Dispatching Automated Guided Vehicles in a Container Terminal

Yong-Leong Cheng, Hock-Chan Sen, Karthik Natarajan
Singapore-MIT Alliance Program

Chung-Piaw Teo*
National University of Singapore

Kok-Choon Tan
PSA Corporation

May 23, 2003

*Corresponding author. Email: bizteocp@nus.edu.sg. Research partially supported by a fellowship from the Singapore-MIT Alliance Program in High Performance Computation for Engineered Systems

Abstract

Automated guided vehicles (AGVs) are increasingly becoming the popular mode of container transport in seaport terminals. These unmanned vehicles are used to transfer containers between ships and storage locations on land. The efficiency of a container terminal is directly related to the amount of time each vessel spends in the port. Hence to maintain competitive advantage and increase the efficiency of the container terminal it is necessary to determine the appropriate number of AGVs to deploy, and formulate good dispatching strategies for these AGVs.

Existing techniques available in the literature use a variety of heuristic methods for dispatching the automated guided vehicles. These methods have been primarily developed to work in a manufacturing context where the network structure is simple and only a small number of such vehicles are required. The situation in seaport terminals however entails a greater network complexity and also a large fleet of 80 or more automated guided vehicles. There is little experimental evidence on how these techniques perform in a container terminal environment. Furthermore, for the handful of papers on AGV dispatching in a terminal environment, to the best of our knowledge, none of the proposed methods have explicitly considered the effects of congestion to determine the appropriate number of AGVs to deploy.

In this paper, we address this gap in existing research. We present a network flow formulation for the dispatching problem of the automated guided vehicles. This formulation can be efficiently solved to obtain deployment strategies for large network sizes. Furthermore, our objective is formulated in a way to minimize the waiting time of the AGVs at the berth side, reducing the possibility that the AGVs will be clustered near there. Simulation results verify that the network flow-based technique significantly increases the throughput of a container terminal. As a by-product, the simulation model can also be used to determine the appropriate number of AGVs to deploy.

Keywords: Automated guided vehicles, Container terminal, Network flow, Simulation, Vehicle scheduling, Port throughput.

1 Introduction

In 1966, the first deep-sea container service was introduced for the transport of general cargo in containers. A shipping container is a box designed to enable goods to be delivered from door to door without the contents being physically handled. The most common sizes are the 20 footers (6.1m long, 2.4m wide and 2.6m tall), and the 40 footers (12.2m long, with the same width and height as the 20 footers). Since its inception, container shipping has become a popular mode to convey products of all types, especially those of high-value. Decreasing costs, lower rates of transport, rising customer demand, and globalization of trade have caused a steady increase in the use of containers for sea borne cargo. Consequently, container terminals have become an important component of logistic networks. To satisfy customer demand, it is paramount that ships are unloaded and loaded promptly at the port. According to industry estimates (see Chan and Huat [6]), the typical operating cost for, say a Post Panamax vessel per day, can easily come to US$ 30,000 (cf. Table 1). Given the high operating cost, it is imperative that vessel operators maximize the yields and the number of voyages made by each vessel.

Table 1: Operating cost for a typical post Panamax vessel.

	US$/day
Vessel Depreciation Cost (25 years life span)	10,000
Fuel Cost (18 knots cruising speed)	10,000
Wages, Maintenance and Insurance	10,000

The above consideration necessitates the development of highly sophisticated container transportation systems, which allow for efficient movement within the container terminal area. As a result, terminal operators over the world have been increasingly pressurized to provide better and faster service to vessel operators. A major challenge in port management is thus to reduce the turnaround time of the container ships. This can be achieved in various ways:

- Deploy more quay cranes per vessel. This is however constrained by the length of the vessels and also the minimum distance required between cranes.
- Improve the handling rate of the individual cranes, by increasing the speeds and semi-automation features of the cranes.
- Improve reliability and maintainability of the cranes, so as to minimize the amount of reworks.
- Train and use skilled operators to man the cranes.

- Provide efficient yard handling and horizontal transportation systems for the loading and discharging/unloading operations.

In this paper, we will focus on the challenges posed by the last method, specifically improving the performance of the horizontal transportation system using an AGV system.

We start by providing a brief overview of the flow of operations that occur when a ship enters the port. When a vessel arrives at the container terminal for transshipment, there are two types of operations that need to be carried out. These are to discharge containers from and/or to load containers onto the vessel. Containers are first discharged from the vessel onto AGVs by quay cranes (cf. Figure 1).

Figure 1: Quay cranes loading/discharging a vessel.

The AGVs then transport the containers to specific storage locations in the yard area where they are dismounted from the AGVs by yard cranes. Typically, outgoing containers are loaded onto the ship after the majority of incoming containers have been discharged. The outgoing containers from the yard are mounted onto the AGVs using yard cranes. These containers are then carried by AGVs from the yard to the quay area where they are loaded onto the ship by a quay crane.

As mentioned earlier, containers handled by the terminal are typically of two standard sizes: twenty-footer (one TEU) or forty-footer (two TEUs). An AGV may carry a box of one TEU or two TEUs, or carry 2 boxes of one TEU each. When a container is discharged from a vessel, it is lifted by a proximate quay crane and mounted directly onto an AGV without first landing it on the ground. Landing a container onto the ground necessitates an additional crane operation to lift it from the ground and mount it later onto the AGV, thus reducing the throughput of the whole operation. In order to not delay the progress of the operations, an AGV needs to be readily present near the crane when a container needs to be loaded onto or discharged from a vessel.

The container terminal considered in this paper is based on the layout and operations of a local port operator in Singapore. As one of the world's leading port operators, it plans to automate the container transportation operations by implementing an AGV system in its newest terminal. The scale of the AGV operations in mega container terminals is typically very large, with free ranging AGVs moving in a complicated network of lanes and junctions. A complex layout of the AGV system consists of a network of lanes and junctions shown in Figure 2.

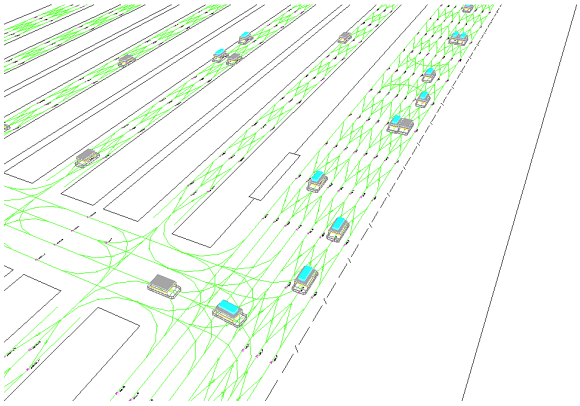

Figure 2: Layout of the terminal.

The AGVs transport containers between the quayside and yard side storage areas. These bi-directional AGVs have an advanced navigation system that guides them through the complex network transferring containers from multiple origins to multiple destinations efficiently. Typical operational, planning and control problems in such a system are: dispatching AGVs to transportation jobs, routing of AGVs and controlling traffic in the network of lanes and junctions. The dispatching module assigns each transportation job to one of the available AGVs. The dispatched AGV will then be instructed to follow the route determined by a routing module, which has details of lanes and junctions to be taken from the origin of the job to its destination.

For the sake of operation safety, the complicated network of lanes and junctions is partitioned into a large number of zones with a restrictive vehicle movement rule. Only one AGV is allowed to occupy a particular zone at any time; thus, any other AGV wishing to use the zone has to wait outside for movement clearance. As the throughput rate of an AGV system depends on the size of each zone; the bigger the zone is, the lower the rate. Typically, the minimum size of a zone is approximately equal to the distance required to stop an AGV from its top speed through the use of a normal controlled braking mechanism. The time required for stopping the AGV is generally less than 10 seconds.

Due to the dynamic nature of terminal operations, breakdown of AGVs or container handling

equipment, unexpected delays in container handling, etc., the planned route of an AGV could interfere with that of another AGV. This in turn leads to a delay in the completion time of transportation jobs involved. For example, when an AGV takes a turn, if there is a vehicle within a certain distance, it may lead to a collision. This is different from routing systems in communication networks where such physical constraints are non-existent. Such issues need to be taken care of by the navigation system along with a host of other conditions that need to be checked by a particular vehicle before it moves.

On top of the complex navigational and control problems faced in the design of such a system, we need to ensure that the AGVs are utilized in a highly efficient manner, to minimize the turnaround time of vessels in the port. Clearly having too many AGVs roaming in the network is not a cost-effective way to reduce the turnaround time of vessels. Furthermore, due to the added congestion, deploying more AGVs than necessary may in fact slow down the entire system and lead to reduced throughput.

