Simulation Study of a Semi-Dynamic AGV-Container Unit Job Deployment Scheme

Yong Leong, Cheng
HPCES Programme, Singapore-MIT Alliance

Abstract- Automated Guided Vehicle (AGV) Container-Job deployment is essentially a vehicle-dispatching problem. In this problem, the impact of vehicle dispatching polices on the ship *makespan* for discharging and/or loading operations is analyzed. In particular, given a storage location for each container to be discharged from the ship and given the current location of each container to be loaded onto the ship, the problem is to propose an efficient deployment scheme to dispatch vehicles to containers so as to minimize the *makespan* of the ship so as to increase the throughput. The *makespan* of the ship refers to the time a ship spends at the port for loading and unloading operations. In this paper, we will compare the performance of current deployment scheme used with the new proposed deployment scheme, both with deadlock prediction & avoidance algorithm done in previous study [1]. The prediction & avoidance algorithm predicts and avoids cyclic deadlock. The current deployment scheme, namely *pmds* makes use of a greedy heuristics which dispatches the available vehicle that will reach the quay with the minimum amount of time the vehicle has to spend waiting for the crane to discharge/load the container from/onto the ship. The new deployment scheme, namely *mcf* aims to formulate the problem as a minimum cost flow problem, which will then be solved by network simplex code. The two simulation models are implemented using discrete-event simulation software, AutoMod, and the performances of both deployment schemes are analyzed. The simulation results show that the new deployment scheme will result in a higher throughput and lower ship *makespan* than the current deployment scheme.

I. INTRODUCTION

One of the world's leading port operators, PSA Corporation based in Singapore, is planning to automate its container transportation within the container terminal by implementing an Automated Guided Vehicle (AGV) System (AGVS) in its new, highly automated container terminal. Typical operational planning and control problems in such system are: dispatching of AGVs to containers in the terminal, routing of AGVs and controlling traffic in the network of lanes and junctions. In this project, we consider one aspect of the terminal operation, which is to dispatch AGVs to containers in the terminal.

Each container corresponding to a ship is a job and there are 2 types of job movements, discharging/unloading (movement from quay to yard) and loading (movement from yard to quay). We assume that we are given the *crane job sequence* for each quay crane serving the ship. The crane job sequence for each quay crane consists of the following information.

- A *sequence* of jobs that will be discharged from/loaded onto the ship,
- A set of potential storage locations in the yard area for each container to be discharged from the ship and is already determined.

The job's *pickup* and *drop off* location is denoted as the source location and destination location of the container to be loaded onto and unloaded from the vehicle.

II. PROBLEM STATEMENT

The AGV dispatching problem is to deploy AGVs to serve all the container jobs such that all the time constraints for all jobs are met. This makes sure that an AGV has to reach the *quay crane* before the time the container is to be *dropped off* or *picked up* by the *quay crane*. If this constraint is satisfied by the deployment scheme, the terminal is operated at a throughput rate that is pre-specified. However, the queuing of AGVs to queue at the *quayside* is undesirable as it creates congestion at the *quayside*, hence another objective of the deployment scheme is to reduce the waiting time of the AGVs at the *quayside* when they are waiting for the quay crane to *pickup* or *drop off* containers onto it. Although an AGV can carry either one or two containers at a time, only the performance of a unit capacity AGV will be studied in the simulation.

An efficient way to solve a unit capacity vehicle-dispatching problem is to first formulate the whole deployment problem as a network flow problem. A network simplex algorithm with an upper bound technique, which is a specialized revised simplex algorithm, can solve the problem efficiently by exploiting the structure of the network flow problem. The linear algebra of the simplex algorithm is replaced by simple network operations. Ahuja, Magnanti, and Orlin [2] describe the (primal) network simplex algorithm and gave pseudo-codes and implementation details. An implementation of the primal and dual network simplex algorithm is presented by Löbel in [3].

III. CURRENT DEPLOYMENT SCHEME, *pmds*

The general rule behind the deployment scheme is based on a greedy heuristic that aims to dispatch vehicles to jobs such that the time each vehicle spends waiting for the quay crane to serve a job is minimized. For each quay crane, the predetermined crane job sequence, consisting of n jobs may consist of only unloading jobs (or "*u*" job) or only loading jobs (or "*l*" job) or a combination of both unloading and loading jobs (or "*u*/*l*" job). In the latter case, the job sequence consists of two parts: the first part includes all the "*u*" jobs followed by all the "*l*" jobs. For each "*u*" ("*l*") job, there is a predetermined *drop-off* (*pickup*) point in the yard, which is the location of the job. All the jobs are arranged in the order of First In First Out (FIFO) basis based on the earliest appointed *pickup/drop off* time of the job at the *quayside*.

In the greedy heuristic, the first *k* jobs are assigned, each to a single AGV. The next job is assigned to the AGV such that the AGV will reach the location at a time that will minimize the AGV waiting time for the crane to unload/load the "*u*"/"*l*" job to the vehicle. Specifically, when assigning a "*u*" job, the AGV that has the closest arrival time at the *quayside* to the appointed *pickup* time of the job will be dispatched to this job. Similarly, when assigning a "*l*" job, the AGV that has the closest arrival time at the *quayside* to the appointed *drop off* time of the job will be dispatched to this job. Normally for "*l*" job, the AGV that can reach the job source at the *yardside* at the earliest time is dispatched since it needs to travel a longer distance from current position to the *yardside* and to the *quay*side compared to a "*u*" job. The deployment scheme focuses on deployment of one job at a time.

Design of the deployment scheme, pmds

Before actual deployment, the planning of the job deployment time is done such that the 1st 4 jobs per crane will be assigned an initial appointed *pickup/drop off* time. The fifth job per crane will be assigned when the service of the 1st job at the *quay* has actually been completed. The assignment of the sixth job will depend on the completion of the 2nd job and so on. The planning period is effectively 4 jobs ahead. For the 1st 4 jobs, the appointed *pick/drop off* time of the i^{th} job will be

Appointed time = ship discharge time + (i-1) * *time window*

For the next subsequent jobs after the 1st four, the appointed *pickup/drop off* time of the j^{th} job will be

Appointed time = (j-4)th actual *pickup/drop off* time + 4* *time window*

Thus, for each crane, there are only 4 jobs at each time that are assigned an appointed pickup/*drop off* time.

