obtain $(1 - \sum_{a \in B_k^+} x_a)$ for k=1, …, m in order to update λ_k. The SPPs are solved using a minimum cost flow algorithm. The sub-gradient method is run for a few thousand iterations, and the final solution is

modified using the NBG Heuristic. We refer to this solution process collectively as the MCF Heuristic.

D. SIMULATION RESULT

As a basis for comparison, small instance of the mathematical model of section II.A.2 were solved to optimality using OPL (Optimization Programming Language) Studio (an Ilog product). Larger instances of the problem were not solved with OPL due to the impractical times taken.

The solutions generated by MCF Heuristic and NBG Heuristic were compared to the optimal solutions given by OPL Studio. The total waiting times for different numbers of randomly generated container jobs are compared. Also the effects of varying the quay crane rate on the performance of both heuristic methods are investigated.

1) Comparison Of Total Waiting Time

In Table 4, the total waiting times of the deployment solutions prescribed by the Greedy Heuristic and MCF Heuristic are compared with the optimal waiting time as well as those obtained via the Lagrangian Relaxation approach. For the purposes of this series of simulations, the quay crane rate was set to 30 container jobs per hour, and the yard crane rate was set to 24 container jobs per hour. All AGVs are assumed to travel with uniform speed. Source and destination locations for all container jobs were randomly generated in all instances.

From Table 4, it is observed that the total waiting time given by the MCF Heuristic was less than that given by the NBG Heuristic. With respect to the total waiting time obtained, the MCF Heuristic performed around 25% better than the NBG Heuristic. For the MCF Heuristic, the quality of the final deployment solution depends on the number of iterations the sub-gradient method was run. If the sub-gradient method is fine-tuned to adapt to actual container terminal conditions during the implementation of these heuristics at the terminal, it is likely that the MCF Heuristic will perform much better than the NBG Heuristic.

However, when we compare the total waiting times given by the heuristics with that obtained via Lagrangian relaxation or with the optimal solution, it is evident that there is much room for improvement of the heuristics.

2) Effect Of Quay Crane Rate

In this section, we compare the performances of MCF Heuristic and NBG Heuristic when the quay crane rates are varied. Problems with different numbers of container jobs, AGVs and quay cranes were solved. Without regard for congestion effects in the container terminal, we assume that the throughput of the container terminal is proportional to the number of AGVs. In the simulation, the number of AGVs used was less than the minimum number of AGVs required to achieve the planned throughput. Hence, there were container jobs that could not be served by the system. This simulation results show the extents to which the obtained throughputs deviate from the planned throughputs for a fixed number of AGVs under different deployment

It is observed that both heuristic methods give similar numbers of container jobs not served regardless of the scenario used. However, the solution of the MCF Heuristic can be improved further by controlling the number of sub-gradient iterations. This suggests that we may achieve higher throughput for the container terminal using the MCF Heuristic.

6Situation	Waiting time (min/100)		MCF Heuristic		NBG Heuristic	
	Lagrangian Relaxation	Optimal value	Waiting time	Running time/sec	Waiting time	Running time/sec
15 *u*	1732	1779	4242	9	7325	2
15 *l*	1872	1910	4628	11	5096	1
10 AGV	2407	2460	4155	8	4809	1
1 quay crane	1889	1931	3582	6	4117	1
15 *u*	1671	Not enough memory	1942	18	3822	2
15 *l*	1755		2394	10	4006	2
30 AGV	1348		2916	12	3675	2
2 quay cranes	1720		2898	20	3667	2
30 *u*	4240	..	8605	28	10113	3
30 *l*	3871	..	8156	35	8784	3
20 AGV	4539	..	9108	29	11850	3
1 quay cranes	3587	..	8032	33	9573	3
30 *u*	4172	..	6958	75	10829	3
30 *l*	4103	..	6220	101	9642	4
30 AGV	4249	..	7018	100	9976	4
2 quay cranes	3996	..	6043	110	9103	3
30 *u*	4138	..	5055	130	7822	4
30 *l*	4329	..	6653	135	7593	4
40 AGV	4597	..	6076	136	7865	4
4 quay cranes	4484	..	6593	129	8176	5

Situation	Quay Crane Rate (Containers/hour)	Number of Container jobs that can't be served	
		MCF Heuristic	NBG Heuristic
30 *u* 30 *l* 2 quay cranes 10 AGV	30	4	5
		3	3
		2	2
		6	5
		3	4
	31.58	6	4
		5	5
		5	6
		2	2
		7	8
	33.33	8	8
		10	9
		7	8
		7	7
		8	8
40 *u* 40 *l* 4 quay cranes 20 AGV	30	4	5
		6	6
		3	2
		4	4
		4	5
	31.58	7	7
		5	4
		5	6
		5	5
		6	6
	33.33	8	7
		5	5
		6	7
		4	4
		6	7

Table 5: Comparison of the number of container jobs that can't be served for Greedy and MCF Heuristic due to lack to capacity to serve these jobs.