Under this rather complex setting, we focus in this paper on developing efficient dispatching techniques that assign AGVs to container jobs. Our main contributions are as follows:

- By focusing on the work rate optimization issue associated with the quay cranes, we reformulate the AGV dispatching problem as a network flow problem. Our model is similar to the classical tanker-scheduling problem (cf. Ahuja et al. [1]), and a similar reformulation that has been reported in the literature (cf. Vis et al. [24]). While earlier models focus on finding the minimum number of AGVs needed to service the vessels (a static problem), the novel feature in our approach is the explicit formulation of waiting time minimization as our primary objective (a dynamic problem). This gives rise to a minimum-cost network flow formulation for the problem of dispatching AGVs to container jobs. For AGVs with unit capacity, solving the minimum-cost-flow network model provides an effective assignment of AGVs to container jobs. Furthermore, this model can be incorporated into a real time dynamic AGV dispatching system, since this problem can be solved efficiently in practice.

- To the best of our knowledge, none of the studies on the AGV dispatching problem in the literature explicitly consider the impact of congestion on the performance of the dispatching algorithm. Overlooking this important aspect, may lead to an erroneous conclusion that the performance will improve as more AGVs are deployed. In fact, due to the complicated zone-based navigational routines and space restrictions, the throughput of the terminal is largely dependent on the number of AGVs deployed. Using an AGV deadlock prediction package developed earlier by the group (cf. Moorthy et al. [20]), we embed the dispatching algorithm within the simulation package to examine the performance of the dispatching algorithm in a

dynamic setting. As a benchmark for comparison, we have compared our algorithm with the performance of a widely used greedy dispatching algorithm. Our simulation results show that the proposed method performs significantly better than the existing greedy heuristic used to dispatch AGVs. By carefully taking care of the effect of deadlocks and congestion caused by the AGVs, our simulation system can actually be used to obtain the optimal number of AGVs to be deployed in the system. In fact, the simulation shows that the throughput of the system suffers if too many AGVs are deployed in the system.

Structure of the paper:
In Section 2, we review some of the previous work done in the scheduling literature primarily in the seaport context. Section 3 describes our modeling approach to the AGV dispatching problem. In Section 4, we describe a greedy heuristic that has been previously proposed for these class of problems. Section 5 deals with the proposed network flow model for the problem. We discuss the connection between the two algorithms in Section 6. To address issues of network congestion, and to facilitate proper empirical performance comparison, we need to augment the vehicle-dispatching scheme with a deadlock prediction and avoidance mechanism. In Section 7, a deadlock prediction and avoidance scheme that has been implemented is described. In Section 8, we present simulation results to quantify the improvements provided by the new method. Finally, we discuss future research possibilities in Section 9.

2 Literature Review

Over the past few years there has been a huge amount of research focused on improving the design and operation of container terminals. This is in response to the enormous increase in the number of containers being used in sea transportation and the concomitant increase in the complexity of container terminal operations. For excellent reviews on the different strategic and operational issues that arise at container terminals, the reader is referred to the articles by Meersmans and Dekker [17] and Vis and Koster [23]. Scheduling of AGVs for container transport is one of the key problems identified in these papers.

Bish [4] considers an integrated problem of determining storage locations for containers along with AGV and crane allocation to minimize the maximum time taken to serve a set of ships. This problem is shown to be NP-hard and a heuristic is proposed for it. In a similar vein, Meersmans and Wagelmans [18], and [19] consider the AGV and crane allocation problem simultaneously and develop a Beam Search heuristic for this problem. While these approaches focus on joint scheduling problems, we concentrate in this paper on the AGV scheduling problem only.

With specific reference to the scheduling of AGVs, most research has been done in the context of Material Handling Systems. Co and Tanchoco [7] work with the assignment of transportation equipment to service requests on the shop floor. With assumptions of a fixed shop layout with predetermined material flow paths and fixed fleet sizes, the problem is modeled as a mixed integer program. Egbelu and Tanchoco [10] develop some heuristic rules for dispatching AGVs in a job shop environment. The heuristics are predominantly either job-based or vehicle-based. Job-based approaches develop heuristics by selecting the nearest vehicle, the farthest vehicle, the longest idle vehicle or the least utilized vehicle to serve the most tightly constrained jobs. Vehicle based approaches on the other hand try to minimize the unloaded travel times in order to maximize the opportunities for jobs to be scheduled. Shortest travel time, longest travel time, maximum outgoing queue size and first-come-first-served are some of the vehicle-based approaches considered.

Kim and Bae [13] propose mixed integer programming formulations for the AGV dispatching problem under a discrete event time setting. These event times correspond to pickup and delivery times for the containers. For a single quay crane with specified event times to be met, the problem is reduced to an assignment problem. For general cases wherein event times cannot be met, a heuristic is developed. Chen et al. [8] propose a vehicle-based dispatching strategy for a mega container terminal. The heuristic proposed deploys vehicles to the earliest possible container jobs once the vehicle is free. This vehicle based greedy heuristic does not presuppose any known information on the sequence of jobs available. Bose et al. [5] obtain an initial solution using either a job-based or vehicle-based approach and subsequently improve on it via an evolutionary algorithm. However, these algorithms only perform well for systems with small numbers of jobs and vehicles. Akturk and Yilmaz [2] propose an algorithm to schedule vehicles and jobs in a decision-making hierarchy based on mixed integer programming. Their micro-opportunistic scheduling algorithm (MOSA), combine job-based and vehicle-based approaches into a single algorithm. However, the computational time requirements for MOSA become impractical when the job number or the size of vehicle fleet is large. Using neural network models to model the decision processes of expert dispatchers is considered by Potvin et al. [21] and [22]. Vis et al. [24] consider the tactical problem of determining the number of AGVs needed at semi-automated container terminals. This paper is most relevant in our context, since they use a network flow formulation to determine the number of AGVs needed at the terminal. In this paper, with suitable modification to the cost function, we show how the method can be in fact turned into an efficient dispatching scheme.

In practical applications, besides the vehicle-dispatching problem one needs to consider the possible formation of deadlocks in the AGV system. Lee and Lin [15] and Viswandham et al. [25] use Petri-net theory to predict deadlock in material handling and AGV systems. The entire network is considered there in a matrix form. The technique needs matrix vector operations in very

large dimensions. Hyuenbo et al. [12] use graph theory to detect impending deadlock situations. To do this a large number of bounded circuits in the AGV system network needs to be found. Yeh and Yeh [26] develop efficient deadlock prediction strategies for identifying cycles in a dynamic directed graph. Developing on this work, Moorthy et al. [20] develop a prediction and avoidance scheme for cyclic deadlocks. This scheme is considered in greater detail in Section 7 since it will be incorporated into the simulation to test the performance of the proposed dispatching scheme. Duinkerken and Ottjes [9] and Evers and Koppers [11] perform simulation studies to analyze traffic control issues in AGV systems. It should be noted that to implement effective simulation studies, proper steps must be taken to ensure the accuracy of results from the model. Systematic approaches to simulation studies have been discussed by Banks et al. [3] and Law and Kelton [14].

3 Problem Description

In this paper, we focus exclusively on AGVs with unit capacity. This can be suitably modified in practice, by pairing up consecutive jobs if possible, to address the situation where each AGV can handle up to two 20 TEU containers or one 40 TEU container. This simplification, however, ensures that the problem remains tractable and that an efficient dispatching scheme can be devised and implemented in real time. In fact, most of the current literature focuses on AGVs with unit capacity that is often encountered in container terminals. Henceforth, we will only consider this situation with unit capacity AGVs.

We assume that yard crane resources are always available, i.e., the AGVs will not suffer delays in the storage yard location waiting for the yard cranes. This is not a restrictive assumption in the real implementation, since a good yard storage plan will be able to minimize the amount of congestion in a particular yard location, and hence reduce the amount of delays suffered by the AGVs. Furthermore, yard cranes are relatively much cheaper to acquire than quay cranes. Hence, yard cranes are assumed to be readily available when necessary.