Before calculating the expected waiting time, the status of the AGVs must be determined. AGVs can be in the following 4 states, "*retrieving*", "*delivering*", "*going to park*" and "*idle or parked"* status. Only AGVs that are neither in the state of "*retrieving*" the next load nor being assigned to the next load are deployed in the current deployment scheme, pmds. The waiting time for job L_i, for the remaining 3 states of the AGV is calculated as follows:

If the job L_i is a "*u*" job,

1. If the AGV is "*delivering*" the current job, L_j,
   Distance traveled by AGV, Dist = distance(AGV current position, L_j destination) + distance(L_j destination, L_i source)
   Waiting time for L_i = L_i appointed *pickup* time – (Dist/Average Velocity + *crane* average operating rate for L_j)

2. If the AGV is "*going to park*", going towards its assigned park location,
   Distance traveled by AGV, Dist = distance(AGV current position, AGV destination) + distance(AGV destination, L_i source)
   Waiting time for L_i = L_i appointed *pickup* time – Dist/Average Velocity

3. If the AGV is "*idle*",
   Distance traveled by AGV, Dist = distance(AGV current position, L_i source)
   Waiting time for L_i = L_i appointed *pickup* time – Dist/Average Velocity

If the job L_i is a "*l*" job, the value of "Dist" will be increased by another distance(L_i source, L_i destination). Moreover, the waiting time will be decremented by another *yard crane* average operating rate for L_i.

The whole deployment scheme can be visualized in Figure 1.

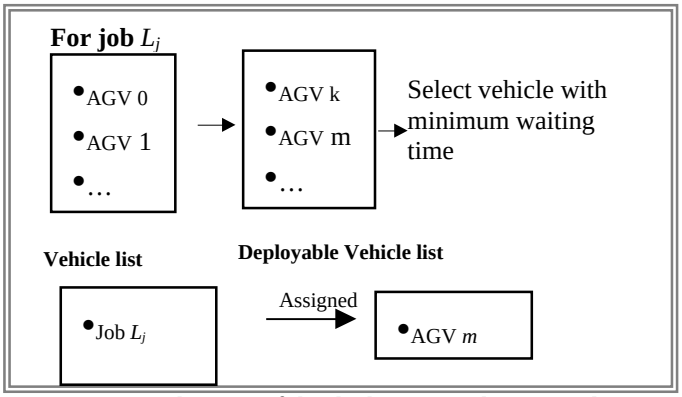


Figure 1:A visualization of the deployment scheme, pmds

The determination of the average velocity of each AGV is important since it is the only parameter that will affect the time information. The average velocity will be determined based on the historical statistical data information.

IV. PROPOSED DEPLOYMENT SCHEME, *mcf*

The proposed deployment scheme can be formulated as a minimum cost flow problem (*mcf*) and solved with the network simplex algorithm. The network simplex code is

written in 'C++' language and is easily available on the Web [3].

The containers to be served and the AGVs to be deployed can be formulated as nodes in the network. Assume in this problem, there are m number of AGVs and n number of containers or jobs. Altogether, there are a total of $m+2n+1$ nodes. The m AGV nodes can be regarded as source nodes and there is 1 sink node. Each container node will be split into 1 container node and 1 virtual container node (or container' node) giving 2n container nodes as the transshipment nodes. The reason for splitting each container node into two is to ensure a flow through the node, i.e. a vehicle must pickup the container. This will be explained in the following.

Given a network $G(N, A)$, where N is a set of nodes and A is a set of arcs. The following needs to be defined before forming the network.

- $N = \{N_{AGV} \cup N_{JOB} \cup N_{JOB'} \cup N_S\}$. The set of nodes, N is split into 4 mutually exclusive set of nodes where N_{AGV} is the set of m number of AGV nodes; N_{JOB} is the set of n number of container nodes; $N_{JOB'}$ is the set of n number of virtual container nodes and N_S is the set of one sink node. The labeling of the nodes in each set is given as follows:
  - $N_{AGV} = \{1,2, \ldots, m\}$
  - $N_{JOB} = \{m+1, m+2, \ldots, m+n\}$
  - $N_{JOB'} = \{m+n+1, m+n+2, \ldots, m+2n\}$. The $m+1$ node in N_{JOB} corresponds to $m+n+1$ node in $N_{JOB'}$, and $m+2$ node corresponds to $m+n+2$ node and so on.
  - $N_S = \{m+2n+1\}$
- $A = \{A_{JJ'} \cup A \setminus A_{JJ'}\}$. Similarly, the set of arcs, A is split into 2 mutually exclusive set of arcs where $A_{JJ'}$ is the set of arcs flowing between container node, N_{JOB} to its corresponding container' node, $N_{JOB'}$ and $A \setminus A_{JJ'}$ will simply be the set of the remaining arcs that excludes arcs in $A_{JJ'}$.
  - $A_{JJ'} = \{(i, j) \in A \mid i \in N_{JOB,k}, j \in N_{JOB',k}, k = 1,2,\ldots,n\}$. $N_{JOB,k}$ and $N_{JOB',k}$ refers to the k^{th} element in the set of N_{JOB} and $N_{JOB'}$ respectively.