However, it must be added at this point that the simulations performed in the present study do not account for all intricacies in practical terminal operations. To obtain a better comparison of the relative effectiveness of the two heuristics, it is probably necessary to conduct more realistic simulations such as those which consider factors such as traffic conditions and varying AGV speed.

IV. CONCLUSION

In this project, an efficient network flow model was developed for the deployment problem in which AGVs have one unit of capacity. The deployment strategy based on this new model outperforms the current deployment strategy.

A model for the problem when AGVs have two units of capacity was also developed. Two heuristics, the MCF heuristic and NBG heuristic, were proposed to solve the model. For MCF heuristic, the Lagrangian relaxation of the model is solved using a sub-gradient method. The Lagrangian dual variable for each container job is viewed as the reward received by an AGV if it serves that container job. After solving the Lagrangian relaxation, the value of the dual variable for a container job can be subtracted from the cost to serve that container job. Subsequently, the deployment solution can be obtained through solving a series of shortest path problems. If the deployment solution thus obtained is still infeasible, the NBG heuristic is then used to assign AGVs to serve those container jobs that were not served by the shortest path solution.

V. REFERENCES

1. Andreas Löbel (2000). MCF, A network Simplex Implementation. *Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB).*

2. Chvatal, V. (1980). Linear Programming. *W. H. Freeman and Company, New York.*

3. Co, C. G., J. M. A. Tanchoco (1991). A Review of Research on AGVS Vehicle Management. *Engineering Costs and Production Economics, vol.32, pp.35-42.*

4. Dantzig, G. B. (1963). Linear programming and Extensions. *Princeton University Press, Princeton.*

5. Ebru K. Bish, Frank Y. Chen, Yin Thin Leong, Qizhang Liu, Barry L. Nelson, Jonathan Wing Cheong Ng, David Simchi-Levi (2000). Dispatching Vehicles in a Mega Container Terminal. *Northwestern University, Department of Industrial Engineering and Management Sciences.*

6. Erhan Kozan, Peter Preston (1999). Genetic algorithms to schedule container transfers at multimodal terminals. *Intl. Trans in Op. Res., vol.6, pp.331-329.*

7. Guy Desaulniers, June Lavigne, Francois Soumis (1998). Multi-depot vehicle scheduling problems with time windows and waiting cost. *European Journal Of Operation Research, vol.111, pp.479-494.*

8. Jean-Yves Potvin, Gina Dufour, Jean-Marc Rousseau (1993). Learning Vehicle Dispatching with Linear Programming Models. *Computers Ops. Res., vol.20, no.4, pp.371-380.*

9. Jean-Yves Potvin, Yu Shen, Jean-Marc Rousseau (1992). Neural Networks for Automated Vehicle Dispatching. *Computers Ops. Res., vol.19, no.3/4, pp.267-276.*

10. Jurgen Bose, Torsten Reiners, Dirk Steenken, and Stefan Vos. Vehicle Dispatching at Seaport Container Terminals Using Evolutionary Algorithms. *Abteilung Allgemeine Betriebswirtschaftslerhre, Wirtschaftsinformatik und Informationsmanagement.*

11. Michael Pinedo, Xuili Chao (1999). Operations Scheduling with Applications in Manufacturing and Services. *McGraw-Hill International Editions (Computer Science Series).*

12. M. S. Akturk, H. Yilmaz (1996). Scheduling of Automated Guided Vehicles in a Decision Making Hierarchy. *INT. J. PROD. RES., vol.34, no.2, pp.577-591.*

13. Noah Gans, Garrett Van Ryzin (1999). Dynamic Vehicle Dispatching: Optimal Heavy Traffic Performance

And Practical Insights. *Operation Research, vol.47, no.5, pp.675-692*.
14. Pius J. Egbelu, Jose M. A. Tanchoco (1984). Characterization of Automatic Guided Vehicle Dispatching Rules. *INT. J. PROD. RES., vol.22, no.3, pp.359-374.*
15. Qiu Ling, Hsu Wen-Jing (1999). Scheduling and Routing Algorithms for AGVs: A Survey. *School of Applied Science, Nayang Technological University.*
16. Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin (1993). Network Flows: Theory, Algorithms and Applications. *Prentice Hall.*
17. T.C.E. Cheng (1986). AGV Despatching in a Flexible Manufacturing System. *International Journal of Operations & Production Management, vol.7, no.1, pp.62-73*.
18. Virginie Gabrel (1995). Scheduling jobs within time windows on identical parallel machines: New model and algorithms. *European Journal of Operation Research, vol.83, pp.320-329*.
19. Sen Hock Chan (2001). Dynamic AGV-Container Job Deployment. Dissertation for Master of Science (HPC) , National University of Singapore.