To maintain a highly efficient automated container terminal, it is crucial to reduce the turn-around time in port of the container ships. This in turn is equivalent to deploying the AGVs in an effective dynamic manner. Our primary goal is to reduce the time that vessels need to spend in the port (i.e., makespan) for their loading and discharging operations. We need to deploy the AGVs such that the jobs are executed as soon as they are ready to be processed. The job ready time, however, depends on the container locations onboard the vessel, and also the speed/work rate of the quay cranes. Past research in this area has focused on finding dispatching policies so that the containers can be processed as early as possible. This, however, leads to complex scheduling problems that can only be solved optimally for special cases (involving single crane, single job type) or when

the number of AGVs to be deployed is small. Instead, we focus on the crane productivity (work rate), which is measured by the number of containers moved per hour. The problem associated with minimizing the makespan/turnaround time of the vessel can be converted to one of maximizing the work rate of the cranes. However in practice, it is not possible to arbitrarily increase the work rate of the cranes since it is constrained by the ability of the horizontal transportation system to cope with jobs. There is no point in setting a faster crane rate, since if the corresponding quayside time for a job is not met, the quay crane has to wait till an AGV comes to serve that specific job. This in turn reduces the productivity of the quay crane, and affects the deployment ready time of later jobs served by the quay crane.

For each quay crane, there is a predetermined crane job sequence, consisting of loading jobs, or unloading/discharging jobs, or a combination of both. For each loading (discharging) job, there is a predetermined pickup (drop-off) point in the yard, which is the origin (destination) of the job. Given a specified job sequence, the corresponding drop-off (for loading) or pickup (for discharging) times of the jobs at the quayside depends on the work rate of the quay cranes. For example, assuming a work rate of one container every 4 minutes (say), we need the horizontal transportation system to feed a container to the quay crane in every 4 minutes. This allows us to compute the appointment time by the quayside for each container job. To minimize the turnaround time of the vessel, we need to run the cranes at the fastest possible rate such that the AGV deployment system is still able to cope.

Our primary goal in the AGV dispatching problem is in trying to ensure that we can dispatch AGVs such that all the imposed appointment time constraints are met. Namely, we need an AGV to reach the quay crane site in time for a container to be deposited or lifted by the quay crane. If these constraints are satisfied by the deployment scheme, the terminal operates at the desired throughput rate. However, a couple of other factors that need to be taken into account in real AGV deployment systems are:

- **Congestion**: A situation whereby all AGVs queue up at the quay site can lead to traffic congestion. This is undesirable as it reduces the speed at which AGVs travel/operate, especially if there are too many near the quayside. This reduction in speed would cause the AGVs to be late for other jobs that in turn decreases the throughput of the terminal.

  To reduce congestion indirectly, we need to try to reduce the idle time of the AGVs at the quay site. This is the time spent waiting for the quay crane to lift/deposit containers from/onto it. Hence it is desirable to have the AGV arrive at the quay in a just-in-time fashion. This performance measure will indirectly reduce the number of AGVs queuing up at the quayside. Hence we are interested in finding a feasible AGV deployment that *minimizes the total waiting*

time for all the AGVs.

- **Late jobs**: Ideally, solving our model should provide a feasible deployment of AGVs such that all the jobs can be processed exactly at the quayside appointment times. However, in practice this is not possible, due to the limited number of AGVs available and traffic conditions in the network that may force some AGVs to arrive late for the jobs. In this case, we need to allow jobs to be served late. However, capturing the impact of the delays into the appointment times of all future jobs will render the model intractable. This is precisely the bottleneck in earlier approaches to this problem.

  In our model, we will allow jobs to be served late, but we ignore the delays imposed on the appointment time of all future jobs. Instead, we impose a huge penalty for the jobs to be served late, and use dynamic replanning to update the problem status in a rolling horizon format, in order to capture the impact of delays.

Terminology and assumptions:

- We consider the unit capacity case wherein each AGV can carry a maximum of one container, regardless of size, at any time.

- Let M denote the set of AGVs available, where $|M|$ is the total number of AGVs. For the AGV dispatching problem, we assume that $|M|$ is fixed and known. In fact, we show that simulation results can be used to determine the optimal number of AGVs to be deployed.

- Container jobs can be of either the discharging or loading type. Let U and L represents the set of discharging and loading jobs to be served. The total set of container jobs is represented by $N = U \cup L$ where the total number of jobs $|N| = |U| + |L|$.

- For each container job, we are specified the time at which it must be either picked up (for discharging) or dropped off (for loading) at the quayside. These predetermined times at the quay crane are chosen such that for the set of given jobs, the quay crane operates at its desired productivity level. We denote this pickup/drop-off time for job $i \in N$ by t_i and refer to it henceforth as the *quayside appointment time* for the job.

- At the epoch of (re)planning, the *AGV ready time* denoted by $\tilde{t}_m$ is known for all $m \in M$. This ready time refers to the time that the AGV is ready to start performing a new job after completing its current job. In reality, at any instant an AGV can be in one of four states - *Retrieving*, *Delivering*, *Going to park* or *Parked*. Each of these states, as the names suggest, corresponds to a different mode of operation for the AGV. As an example, consider

the calculation of $\tilde{t}_m$ for an AGV that at current time t in the state of delivering a loading job i. The distance that the AGV needs to cover the distance from its current position to the destination of job i (at the quay side) before it is ready to serve a new job. Thus $\tilde{t}_m$ is the sum of the time needed for transportation from the current position to the destination of the job i (calculated as Distance/Average velocity) and the quay crane operating time at destination of i (determined by the work rate). Calculation for other states can be done likewise.

- Let T_{ij} denote the *travel time* between two distinct jobs i and j. The calculation of travel time between jobs is illustrated in Figure 3. The AGVs are assumed to operate at a known average

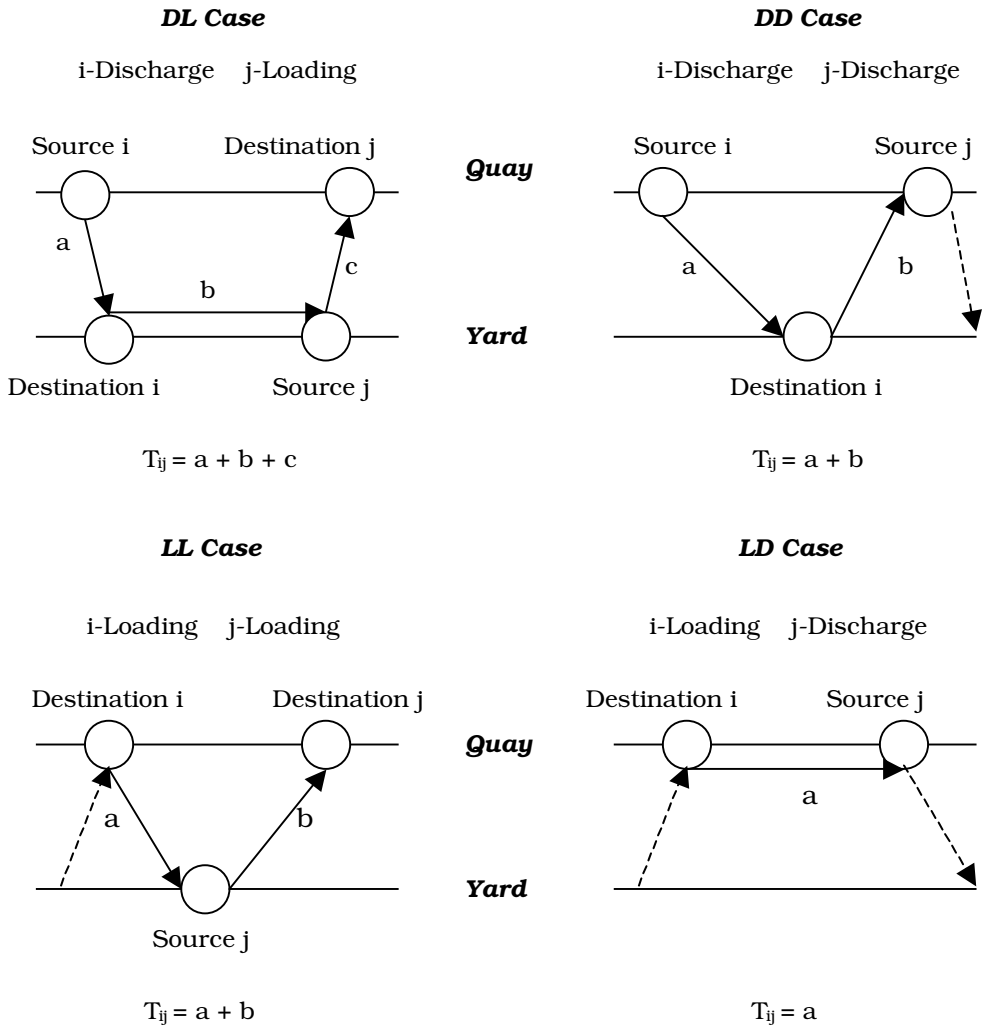

Figure 3: Travel time computation.

speed throughout the transportation operation. Clearly the computation of T_{ij} depends on

the type of job i and j. The travel times can be computed once the distance covered, and the type of operations associated with each job (discharging or loading) is known.