Alternatively, the problem can be formulated mathematically as:

(N1) minimize $\sum_{(i,j)\in A} c_{ij} f_{ij}$

subject to

$$\sum_{j \in N_{JOB} \cup N_S} f_{ij} = 1, \qquad \forall i \in N_{AGV} \tag{1}$$

$$\sum_{j \in N_{AGV} \cup N_{JOB'}} f_{ji} = -m, \quad \forall i \in N_S \tag{2}$$

$$\sum_{(i,j)\in A \setminus A_{JJ'}} f_{ij} - \sum_{(j,i)\in A_{JJ'}} f_{ji} = 1, \qquad \forall i \in N_{JOB'} \tag{3}$$

$$\sum_{(i,j)\in A_{JJ'}} f_{ij} - \sum_{(j,i)\in A \setminus A_{JJ'}} f_{ji} = -1, \qquad \forall i \in N_{JOB} \tag{4}$$

$$f_{ij} = 0, \qquad (i, j) \in A_{JJ'} \tag{5}$$

$$0 \le f_{ij} \le 1, \qquad (i, j) \in A \setminus A_{JJ'} \tag{6}$$

In (N1) both equations (3) and (4) will ensure the container i be picked up by the AGV where each container i is a sink node and virtual container i' is the source node. Take note that the arc costs for arcs $(i, j) \in A_{JJ'}$ and for arcs $(i, j) \in A \setminus A_{JJ'}$, $\forall i \in N \setminus N_S$ and $\forall j \in N_S$ are equal to zero since container i and j belongs to the same container and for vehicles to end at final destination after pickup respectively.

An example for both formulations is illustrated below. Assume there are 2 AGVs to be deployed and 4 container jobs to be served. Thus,

- $N_{AGV} = \{1,2\}$
- $N_{JOB} = \{3,4,5,6\}$
- $N_{JOB'} = \{7,8,9,10\}$
- $N_S = \{11\}$
- $A_{JJ'} = \{(i,j) \in A \mid i \in N_{JOB,k}, j \in N_{JOB',k}, k = 1,4\}$

The shaded nodes are the AGV nodes. An example of the network based on the formulation in (N1) is illustrated in Figure 2. The numbers in brackets indicate the lower and upper capacity of the arc flows in $A_{JJ'}$.

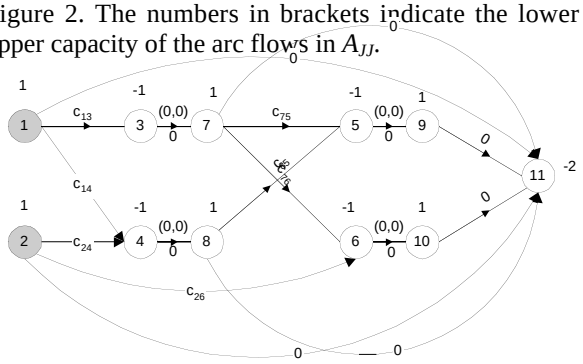


Figure 2:Example of a network of 2 AGVs and 4 containers using formulation (N1)

Based on the idea of the waiting time from the current deployment scheme, pmds, each arc cost can be thought of as the value of the time difference between the next job's appointed *pickup*/*drop off* time at the quayside to the current job's appointed time in the quayside plus some **travel time**; or the time difference between the next job appointed *pickup*/*drop off* time in the quayside to the current AGV ***ready time*** plus some **travel time**. To elaborate further, the calculation of the arc costs can be split into 2 sections: *arc cost between an AGV node and the next job/container i node* and *arc cost between job i' node and the job j node*.

Arc cost of arcs flowing from AGV to container node

As mentioned, the job's *pickup* location is denoted as the source of the container to be loaded onto the AGV and the job's *drop off* location is denoted as the destination of the container to be unloaded from the vehicle. In addition, the time at which the vehicle is ready to be deployed is denoted as the vehicle ***ready time***. Given that the job's source and destination are known and the AGVs' positions can be monitored at all times, the vehicle *ready time* can be calculated.

Each AGV can be in the following 4 states. Based on the status of the vehicle and the information given above, each vehicle's ready time can be calculated as follows:

1. If the AGV is in the state of "*retrieving*" the next job, L_j at current time,
   AGV expected total distance travel, Dist = distance(AGV current position, L_j source) + distance(L_j source, L_j destination)
   AGV ***ready time*** = current time + Dist/ Average velocity + Crane operating time rate at L_j source + Crane operating time rate at L_j destination

2. If the AGV is in the state of "*delivering*" the job, L_i it carries at current time,
   AGV expected total distance travel, Dist = distance(AGV current position, L_i destination)
   AGV ***ready time*** = current time + Dist/ Average velocity + Crane operating time rate at L_i destination

3. If the AGV is in the state of "*going to park*" at some location at current time,
   AGV expected total distance travel, Dist = distance(AGV current position, AGV destination)
   AGV ***ready time*** = current time + Dist/Average velocity

4. If the AGV is in the state of "*idleness*" or remain at its current position at current time,
   AGV ***ready time*** = current time

If the jobs L_j and L_i are "*u*" jobs, the crane-operating rate at the source is the *quay crane*-operating rate; otherwise the crane-operating rate at the destination is the *yard crane*-operating rate.

The arc cost from the AGV node to the container/job node is ready to be determined. There are 2 cases to this situation

1. If the next job, L_j is a "*u*" job,
   Dist = distance(AGV final destination, L_j source)
   Arc cost = L_j appointed *pickup* time at quay – (AGV *ready time* + Dist/Average Velocity)

2. If the next job, L_j is a "*l*" job,
   Dist = distance(AGV final destination, L_j source) + distance(L_j source, L_j destination)
   Arc cost = L_j appointed *drop off* time at quay – (AGV *ready time* + Dist/Average Velocity + *yard crane* average operating time rate for L_j).

   The AGV final destination will be the expected destination of the next or current job served if it is in the state of "*retrieving*" or "*delivering*" at the current time respectively; or the expected destination of its park location or current position if it is in the state of "*going to park*" or "*idle*" at the current time respectively.

Arc cost of arcs flowing from container i' to container j node

The calculation of the arc costs from one container to another container node is slightly than the former. Since a job can be either a "*u*" or "*l*" job, there are 4 cases to this situation.