- Let $\tilde{T}_{mi}$ denote the travel time from the final destination of AGV m to the source of job i, at the time of deployment. This travel time can be similarly calculated based on the final destination of the job that AGV m is currently serving and the type and source of job i.

Under these assumptions, that realistically model the container terminal operations, we develop a minimum-cost network flow model to obtain an optimal dispatching strategy that minimizes the total waiting time of the AGVs at the quay cranes (cf. Section 5). It is of paramount importance that such a model should be solvable quickly in practice as is needed by the seaport container terminal in real time operations.

4 Greedy Deployment Scheme

Before we present a comprehensive framework for addressing the AGV deployment problem, we consider a simple and popular heuristic dispatching strategy that has been proposed in the literature (cf. [8, 10]) for vehicle dispatching. This strategy, which is easy to implement, has been used in at least one seaport that we are aware of. Its simplicity allows it to be used easily for AGV dispatching in a dynamic fashion. As there is no published benchmark for this class of AGV dispatching problems, we will use the greedy deployment strategy (henceforth called GD) as a benchmark to compare our network flow model with. The working of the GD algorithm is described next.

The goal of the greedy heuristic is to minimize the total time AGVs spend waiting at the quay crane locations to serve their jobs. The jobs are initially arranged in a first-in-first-out manner based on the earliest quayside appointment time t_i at each quay crane. Suppose we have already assigned a set of jobs to the AGV, and the next job in the list is considered. We first choose a list of AGVs that can reach the quay crane location in time after it has completed its previous job. From this list, we pick the AGV that will incur the minimum waiting time at the quay crane location for the job. This process is recursively performed as the jobs are scanned. This job list expands with time as the arrival of new vessels to the terminal necessitates the transportation of more containers. The GD algorithm is best illustrated with the example below.

Example 1:
Consider a terminal with $|M| = 4$ AGVs and $|N| = 4$ container jobs to be processed. The quayside times for the container jobs are displayed in the Table 2.

From the container job list, the earliest available job 1, is designated to be served first. To job 1, an AGV is assigned such that the AGV that serves it will incur the minimum waiting time. From

Table 2: Quay side time for jobs.

Job i	Appointment time t_i
1	00:30
2	00:31
3	00:32
4	00:36

Figure 4, we note that based on the current positions of the AGV only three AGVs namely 1, 2 and 4 can reach the quay crane location of container job 1 in time before 00:30. Since the waiting time for AGV 2 is minimum, AGV 2 is assigned to job 1. This procedure is performed recursively to assign the next few available jobs to AGVs.

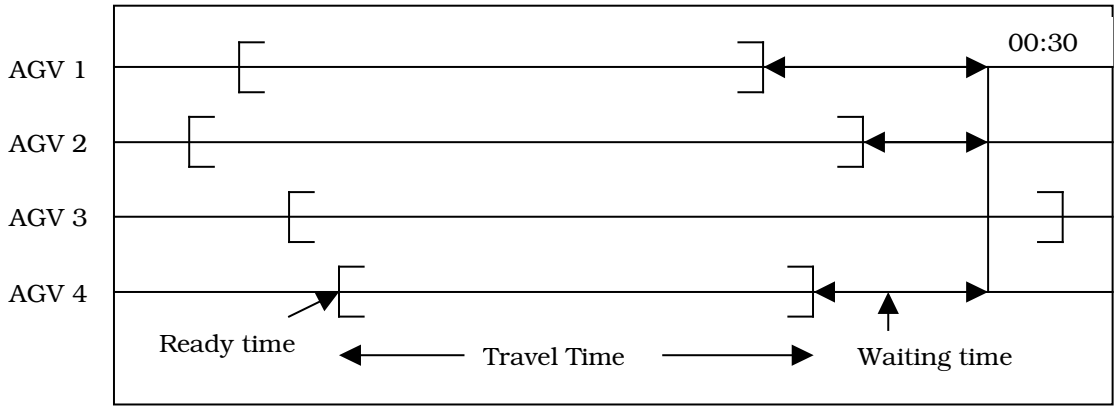

Figure 4: Waiting time of AGVs in Example 1.

Dynamic implementation:

We now describe how the greedy strategy can be implemented in a dynamic fashion for the AGV dispatching problem. In our implementation, the planning of the time to dispatch each job is done in the following manner: The first k jobs per crane will be assigned an appointed pickup/drop-off time initially. The $(k+1)^{th}$ job for each crane will only be assigned an appointment time when the service of the first job at the quay has actually been completed. The assignment of the $(k+2)^{th}$ job will depend on the completion of the second job and so on. The number k depends on when the re-planning should be done based on historical traffic condition. In our implementation, we used $k = 4$ for dynamically assigning appointment times to jobs (cf. Figure 5).

Given the time window W between jobs, the quay side time t_i $(i = 1, \ldots, k)$ for the first k jobs

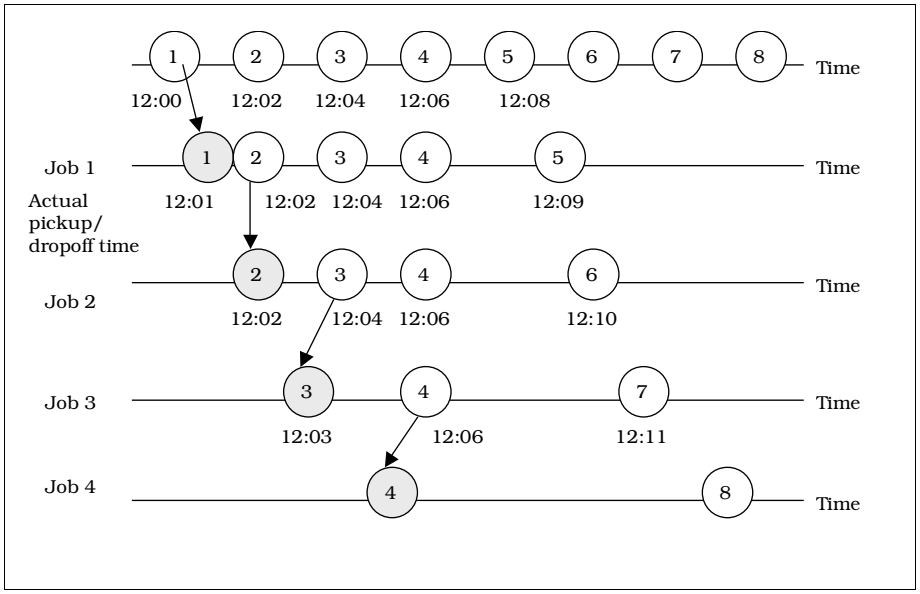

Figure 5: Dynamic assignment of appointment time under GD.

is computed as:

$$t_i = Ship - discharge - time + (i-1) \times W. \tag{1}$$

The $Ship - discharge - time$ is the time the ship is ready for discharge at the terminal. The time window W is the interval between successive discharging of containers from the vessels and depends on the quay crane work rate (i.e. number of containers moved per hour). By setting a time window of say 4 minutes (i.e. work rate of 15 containers per hour), the operator hopes to work the quay crane at a rate of one container for every 4 minutes.

For any subsequent job i after the first k jobs, the quay side time is computed as:

$$t_i = C_{i-k} + k \times W. \tag{2}$$

where C_{i-k} denotes the actual completion time of the $(i-k)^{th}$ job. Note that C_{i-k} is available at the time of deployment of the i^{th} job. Thus given that the ready time for an AGV m is $\tilde{t}_m$, the waiting time for AGV m to serve job i is calculated as:

$$t_i - (\tilde{t}_m + \tilde{T}_{mi}). \tag{3}$$

Recall that at any instant an AGV can be in one of four states - *Retrieving*, *Delivering*, *Going to park* or *Parked*. In our implementation of the GD scheme, for a particular job, only AGVs in the

last three status are considered for assignment. An AGV in the first state is on its way to retrieving a job and hence will not be considered for assignment. For each state of the AGV, the AGV ready time and the travel times are explicitly computed.

In practice, there will be instances where none of the AGVs can be deployed in time to serve a particular job. In such situations, the AGV that first reaches the quay crane location of a pending container job i will be selected to serve that job, even though the AGV cannot arrive before the appointment time t_i.

5 Network Flow Method

We now propose a network flow model to solve the AGV dispatching problem. The optimal deployment of the AGVs such that the total waiting time is minimized is found by solving a minimum-cost network flow problem. Such problems can be solved efficiently in practice even for large-scale networks [1], making the proposed method extremely attractive. A detailed description of the model is presented next. Our model can be viewed as an extension of a model proposed by Vis et al. [24], where our goal is to find a schedule that will minimize the impact of delays and maximize the utilization of the AGVs. In this regard, we find that the objective to minimize the total AGV waiting time at the quay is a reasonably good surrogate of the penultimate goal.

We construct a directed graph $G(V, E)$ to represent the complete network where V denotes the set of nodes and E denotes the set of arcs. The graph is constructed in the following manner. Every container job $i \in N$ (discharging and loading) is represented as a node in G. There is a node corresponding to each AGV $m \in M$, capturing the state of the AGV at the time of deployment. We also insert one sink node s corresponding to the end state of the AGVs, after all jobs have been served. Thus we have a total of $|N| + |M| + 1$ nodes in the network, denoted as:

$$V = N \cup M \cup \{s\}.$$

To define the (directed) arcs in the network, we introduce the following notation. We call an ordered pair of distinct jobs (i, j) *compatible* [24] if a single AGV can be used to serve job j (arriving at the quay side before t_j) after serving job i. Hence job pair (i, j) is compatible if:

$$t_i + T_{ij} \leq t_j.$$

Similarly an ordered pair (m, i) of AGV m and job i is compatible if:

$$\tilde{t}_m + \tilde{T}_{mi} \leq t_i.$$

The three types of arcs that connect the various nodes in this directed graph are:

- There exists a directed arc from AGV m to a container job i if the AGV and job pair (m, i) is compatible. The cost of such an arc (m, i) corresponds to the waiting time that the AGV m incurs at the quay crane location of job i if it is used to serve job i immediately after finishing the initial job. We assume that the AGV travels at certain predetermined average speed for this computation. Hence:

$$\textit{Cost between AGV node } m \textit{ and job node } i,\ c_{mi} = t_i - (\tilde{t}_m + \tilde{T}_{mi}) \quad \forall (m, i) \textit{ compatible.} \quad (4)$$

- There exists a directed arc from container job i to container job j if job pair (i, j) is compatible. The cost of such an arc (i, j) is the waiting time that the AGV incurs if it serves job j immediately after serving job i. Hence:

$$\textit{Cost between job node } i \textit{ and job node } j,\ c_{ij} = t_j - (t_i + T_{ij}) \quad \forall (i, j) \textit{ compatible.} \quad (5)$$

- There exists a directed arc from each of the AGV nodes and the container nodes to the sink node s. 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 are assigned a cost of zero. Hence:

$$c_{ms} = 0 \ \ \forall m \in M, \qquad c_{is} = 0 \ \ \forall i \in N. \quad (6)$$

However in the practical implementation, it may not be possible to process the jobs within the specified time restrictions. To obtain a feasible solution in such cases, we introduce arcs between job pairs (i, j) that are not compatible, i.e., when:

$$t_i + T_{ij} > t_j.$$

Such arcs are highly unattractive as it decreases the quay crane productivity. Hence we weigh the cost of such arcs with a large penalty value K as:

$$c_{ij} = K(t_i + T_{ij} - t_j) \quad \forall (i, j) \textit{ not compatible.} \quad (7)$$

Similar arcs with high costs are introduced between AGVs and jobs for pairs that are not compatible where the AGV reaches the quay crane location of the job after the quayside time. The set of arcs in the graph is hence denoted as:

$$E \ = \ \{(m, i) : m \in M, i \in N\} \ \cup \ \{(i, j) : i, j \in N, i \neq j\} \cup \ \{(m, s) : m \in M\} \ \cup \ \{(i, s) : i \in N\}.$$

The AGV dispatching problem then corresponds to finding $|M|$ directed paths in this network (one for each AGV starting from its node and ending at node s), visiting all job nodes once, at

minimum cost. Let x_{ij} represent the flow on arc (i, j). Mathematically, the problem is formulated as:

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{i\in V:(m,i)\in E} x_{mi} = 1, \quad \forall m \in M, \\
& \sum_{i\in V:(i,j)\in E} x_{ij} = 1, \quad \forall j \in N, \\
& \sum_{i\in V:(j,i)\in E} x_{ji} = 1, \quad \forall j \in N, \\
& \sum_{i\in V:(i,s)\in E} x_{is} = m, \\
& x_{ij} \in \{0, 1\}, \qquad \forall (i, j) \in E.
\end{aligned} \tag{8}$$

The above problem can be transformed to a network flow problem, using the following well-known node-duplication technique (cf. Ahuja et al. [1]):

- Split each container job node $i \in N$ into two nodes i' and i'' and add an arc (i', i''). We thus expand the number of container job nodes from $|N|$ to $2|N|$. Let N' and N'' denote these two new sets of container job nodes. Hence the set of nodes in the expanded network is:

  $$V' = N' \cup N'' \cup M \cup \{s\}.$$

  The set of the arcs in this expanded network is:

  $$\begin{aligned}
  E' = \; & \{(m, i') : m \in M, i' \in N'\} \cup \{(i'', j') : i'' \in N'', j' \in N', i \neq j\} \cup \{(m, s) : m \in M\} \\
  & \cup \{(i'', s) : i'' \in N''\} \cup \{(i', i'') : i \in N\}.
  \end{aligned}$$

- We set the upper bound and lower bound on the flow traversing through each arc (i', i'') to 1 so that exactly one unit of flow passes through it. The lower and upper bounds on all other arcs are set to 0 and 1 respectively. We let l'_{ij} and u'_{ij} denote the lower and upper bound for each arc (i, j) in the graph.

- The cost of the newly introduced arc (i', i'') is set to zero. Transforming the arc costs from the original model to the new model we obtain:

  $$\begin{aligned}
  c'_{mi'} &= c_{mi}, \quad \forall m \in M, i \in N, \\
  c'_{i''j'} &= c_{ij}, \quad \forall i, j \in N, i \neq j, \\
  c'_{ms} &= c_{ms}, \quad \forall m \in M, \\
  c'_{i''s} &= c_{is}, \quad \forall i \in N, \\
  c'_{i'i''} &= 0, \quad \forall i \in N.
  \end{aligned}$$

The purpose of this transformation, shown in Figure 6 will ensure that each job is served by one AGV.

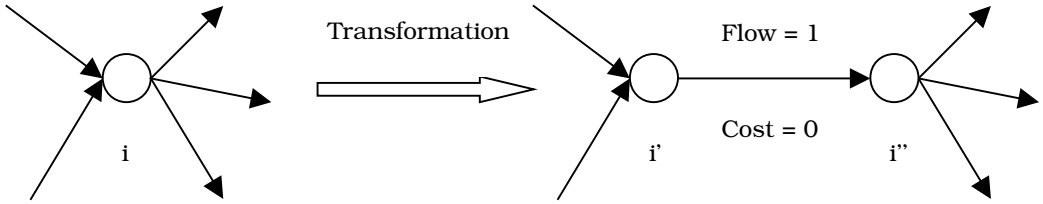

Figure 6: Transformation of the network.

Formulation (8) is thus reformulated as a minimum cost network flow model as:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in E'} c'_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{i\in V':(m,i)\in E'} x_{mi} && = \quad 1, \quad \forall m \in M, \\
& \sum_{i\in V':(i,j)\in E'} x_{ij} - \sum_{i\in V':(j,i)\in E'} x_{ji} && = \quad 0, \quad \forall j \in N' \cup N'', \\
& \sum_{i\in V':(i,s)\in E'} x_{is} && = \quad m, \\
& l'_{ij} \le x_{ij} \le u'_{ij}, && \quad \forall (i,j) \in E'.
\end{aligned}
\tag{9}
$$

The first three constraints in Formulation (9) are standard flow conservation constraints while the last constraint provides upper and lower bounds on the flow values. It is well known the linear programming relaxation of the capacitated minimum cost network flow problem can be solved in polynomial time to yield optimal integral flows. Furthermore specialized algorithms such as the network simplex method [16] can be used to solve large-scale problems extremely efficiently in practice. Solving the network flow model generates $|M|$ paths, each of which commences from one AGV node and terminates at the sink node s. In totality the $|M|$ paths cover all the nodes in the network only once. Each path from a source AGV node to the sink node prescribes the container job sequence that the AGV should be assigned to. This deployment strategy is referred to as the Minimum Cost Flow (MCF) algorithm from hereon.