1. If L_i is a "*u*" job and L_j is a "*l*" job,
   Dist = distance(L_i source, L_i destination) + distance(L_i destination, L_j source) + distance(L_j source, L_j destination)
   Arc cost = L_j appointed *drop off* time at quay – (L_i appointed *pickup* time + Dist/Average velocity + *yard crane* average operating rate for L_i and + *yard crane* operating rate for L_j)

2. If L_i is a "*u*" job and L_j is a "*u*" job,
   Dist = distance(L_i source, L_i destination) + distance(L_i destination, L_j source)
   Arc cost = L_j appointed *pickup* time at quay – (L_i appointed *pickup* time + Dist/Average velocity + *yard crane* average operating rate for L_i)

3. If L_i is a "*l*" job and L_j is a "*l*" job,
   Dist = distance(L_i destination, L_j source) + distance(L_j source, L_j destination)
   Arc cost = L_j appointed *drop off* time at quay – (L_i appointed *drop off* time + Dist/Average velocity + *yard crane* average operating rate for L_j)

4. If L_i is a "*l*" job and L_j is a "*u*" job,
   Dist = distance(L_i destination, L_j source)
   Arc cost = L_j appointed *drop off* time at quay – (L_i appointed *drop off* time + Dist/Average velocity)

The distance traveled or the 4 cases are illustrated in Figure 3. It is possible that there is a negative arc cost value. For negative arc cost value between the two nodes, it means that the AGV that flows through this node is late for the job appointed time in the quay or the AGV can serve job *i* but will be late for the next job *j*. Thus, a large positive penalty cost will be assigned for flow on this arc to allow some lateness for the AGV to pickup the job to safeguard against insufficient resources (AGVs).

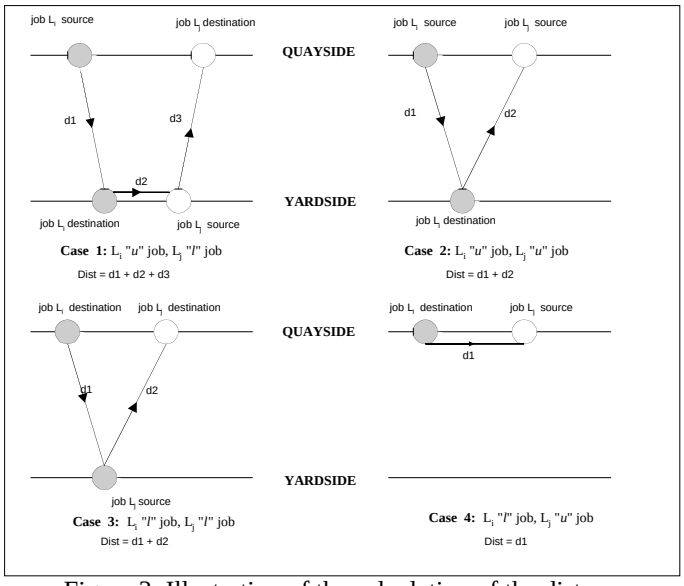


Figure 3: Illustration of the calculation of the distance traveled for the 4 cases

Design of the proposed deployment scheme model, mcf

All incoming and unassigned containers at different points in time will be inserted into a queue based on the job appointed time in the quayside on First-in First out basis (FIFO) basis. We denote this queue as *Q*1. All the jobs in *Q*1 and all AGVs will be formulated in the network based on formulation in (N1). The jobs that are assigned to vehicles generated by the network simplex solution will be taken out of the queue, *Q*1.

However, it is not practical to assign all the jobs in *Q*1 based on the given solution because of the uncertainty of the traffic conditions. Some late jobs could affect the rest of the later jobs exponentially. Moreover, the solution might not be optimal due to the change of the job status. Thus, there will be some cut-off time where the simulation needs to be halted temporarily and re-planning for the job appointed pickup/drop off time needs to be done. The change in the job status results in a semi-dynamicness of the deployment scheme.

Before actual deployment, all jobs will be assigned with an initial appointed *pickup/drop off* time. The time interval between jobs will be a constant time interval dictated by the length of *time window*. After re-planning, the rest of the later un-deployed jobs will be assigned a new appointed *pickup/drop off* time based on the same *time window* interval from the 1st un-deployed job. The new appointed time of the 1st un-deployed job will be determined by the actual service time of the last job at the *quayside* with a grace period of 4 minutes.

After **every** *k* number of jobs for **all** cranes have been deployed, the re-planning of the appointed *pickup/drop off* time and its mcf problem formulation will be done based on the existing jobs in Q1. For every simulation halt, any new incoming jobs will be appended after existing jobs in *Q*1. The number *k* depends on when the re-planning should be done based on historical traffic condition. Similar to the pmds model, *k* is selected to be 4.

Due to lesser resources (AGVs) compared to the number of jobs waiting to be serviced, it is possible that there is more than one job assigned to an AGV. In order to prevent AGVs serving more than one jobs at the same time, each AGV will be given a "to-do" queue list. Thus, there is equal number of "to-do" queues as the number of AGVs. Similarly, the jobs for each "to-do" queues are based on FIFO and the AGV will service the earliest assigned job. Only jobs serviced by their respective AGVs will be removed from the *Q*1 but the remaining jobs in each "to-do" queues will still remain in the *Q*1 either waiting to be serviced later or reformulated and solved by mcf code. For each simulation halt, the existing jobs in the "to-do" list will be emptied.

The design of the mcf model in AutoMod simulation model needs to interface with the MCF solver (network simplex solver) in order to solve the network and obtain the solution to deploy the vehicles to the jobs. Due to the restriction of the file handling process in Automod simulation, a C function needs to do the input/output file interface. This interface is shown in Figure 4.

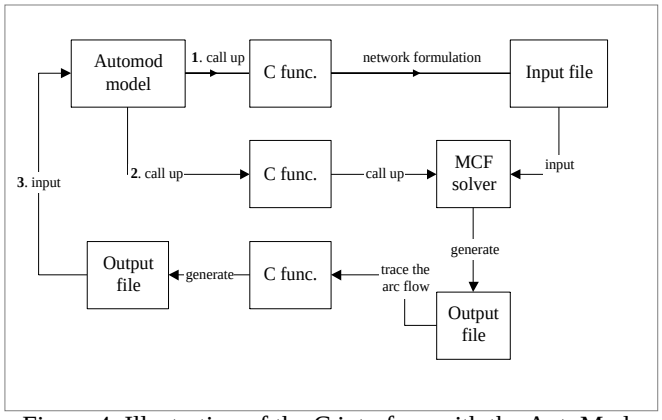


Figure 4: Illustration of the C interface with the AutoMod model for the *mcf* model

V. DEADLOCK PREDICTION & AVOIDANCE ALGORITHM

From Coffman, Elphick and Shoshani [4], the four conditions that must be satisfied before a deadlock is concluded are as follows:

- *Mutual exclusion*: 2 or more processes cannot use a resource at a time
- *No preemption*: When a resource is being used, it is not released until the process using it finishes with it.
- *Hold and wait*: A process that is holding at least one resource and is waiting to acquire additional resources that are currently being seized by other processes.