Dynamic implementation:

In the practical implementation, one needs to consider the effects of uncertainty of traffic conditions on the job assignment. In a prescribed job assignment, some of the jobs could be late due to interruptions and this lateness will affect the rest of the later jobs exponentially. Moreover, the

solution might not be optimal due to the change of the job status. Thus, re-planning needs to be done frequently. Hence re-planning is done for each crane after every k number of jobs have been deployed. At that instant, a new MCF problem will be formulated based on the number of jobs remaining, the latest status of all jobs and AGVs. Following the GD model, k is selected to be 4. An example of the assignment of the appointed time sequence for 9 jobs for a single crane is illustrated in Figure 7.

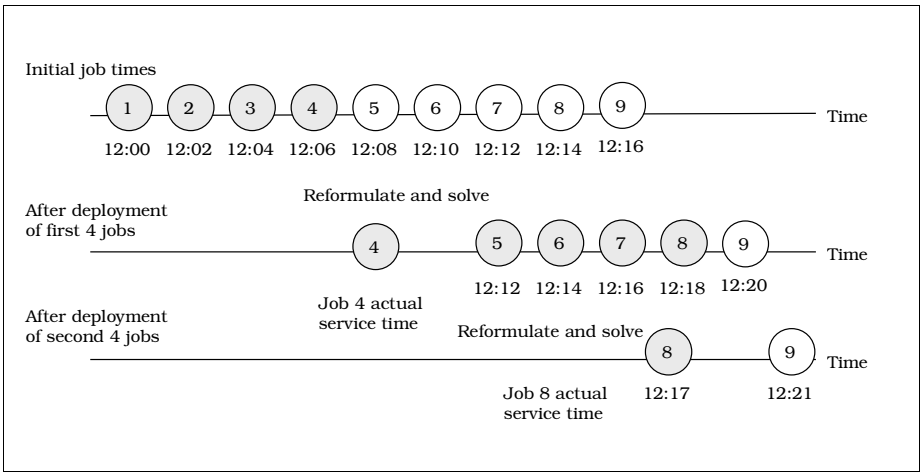

Figure 7: Dynamic assignment of appointment time under MCF.

Note that unlike the GD algorithm, the MCF algorithm uses the expected appointment time information of all future jobs to assign the next job to the AGV. The main advantage is that by doing so, the system is able to anticipate problems that may arise if the AGVs are deployed greedily. However, the solution obtained is dependent on the expected appointment time of all future jobs and could be adversely affected if there are delays in serving certain jobs. Hence it is not apriori clear, whether the dynamic implementation of the MCF algorithm is indeed superior to the GD algorithm.

6 Performance Comparison

6.1 Single Crane Scenario

The GD algorithm can be viewed as a heuristic way to solve the minimum-cost network flow problem, since it tries to design $|M|$ paths from the AGV nodes to the sink s in the network, albeit in a greedy fashion. We first focus on the single crane case when there is only one job type (either

discharging or loading jobs, but not mixed). In this case, we show that the GD algorithm actually gives rise to the optimal minimum-cost flow solution.

Theorem 1 *For a single quay crane model with sequences of one type of job, either loading or discharging, the solution obtained by GD is as good as the solution obtained by the MCF algorithm.*

Proof. We consider only the case when the jobs are all discharging jobs. The case when all jobs are loading jobs can be handled using a similar argument.

Consider the optimal solutions provided by the MCF and GD algorithms for the same set of discharge jobs N. Suppose both algorithms prescribe identical AGV deployment solutions for the first $(k-1)$ container jobs where $(k-1) < |N|$, and suppose they differ in their assignment of the k^{th} job. Let AGV p be assigned to job k by MCF while AGV q is assigned to the same job by GD where p and q are distinct vehicles.

Since MCF has selected AGV p to serve job k, it means that AGV p will reach the source of job k (i.e., the quay side, since all jobs are discharging job) before its appointment time. Furthermore, since GD uses AGV q to serve the same job, we conclude that AGV q will arrive at the quayside of the crane later than AGV p, but before the appointment time of job k.

Let r be the next discharging job, after job k, assigned to AGV q in MCF. Note that now both AGVs arrive before the appointment time of both the jobs k and r. Hence we can interchange the assignments, i.e., using AGV p to serve r and jobs after r served by AGV q previously, and AGV q to serve job k and subsequent jobs served by AGV p, without increasing the total waiting time of the two AGVs. In this way, we obtain a new solution to the minimum-cost network flow problem such that the assignment of the first k jobs are identical to algorithm GD.

By repeating this process for job $(k+1)$ to $|N|$, we can transform an optimal solution for the minimum-cost network flow problem into a solution identical to that given by the GD algorithm without affecting the total waiting time. Hence GD solves the problem optimally in this special case. ■

Note that the assumption that the jobs are all of the same type is crucial for the above to hold. For example, if job r is a loading job, then the fact that AGV q can be used to serve r (i.e., bringing the container from yard to the quay side before the appointed time) does not guarantee that AGV p can also be used to serve r, although AGV p will arrive at the quay side of job k earlier than AGV q. This happens if AGV q is nearer to the source of job r (in the yard) than AGV p.

6.2 Multiple Crane Scenario

The performance of GD algorithm however deteriorates considerably in a multiple crane scenario. We demonstrate this with a simple simulation experiment.

Consider a multiple crane AGV dispatching problem with 4 quay cranes. The total number of container jobs is set to 200. Twenty AGVs are used to process these jobs. The container jobs are randomly generated. The quay crane rate is increased from 30 containers per hour to 75 containers per hour. The yard crane rate is set to 24 containers per hour. Each AGV is assumed to travel at an uniform speed. Table 3 displays the simulation results obtained for this problem. For the quay

Table 3: Effect of quay crane rate on waiting time and late jobs.

Quay Crane Rate (Containers per hour)	Waiting time for GD (Minutes)	Waiting time for MCF (Minutes)
30.00	255	104
33.33	208	108
40.00	212	90
50.00	155	80
54.55	6 late jobs	2 late jobs
60.00	17 late jobs	5 late jobs
66.67	30 late jobs	6 late jobs
75.00	45 late jobs	8 late jobs

crane rate up to 50 containers per hour, it is observed that both the MCF and the GD algorithms provide feasible deployment solutions. Up to this quay crane rate, AGVs can be deployed such that jobs are processed at the quay when it is ready. The total waiting time for the MCF algorithm is however observed to be as much as fifty percent lesser for these quay crane rates. For the higher quay crane rates, it is observed that both AGV deployment strategies cause some of the container jobs to be late. Clearly, the number of late jobs for the MCF algorithm is much lesser than the number of late jobs for the GD algorithm. This simple experiment clearly shows that the performance of the MCF algorithm is significantly better than the GD algorithm in the multiple crane scenario.

7 Deadlock Prediction and Avoidance Algorithms

The deployment schemes obtained from the GD and the MCF algorithms provide dispatching rules for the automated vehicles. In practice, however constrained space near the quayside locations and fixed paths for AGVs cause deadlocks to occur. Such deadlocks cause part of or the entire system to stall, further delaying job processing. Hence it is essential to integrate deadlock prediction and avoidance algorithms with the dispatching algorithms.

The AGV system consists of a complicated network of lanes and junctions that is partitioned into zones. For operational safety at most one vehicle is allowed to occupy each zone. However it is possible for a dispatching algorithm to create a cyclic deadlock in the AGV system [20]. This generic form of deadlock occurs when a chain of vehicles is formed where each vehicle requests for a zone such that we have a cycle. A cyclic deadlock for four AGVs is shown in Figure 8. To account for such situations, we need to integrate a deadlock prediction strategy and an avoidance algorithm with the proposed AGV dispatching algorithm to test its merits.

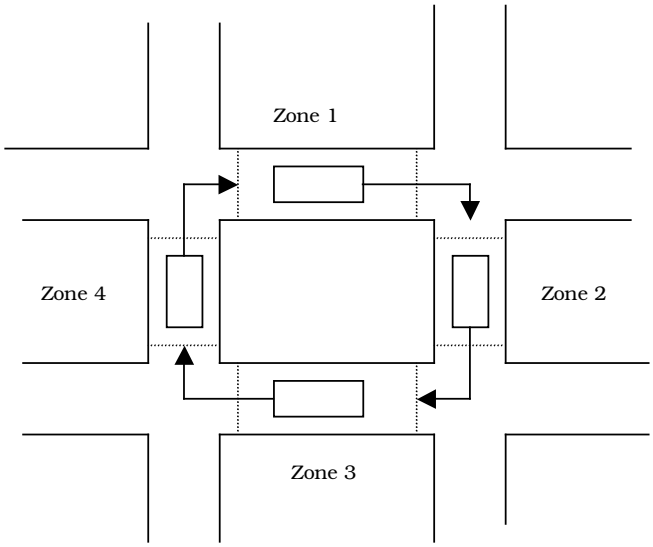

Figure 8: Cyclic deadlock in zone control AGV system.