- *Circular wait*: A closed chain of processes in which each process is waiting for a resource occupied by next process in the chain

Thus, a deadlock will not occur if one of the conditions does not hold. For AGVS, the 1st three conditions are always true and only the last condition can be prevented. The resources mentioned in the four conditions refer to zones of the path and the processes refer to the AGVS. "Mutual exclusion" is obvious since two vehicles cannot occupy a zone at the same time. This is a condition required by the control system to prevent two vehicles from colliding with each other. "No preemption" is also obvious because, the vehicle must be in any zone at a given time and the movement of the vehicle into another zone satisfies the condition. As for the "Hold and Wait" condition, it is also satisfied in the case of the AGVS, as each vehicle has to be in a zone at any one time and is waiting to move into its next designated zone. The last condition, "Circular Wait" is not always true in AGVS and this is where the deadlock prediction can be used to detect whether a deadlock is imminent.

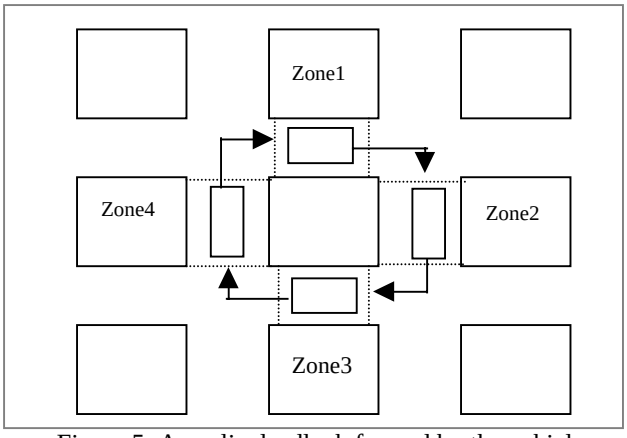


Figure 5: A cyclic deadlock formed by the vehicles

Proposed Deadlock Prediction Strategy

Due to the unavailability of fast methods of predicting deadlock in the literature, Wee and Moorthy[1] proposed a proposed one-zone step deadlock prediction algorithm as follows:

Its definition of the numbers in Figure 6 is listed as follows:

1. Extract the location (L_p) (or control points) of its next zone of the selected vehicle (say Vi) that is about to enter a new zone. For every sampling time in the control system, i.e. 1.5 sec to 2 sec, a check is done to see if a vehicle has moved to a new zone or not. If it has, the vehicle is selected so that a deadlock prediction for its next zone step is done
2. Check whether this next zone (L_p) is occupied by another vehicle
3. Extract the location (L_q) of Vi's next 2 zone (i.e. the "next next" zone)
4. Check whether any other vehicle occupies L_q
5. Extract next zone location (L_r) of the vehicle that is occupying L_q and update L_q to the location L_r
6. Return "vehicle is waiting for the block to clear"
7. Return "vehicle is safe to proceed, deadlock is not predicted"
8. Check whether L_p is equal to L_q
9. Return "vehicle is not safe to proceed, deadlock is predicted"

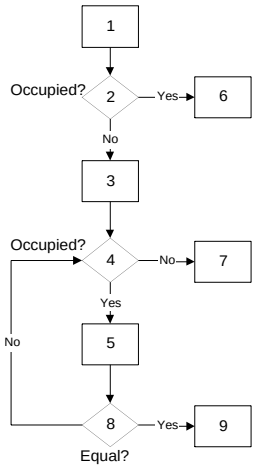


Figure 6: Flowchart of the one-zone step deadlock prediction algorithm

It is possible to extend this idea of one-zone step to the two-zone step prediction to facilitate a better performance by predicting the deadlock earlier. However, there is a disadvantage of this form of prediction in the implementation as mild approximations are done here. These kinds of approximations come into effect because the vehicles do not travel from one point to another in exactly the theoretical time required. This difference between the expected time and the actual time creates an error. This error of prediction gets larger as more zone steps are predicted in advance. Thus, it is sufficient to use one-zone step prediction and implement the necessary deadlock avoidance strategy.

Proposed Deadlock Avoidance Strategy

Normally, there are two ways to resolve deadlock: detection-resolution and prediction-avoidance. The prediction-avoidance strategy is used since this strategy will minimize the number of formation of deadlocks.

The two strategies used in the avoidance measure consist of:

- **Wait and Proceed**

As the name suggests, if a vehicle predicts a deadlock in its route, it will stop and wait at the same location until at least one vehicle is cleared from the deadlock prediction region

- **Rerouting**

Due to the time constraints, a semi-dynamic routing strategy instead of dynamic routing strategy is proposed. To facilitate deadlock prediction and avoidance, a set of static route for each AGV is ascertained before moving to the destination. The routes are stored in a table and when an AGV requests for its next location, the next zone or

location is returned for the particular requesting AGV. Thus, every AGV will follow its ascertained route from the stored information at one control point or location at a time.

The semi-dynamic rerouting strategy comes to play, and routes need to be recalculated when one of the conditions holds:

- The AGV reaches the destination and picks up a new job and is ready to move to the job's destination.
- The AGV reaches the destination and drops off a job and is ready to move to the next job's pickup location.
- The deadlock prediction algorithm predicts the formation of a deadlock in the next location or a new zone and requests for a new route for the AGV.