7.1 Deadlock Prediction Strategy

In practice, the sample time for the control system of the AGV is very small in the range of 1.5 to 2 seconds. Hence we need to ensure that the deadlock prediction strategy is extremely quick and effective. As noted in Section 2, previously proposed methods are either too complicated or computationally expensive to use for this application.

We now describe the basic idea of the deadlock prediction algorithm that is used in our model [20]. For every sampling instant of about 2 seconds, a check is done to see if a specific vehicle i has moved to a new zone. For each vehicle that changes zones, a deadlock prediction is performed for its next step. If the next zone is occupied by a different vehicle j, then the vehicle must wait for the zone to clear. However if the next zone is not occupied, we still need to make sure that moving into the next zone will not lead to a deadlock that will cause the system to stall. To do this, we look further and examine the next destination of the AGV after the next zone (which is called the next-next zone). If this zone is unoccupied, then the vehicle can proceed as no deadlock can occur. However if the next-next zone is occupied by vehicle k, then the next zone location of vehicle k is found. The same check is performed to see if this next location is occupied. If not, then no deadlock is predicted. Else we proceed and check if we create a cycle and come back to the original next zone of vehicle i. We have a cyclic deadlock in that case. The worst-case complexity of this algorithm is $O(|M|^2)$ where $|M|$ is the total number of AGVs in the network. This worst-case scenario occurs in case there is a huge cyclic deadlock involving all the vehicles. This algorithm is an one-zone-step deadlock prediction method. This technique is seen to be effective in practice from [20].

An example of a cyclic deadlock that is predicted by the algorithm is illustrated in Figure 9. Here, AGV 1 request to enter it's next zone 2, which is currently free. However by looking at the next-next zone of the AGVs, it can be verified that a cyclic deadlock is formed.

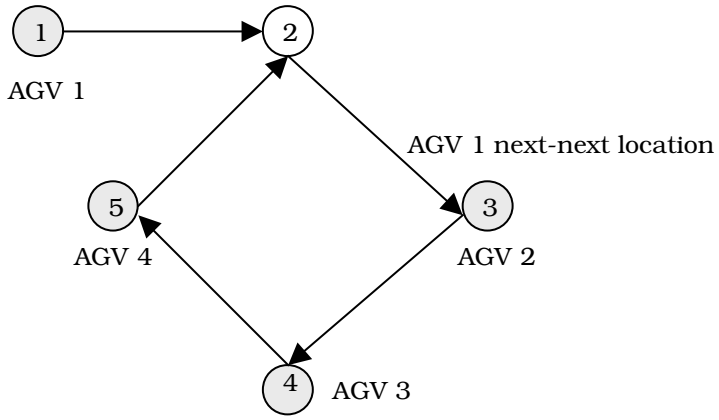

Figure 9: Example of cyclic deadlock forming.

7.2 Deadlock Avoidance Strategy

The deadlock avoidance scheme is basically a wait and proceed strategy. If a deadlock is predicted on a vehicle route, it will stop and wait until at least one vehicle is cleared from the region in which the deadlock is predicted. However implementing the one-zone step deadlock prediction and the wait and proceed avoidance algorithm might cause other deadlock situations to occur. This form of deadlock is formed because of multiple loops sharing only one unoccupied node (cf. Moorthy et al. [20]). An example of such a deadlock is provided in Figure 10 where a cyclic deadlock is avoided by the wait and proceed strategy but another deadlock is created which has many cycles that share a single empty resource.

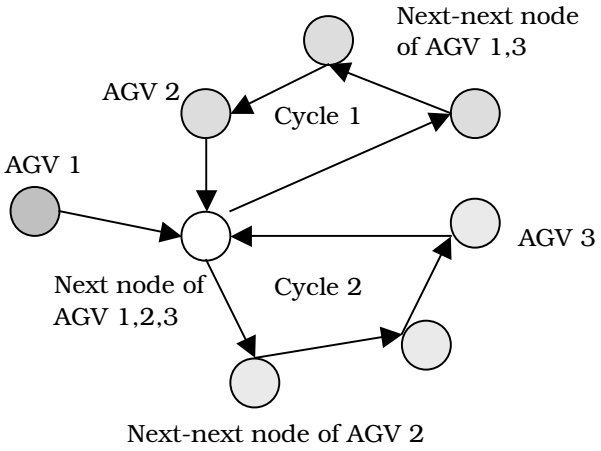

Figure 10: Example of multi-cycle deadlock forming.

Moorthy et al. [20] proposed a forward-arc strategy to resolve these deadlocks. For detailed simulation studies that test and evaluate the performance of these deadlock prediction and avoidance strategies, the reader is referred to [20]. Our primary interest is to integrate these deadlock prevention strategies with the deployment models.

8 Simulation Studies

The results obtained from the simulations performed to compare the performance of the MCF and the GD algorithms are provided next.

The simulation study is performed using a discrete event simulation software AutoMod, V 9.0 [27]. The entire system is modeled in terms of its state at each point in time, entities that pass through the system and events that cause the state to change. Software included with AutoMod

is AutoView to perform 3-dimension animation and AutoStat to do statistical analysis. For the simulation, the performance of the GD and the MCF algorithm integrated with the one-zone-step deadlock prediction and avoidance algorithms are compared.

Model assumptions:

- The layout of the terminal consist of 4 berths and 16 quay cranes with 4 quay cranes assigned to each berth.
- We consider AGVs with unit capacity and vary the number of AGVs in the system between 40, 60 and 80.
- Berths are randomly assigned to incoming ships and the arrival of the ship is assumed to follow an exponential distribution of mean 60 minutes.
- Each container storage yard is made up of 9 clusters wherein each cluster has 3 control points. At any one time, a quay crane for either discharging or loading process can only use a single cluster. In the real application, it is possible to move the quay cranes but the movement was not simulated here.
- The distribution of workload of each quay crane is set to 18%, 25%, 27% and 30% respectively for the first to fourth quay crane.
- The actual quay crane rate is set to a triangular distribution of (1.375, 1.708, 2.113) minutes. The yard crane rate is set to a triangular distribution of (1.593, 2.172, 2.728) minutes. The crane average operating rate is taken to be the average of the 3 given values of the triangular distribution. This average operating rate is used in the computation of solutions for both the MCF and the GD algorithms.
- The average velocity of each AGV is important for calculating expected travel times between two points. To realistically determine this average velocity, the simulation model was run a few times before the actual simulation by varying the number of AGVs. The collected statistical data is used to set the average velocity of the AGV in both the models. Based on this data, the average velocities used in meters per second were 3.573, 3.616 and 3.770 for 40, 60 and 80 AGVs respectively.

For a given set of system parameters, the simulation was run for a deterministic period of 4 days. In our first simulation, we evaluate the effect of varying the number of AGVs in the container terminal, maintaining the time window between jobs constant.

Variation in the number of AGVs:

The time window between the jobs, W was fixed at 2 minutes. We compare the performance of the MCF and GD algorithms by varying the number of AGVs in the container terminal. The results obtained over a period of 4 days is provided in Table 4.

Table 4: Effect of varying number of AGVs on performance.

	GD			MCF		
	40 AGV	60 AGV	80 AGV	40 AGV	60 AGV	80 AGV
Number of boxes	17769	20499	21797	18793	21471	22825
Average makespan	7.792	6.213	6.369	7.433	5.910	6.050
Average throughput	53.346	66.297	64.678	58.798	72.660	71.183

The first row in Table 4 measures the total number of container jobs that are served over the entire period. Clearly, for an identical number of AGVs the deployment scheme provided by the MCF algorithm serves more jobs than the GD algorithm. Similarly for the remaining two measures: the average makespan of the ship (i.e. the duration that a ship remains in the terminal for loading and unloading operations) and the throughput (i.e. the number of boxes processed per hour) the MCF algorithm significantly outperforms the GD algorithm.

From Table 4, it is observe that as the number of AGVs is increased from 40 to 60, there is a significant increase in throughput due to the increase in amount of available resources. However increasing the number of AGVs from 60 to 80, in fact decreases the throughput. The deadlock effects caused by AGV congestion for a constant layout of the berths is the primary reason for this. Based on the simulation, in fact we can estimate the optimal number of AGVs to be deployed in the system, by explicitly consider the effects of congestion. For the current system, about 4 to 5 AGVs per crane per berth seems to be optimal.

The observed mean deviation in time from the appointed times is provided in Table 5. Ideally

Table 5: Effect of varying number of AGVs on mean time deviation.

	GD			MCF		
	40 AGV	60 AGV	80 AGV	40 AGV	60 AGV	80 AGV
Early Jobs (min)	3.609	2.290	2.235	2.993	2.078	1.850
Late Jobs (min)	6.589	4.848	5.747	4.518	4.026	4.773

we would like the AGVs to have zero waiting time. If we are late then we decrease the quay crane productivity and if we are early we cause congestion. Clearly from Table 5, the MCF algorithm on the average outperforms the GD algorithm with respect to the total waiting time. The improvement in deviation of the waiting time for the MCF algorithm over the GD algorithm is in the range of 15 to 30 percent.

Variation in the time window:
In the second simulation, we evaluate the effect of varying the time window between jobs, maintaining the total number of AGVs constant. The number of AGVs is held constant at 80 over 4 days while the time window W for the jobs is varied. Other specifications for the 4-berth model remain the same as before. The time between jobs is varied between 1.8, 2 and 2.5 minutes. Similar to the previous simulation, we measure the total number of container jobs that are served, average makespan and the throughput of the terminal. The results are displayed in Table 6. Clearly, the

Table 6: Effect of varying duration of time window on performance.

	GD			MCF		
	1.8 min	2 min	2.5 min	1.8 min	2 min	2.5 min
Number of boxes	21333	21797	20726	22618	22825	21762
Average makespan	6.889	6.369	6.533	6.577	6.050	6.200
Average throughput	63.264	64.678	63.558	70.186	71.183	70.202

throughput for the MCF model is yet again about 10% higher than the GD algorithm. As the time window increases from 1.8 to 2 minutes, the throughput for both the algorithms increase. However on increasing the time window from 2 to 2.5 minutes, the throughput for both the algorithms decrease. This is because for the tight time window of 1.8 minutes, more jobs will be late as the AGVs cannot keep up with the crane productivity level. However for a loose time window of 2.5 minutes the AGVs reach the quay crane before the job is ready, causing congestion and hence decreasing the throughput on a whole.

These results clearly indicate that the proposed MCF algorithm outperforms the GD algorithm. An increase of as much as 10% in throughput is observed. Furthermore the tradeoff between faster processing and increased congestion caused by increasing the amount of AGVs is evident. We can in fact, estimate the number of AGVs to deploy at the container terminal based on this tradeoff.

9 Conclusions and Future Research

A container terminal must operate efficiently to ensure that the time in port for seaport vessels are reduced. This in turn entails formulating efficient dispatching strategies to load and discharge containers from the vessels. In this paper, we focused on finding efficient deployment strategies for AGVs to perform these operations. Current techniques use a simple greedy heuristic to solve this problem. In this paper, a new technique based on a network flow formulation is proposed for the dispatching of AGVs with unit capacity. This technique has two distinct advantages. One is that it provides an optimal solution to the problem if the goal is to minimize the total waiting time of AGVs. The improvement in the performance in seen to be significant from the simulation studies. Furthermore this solution can be computed extremely efficiently in practice even for a large AGV network. Thus the proposed algorithm is extremely suitable for the AGV deployment problem in complex and large seaport container terminals.

The network flow technique proposed is applicable for AGVs with unit capacity as is the case in many automated terminals. In the particular terminal considered, the AGV has the additional feature of carrying two units of load. An efficient model to obtain dispatching strategies under the multiple load capacity is an interesting and open problem. Extensive simulation studies for the case of multiple unit capacity need to be performed to compare the performance relative to the unit capacity case. Another feature of our model is that it is completely deterministic. However in practice there may be some randomness involved especially in travel times of AGVs and the processing times of jobs. Efficient models to solve such stochastic models is another area of future research.

References

[1] Ahuja, R.K., T.L. Magnanti, J.B. Orlin. 1993. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall.

[2] Akturk, M.S., H. Yilmaz. 1996. Scheduling of automated guided vehicles in a decision making hierarchy. *International Journal of Production Research.* **34**, 2, 577-591.

[3] Banks, J., J.S. Carson, B.L. Nelson, D.M. Nicol. 2001. *Discrete-Event System Simulation.* 3rd Edition, Prentice Hall.

[4] Bish, E.K. 2003. A multiple-crane-constrained scheduling problem in a container terminal. *European Journal of Operational Research.* **1441**, 83-107.

[5] Bose, J., T. Reiners, D. Steenken, S. Voβ. 2002. Vehicle dispatching at seaport container terminals using evolutionary algorithms. *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences.* R.H. Sprage (Ed), IEEE, Piscataway, 1-10.

[6] Chan, C.T., L.H. Huat. 2002. *Containers, containerships and quay cranes: a practical guide.* Singapore: Genesis Typesetting & Publication Services.

[7] Co, C.G., J.M.A. Tanchoco. 1991. A review of research on AGVS vehicle management. *Engineering Costs and Production Economics.* **32**, 35-42.

[8] Chen, F.Y., E.K. Bish, Y.T. Leong, Q. Liu, B.L. Nelson, J.W.C. Ng, D. Simchi-Levi. 1998. Dispatching vehicles in a mega container terminal. INFORMS, Montreal, Canada.

[9] Duinkerken, M.B., J.A. Ottjes. 2000. A simulation model for automated container terminals. *Proceedings of the Business and Industry Simulation Symposium.* Washington, ISBN 1-56555-199-0. ISCS.

[10] Egbelu, P.J., J.M.A. Tanchoco. 1984. Characterization of automatic guided vehicle dispatching rules. *International Journal of Production Research.* **22**, 3, 359-374.

[11] Evers, J. J. M., S. A. J. Koppers. 1996. Automated guided vehicle traffic control at a container terminal. *Transportation Research A.* **30**, 1, 21-34.

[12] Hyuenbo, C., T.K. Kumaran, Richard, A. Wysk. 1995. Graph theoretic deadlock detection and resolution for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation.* **11**, 3, 413-421.

[13] Kim, K.H., J.W. Bae. 2000. A dispatching method for automated guided vehicles to minimize delays of containership operations. *International Journal of Management Science.* **5**, 1, 1-25.

[14] Law, A.M., D.W. Kelton. 1991. *Simulation Modeling and Analysis.* 2nd Edition, McGraw-Hill.

[15] Lee, C.C., J.T. Lin. 1995. Deadlock prediction and avoidance based on Petri nets for zone control automated guided vehicle systems. *International Journal of Production Research.* **33**, 12, 2349-3265.

[16] Lobel, A. 2000. MCF - A network simplex implementation version 1.2. *http://www.zib.de/Optimization/Software/Mcf/index.html.*

[17] Meersmans, P. J. M., R. Dekker. 2001. Operations research support container handling. *Econometric Institute Report EI 2001-22.*

[18] Meersmans, P.J.M., A.P.M. Wagelmans. 2001. Effective algorithms for integrated scheduling of handling equipment at automated container terminals. *Econometric Institute Report EI 2001-19.*

[19] Meersmans, P.J.M., A.P.M. Wagelmans. 2001. Dynamic scheduling of handling equipment at automated container terminals. *Econometric Institute Report EI 2001-33.*

[20] Moorthy, R.L., H.G. Wee, W.C. Ng, C.P. Teo. 2003. Cyclic deadlock prediction and avoidance for zone controlled AGV system. *International Journal of Production Economics.* **83**, 3, 309-324.

[21] Potvin, J.Y., G. Dufour, J.M. Rousseau. 1993. Learning vehicle dispatching with linear programming models. *Computers and Operations Research.* **20**, 4, 371-380.

[22] Potvin, J.Y., Y. Shen, J.M. Rousseau. 1992. Neural networks for automated vehicle dispatching. *Computers and Operations Research.* **19**, 3/4, 267-276.

[23] Vis, I.F.A., R.de Koster. 2003. Transshipment of containers at a container terminal: an overview. *European Journal of Operational Research.* **147**, 1-16.

[24] Vis, I.F.A., R.de Koster, K.J. Roodbergen, L.W.P. Peeters. 2001. Determination of the number of automated guided vehicles required at a semi-automated container terminal. *Journal of the Operational Research Society.* **52**, 409-417.

[25] Viswanadham, N., Y. Narahari, T.L. Johnson. 1990. Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models. *IEEE Transactions on Robotics and Automation.* **6**, 6, 713-723.

[26] Yeh, M.S., W.C. Yeh. 1998. Deadlock prediction and avoidance for zone control AGVs. *International Journal of Production Research.* **36**, 10, 2879-2889.

[27] AutoMod V 9.0 Reference Manual.