It is not necessary for the routing strategy to be dynamic since it is costly and moreover, the deadlock formation is not very frequent. The route is calculated using the Dijkstra's algorithm to find the shortest path tree (SPT). The general outline of the Dijkstra's algorithm and the implementation are explained in detail by Ahuja, Magnanti and Orlin [2] and Gallo [5]. The performance of the shortest path algorithms is dependent on the network and there are a lot of variants of the shortest path algorithms specific to the underlying network.

To further enhance the prevention-avoidance measure, a one-zone step deadlock resolution strategy is also implemented. It is still possible for deadlocks to form due to the uncertainty of traffic especially from the loading and unloading effects. Thus, deadlock resolution is implemented at the quay and yard side where loading and unloading occurs. The resolution algorithm will assign a virtual intermediate control point to the AGV when deadlock occurs at the quay or yard side. After completion of the loading/unloading process, this AGV is prepared to move on to its new destination. If it encounters a cyclic deadlock meaning there is a cyclic request of their next control points by other AGVS, this particular AGV will reroute to this virtual location that will break the cyclic request. Otherwise, this AGV will follow the pre-assigned shortest route.

VI. RESULTS & DISCUSSION

The layout of the four berths used in the simulation of the two models, pmds & mcf is shown in Figure 7. A scenario showing the AGVs waiting to pick up or drop off container jobs on the berth side and container storage area in the yard side can be shown in Figure 8

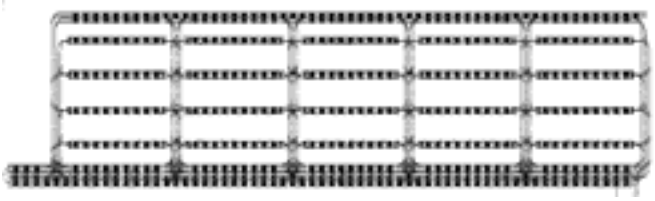


Figure 7: Layout of the four berths in the simulation model

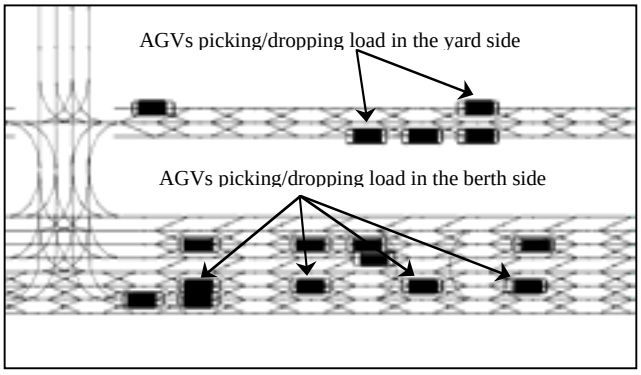


Figure 8:Scenarios showing AGVs picking/dropping jobs in the berth and yard areas

Specifications of the two models

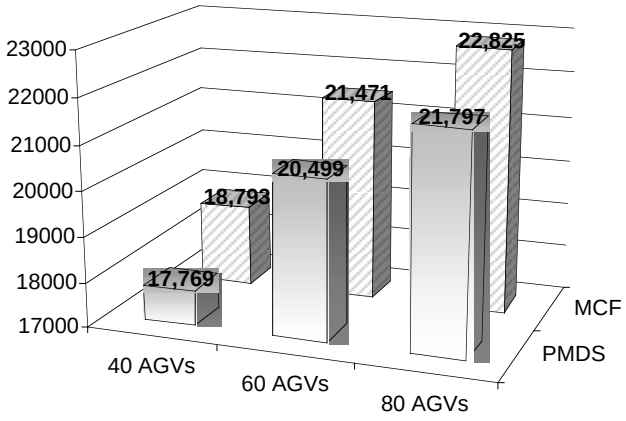
Both models use the same specifications and are as follows:

- Each vehicle is assumed to be able to ferry only a *unit load* from quay to yard side or vice versa at any point in time.
- The service time interval is deterministic and the re-assignment of the service time will be done mentioned previously.
- A berth is randomly assigned to an incoming ship. If the berths are full, the next incoming ship will wait in the queue until the previous ship has completed discharging/unloading.
- The arrival of the ship is assumed to follow an exponential distribution of mean 60 minutes. This means the arrival rate for each ship follows an exponential distribution where 63% of the time the arrival rate is less than 60 minutes and 37% of the time the arrival rate is more than 60 minutes.
- Each container storage yard is made up of 9 clusters.
- Each cluster is made up of 3 control points.
- At any one time, a single cluster can only be used by a quay crane for either discharging or loading process. It is possible to move the quay cranes but the movement is not simulated here.
- 4 quay cranes are assigned per vessel.
- The distribution of workload of each quay crane is as follows:
  - 1^{st} quay crane: 18%
  - 2^{nd} quay crane: 25%
  - 3^{rd} quay crane: 27%
  - 4^{th} quay crane: 30%
- The time taken for a quay crane to load and unload a container follows a triangular distribution of (1.375, 1.708, 2.113) minutes.
- The time taken for a bridge crane to load and unload a container follows 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 illustrated above.
- The average speed of the vehicles for each scenario is determined by the lower statistical mean of the historical data of previous runs of the two models.

The results are illustrated as follows:

Comparison of effects of the number of AGVs

In this simulation, the comparison of the effect of the number of AGVs on the throughput of the two models is made. The simulation was run with 40, 60, 80 AGVs over a period of 4 days. The inter-service time or time window is fixed at 2 minutes. The total number of boxes in the four berths in a period of 4 days for the two models is shown in Graph 1.

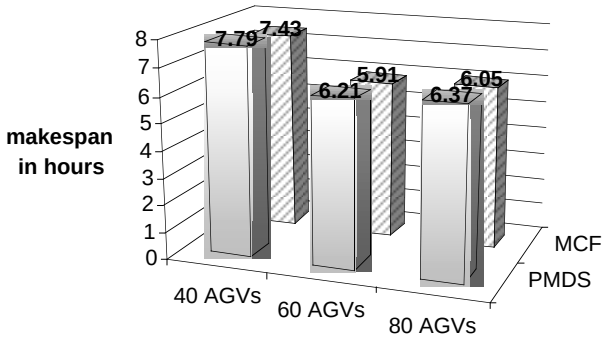


Graph 1: Total number of boxes serviced

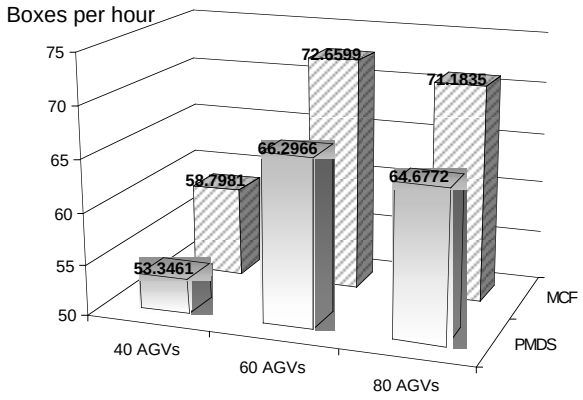
The *makespan* of the ship refers to the time a ship spends at the port performing loading and unloading operations. The throughput of a ship, measured in number of boxes per hour is defined as

Throughput = no. of boxes serviced per ship/ship *makespan* time

The ship mean *makespan* in hours for both models are shown in Graph 2. The average throughput in number of boxes per hour for both models is shown in Graph 3. The mean *makespan* and mean throughput values are calculated by taking the mean of all the ships' throughput for all four berths.



Graph 2: Ship average makespan

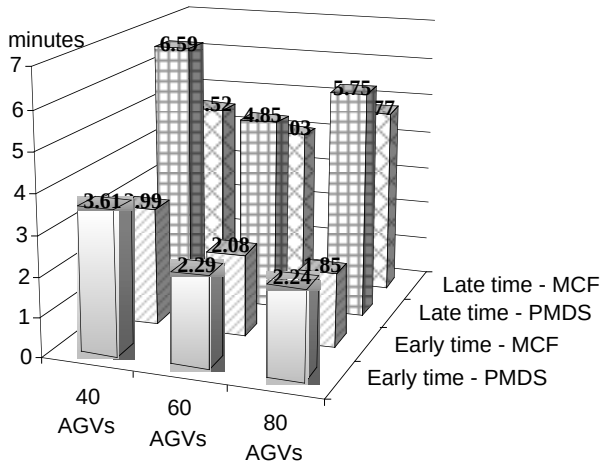


Graph 3: Average throughput

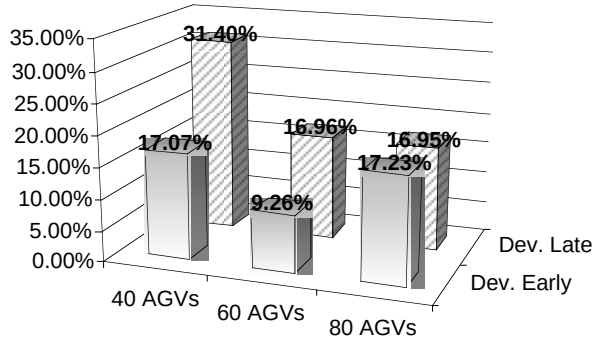
From the results above, the throughputs of the ships in the *mcf* model are 9-10% higher than that of the *pmds* model. In both models, there is a very slight 2 % drop in throughput for 80 AGVs compared to the case of 60 AGVs. The throughput seems to increase as the number of AGVs increase. Interestingly, the throughput seems to remain at a constant rate and drop as the number of AGVs increases to large numbers. This is most probably because congestion effects start to become significant when the number of AGVs increases to a large number while the space of the layout of the four berths remains the same. The result shows that the optimal number of AGVs per crane per berth is in the range of 4-5 AGVs.

Throughput is one of the means to measure the efficiency of port operations. However, other factors that could affect the cost of operation should not be neglected. Examples are the length of time the AGV has to spend waiting for the quay crane (i.e. AGV is early for the appointed service time) or the length of time the AGV is late for the appointed service time.

Theoretically, the selected AGV for each job is expected to arrive before the job service time. However, due to the uncertainty of the actual traffic conditions and comparatively lesser number of resources (AGVs) to jobs, most AGVs will not arrive exactly on time i.e. at the job's appointed service time. There will be some deviations from the appointed service time. The effects on the length of mean waiting time for the case where the AGV is early for the appointed service time and the length of mean late time for the case where AGV is later than the appointed service time are investigated. The number of AGVs and their mean time deviations in minutes for the two models are shown in Graph 4. The improvement of the time deviations of the *mcf* model compared to the *pmds* model is shown in Graph 5



Graph 4:Mean Time deviations from appointed service time

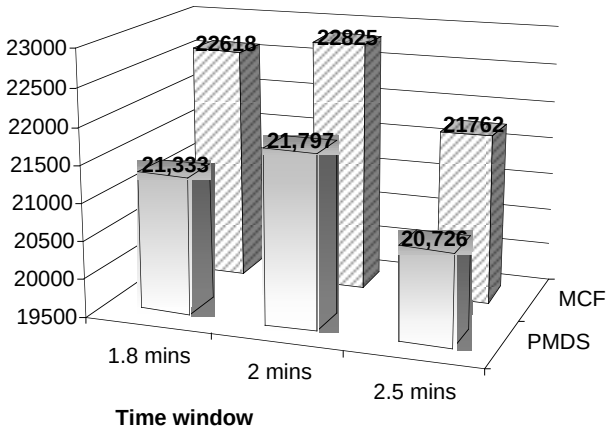


Graph 5:Improvement of mean time deviations of MCF model

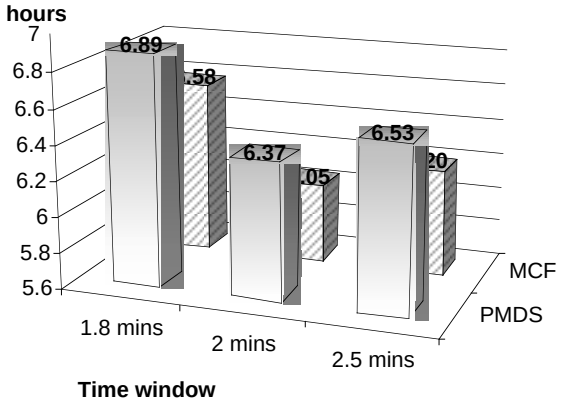
There is some improvement of the mean time deviation of the mcf model compared to pmds model. This is logical because the greedy heuristics do not make efficient use of the resources (i.e. AGVs) compared to the minimum cost flow model. For both models, the lengths of the mean waiting time and late time decrease as the number of AGVs increases as there are more resources catering to the jobs. The number of AGVs late for the jobs is much more than the number that are early for jobs because the inter-service time is too short to factor in the time-travel, the quay and yard operating rate and possible congestion effects. It is noted in the simulation that the AGVs are early for jobs initially but are late for jobs as more jobs are generated.

Effect of the variation of time window

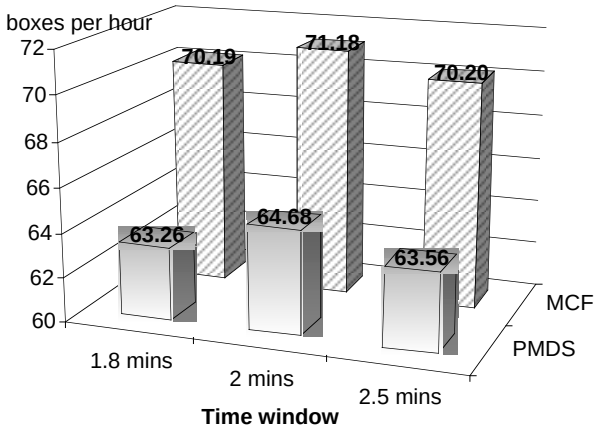
The simulation was run with 80 AGVs over a period of 4 days. The inter-service time or time window was varied from 1.8 to 2 and 2.5 minutes. Similarly, the total number of boxes, the ship mean *makespan* and the mean throughput in each berth for the two models are shown in Graph 6, 7 and 8.



Graph 6: Total number of boxes serviced



Graph 7: Average ship makespan



Graph 8: Mean throughput

The performance of the throughput for mcf model is still 9-10% better than that of the pmds model. For both models, the throughput will increase when the time window varies from 2.5 minutes to 2 minutes and decrease when the time window varies from 2 minutes to 1.8 minutes. Both models use the job-based approach, so jobs tend to be late. If the time window is too tight (e.g. 1.8 minutes), there might be

more AGVs that are late for the job. On the other hand, if the time window is more relaxed (e.g. 2.5 minutes), the number of AGVs that are late is less. The time window must factor in some allowance for traveling time and the crane-operating rate.

Discussions and Conclusions

Clearly, the performance in terms of throughput and time deviations for the *mcf* model seems to be better than that of the *pmds* model. The main reason for the inferior performance of the greedy approach used in the *pmds* model is its "myopic" nature. It assigns vehicles to jobs without considering other jobs that need to be completed in future. It only looks one job ahead for each crane at a time. The min cost flow algorithm in the *mcf* model on the other hand considers all jobs and tries to seek an optimum solution for assigning vehicles to job.

In practice, however, the greedy algorithm is an appealing solution procedure due to its simplicity and flexibility. The implementation of the algorithm does not take up too much of the computational process time and there is no file input/output involved. The minimum cost flow algorithm in the *mcf* model requires a lot of file input/output, which is time-consuming. This problem is worsened by the poor file input/output interface provided in AutoMod version 9.0. Hence, the process time of network formulation is heavily penalized. However, an improved file-handling interface will most likely be provided in later versions of AutoMod. The solving time, on the other hand is very fast. The number of nodes in the model we have solved is in the range of $O(10^2)$ and the number of arcs is in the range of $O(10^4)$.

We have also found from results presented in this paper that the recommended number of deployable AGVs for each crane appears in the range of four to five. The results also show that a time interval of two minutes is a suitable length of time to use for the time window between jobs.

For future research, it will be useful to investigate whether the routing of the vehicles could be further improved since efficiency of the port operations rests heavily on both deployment and routing strategies. In the *pmds* model, only four jobs are assigned appointed service times at any one time. This number of four was selected experimentally from the observation of traffic conditions during simulation. The number four may not be optimal and the optimal number of jobs to have appointed service times is an area for investigation. In general we want to maximize the number of jobs that are assigned, subjected to the constraint that uncertainty in traffic conditions may render assignments of large number of jobs at any one time impractical. The assignment of many jobs at one time may lead to exponential accumulation of late time for jobs when a vehicle is late for a job. Similarly for the *mcf* model, we want to find the optimal number of jobs ***k*** to dispatch each time the network model is solved.

.

VII. REFERENCES

1. H. G Wee, R. Moorthy (2000), Deadlock Prediction And Avoidance in an AGV System. SMA Thesis.

2. Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin (1993). Network Flows: Theory, Algorithms and Applications. *Prentice Hall*.

3. Andreas Löbel (2000). MCF-A network simplex implementation ver. 1.2.

URL:http://www.zib.de/Optimization/Software/Mcf/index.html

4. E.G Coffman, M.J Elphick, and A. Shoshani (1971). System deadlocks. *ACM Computing Surveys*, 3(2), pp. 67-78.

5. Goirgio Gallo, Stefano Pallottino (1988). Shortest Path Algorithms. *Annals of Operations Research. J.C. Baltzer Scientific Publishing Company.*

6. David Simchi-Levi (1999). Project Summary Report on AGV deployment Scheme. Northwestern University, waiting to be submitted in *IEEE Transactions*

6. Jerry Banks, John S. Carson. II, Barry L. Nelson, David M. Nicol (2001). Discrete-Event System Simulation, 3rd Edition, *Prentice Hall1*

7. AutoSimulations (1999). AutoMod V 9.0 Reference Manual Volume 1 & 2.