Cyclic deadlock prediction and avoidance for zone-controlled AGV system

Rajeeva Lochana Moorthy\textsuperscript{a}, Wee Hock-Guan\textsuperscript{a}, Ng Wing-Cheong\textsuperscript{b}, Teo Chung-Piaw\textsuperscript{c,*}

\textsuperscript{a} Singapore-MIT Alliance, Singapore
\textsuperscript{b} University of Hong Kong, China
\textsuperscript{c} Department of Decision Sciences, National University of Singapore, 15 Law Link, Singapore 117591, Singapore

Received 20 February 2001; accepted 28 August 2002

Abstract

Automated guided vehicles (AGVs) are now becoming popular in automated materials handling systems, flexible manufacturing systems and even containers handling applications at seaports. Its performance affects the efficiency of the entire system. Deadlock formation is a serious problem as it stalls the AGVS. The objective of this paper is to develop an efficient AGV deadlock prediction and avoidance algorithm for container port operations. Deadlock in a broad sense is a situation in which at least a part of the system stalls. There are a lot of situations in which the system may stall and most of these situations can be avoided through the control and navigation system. This paper discusses the development of an efficient strategy for predicting and avoiding the deadlocks in a large scale AGVS.

A variety of deadlock-detecting algorithms are available, but these methods work mainly for manufacturing systems where the network layout is simple and uses only a small number of AGVs. The AGVS under study has complex layout and involves close to 80 AGVs. The proposed solution is implemented using the Automod simulation software, and the performance of the technique is presented. Simulation shows that all the potential deadlock situations can be detected and avoided via this methodology. Also, the proposed strategy is computationally efficient and is able to provide the movement decision to the vehicles within the 1.5-s sampling time.

© 2002 Published by Elsevier Science B.V.

Keywords: Deadlock prediction; Automated guided vehicle; Cycle detection method; Container port

1. Introduction

Quickening pace of globalization has significantly increased the demand for logistics and transportation, and in particular, the demand for containerized marine shipping. Due to increased volume of worldwide container traffic, container terminals have become an important component of logistics networks. Container terminals serve as hubs for the transshipment of containers from ship to ship or to other modes of transportation, e.g., rail and trucks. As every manufacturer is competing with one another on shorter time to market, lower inventory costs and better customer service, the need for efficient terminals is more important.
than ever. An efficient container terminal is one that allows speedy transshipment of containers to and from the ships. Such a speedy operation is important to both the carrier, since it provides significant operations efficiencies, and to the port, which can handle a large number of ships per day. Unfortunately, many terminals are now working at, or close to, capacity and there is significant pressure from the political and business sectors to increase terminal throughput and, in particular, decrease ship turnaround time at the port. In most cases, this needs the development of sophisticated, highly automated container transportation systems, which will allow the efficient container movement within the terminal area.

The mega container terminal in Netherlands has been operating a fleet of Automated Guided Vehicles (AGVs) since 1993 and the world largest container terminal operator, based in Singapore, is planning to operate a fleet of 120–150 AGVs in its new, highly automated container terminal. The scale of AGV operations in mega container terminals is typically very large, with free ranging AGVs moving in a complicated network of lanes and junctions (see Fig. 1). The AGVs transport containers between quaysides and container storage areas. These bi-directional AGVs have an advanced navigation system that guides them through the complex network to transport containers from multiple origins to multiple destinations efficiently.

Typical operational planning and control problems in such system are: dispatching of AGVs to transportation jobs, routing of AGVs and controlling traffic in the network of lanes and junctions. Transportation jobs generated by handling operations of ships and trucks are grouped together and they will be assigned to one of the available AGVs through an AGV dispatching module. 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 operations safety, the complicated network of lanes and junctions is partitioned into a large number of zones with a restrictive vehicle movement rule. Within each zone, only one AGV is allowed to occupy it at any time; thus, any other AGVs wishing to use the zone have to wait outside for movement clearance. As the throughout 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

![Fig. 1. Part of the AGVS layout showing one berth.](image-url)
required to stop an AGV from its top speed through the use of normal controlled braking mechanism and the time required for stopping the AGV is less than 10 s.

Due to the dynamic nature of terminal operations, such as breakdown of AGV or container handling equipment, unexpected delays in container handling, etc., the planned route of an AGV may interfere that of another AGV and thus leading to 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 collision (Fig. 2). This is different when compared to routing systems in communication networks where such physical constraints are non-existent. Such cases are to be taken care of by the navigation system and there are a host of other conditions to be checked by a particular vehicle before it moves.

In some serious cases, a deadlock may be formed, which causes the AGVs involved in the deadlock grinding to a halt. Without some forms of intervention, each of the AGVs will be waiting forever for an instruction to proceed. Deadlocks in AGV systems are mainly due to the zone-partitioning strategy used for avoiding vehicle collision. For a particular group of AGVs, when the next movement zone in the planned route of each of these AGVs is blocked by another AGV in the group, a deadlock is formed (see Fig. 3).

Occurrences of deadlocks reduce the efficiency of the system. Hence, there is a need to monitor the movement of each AGV and to predict the possibility of formation of a deadlock. The frequency of predictions depends on the size of each zone, the characteristics of the AGV kinematics and the frequency of issuing new movement instructions to AGVs.

In a typical AGV system, each AGV is equipped with a lot of sensors to periodically capture essential information such as location, speed, vehicle heading, images of AGV vicinities, etc. After being processed by an on-board computer, the information is sent to a central AGV control system through the vehicle’s communication device. The period is usually very short (less than 2 s), so that the central control system has the accurate information and sufficient time to react to unexpected events like vehicle malfunction. After the central control system receives the information, it has to decide whether to issue a new movement instruction to each AGV to adjust its heading or speed. In making such a decision, an efficient algorithm is needed to predict the possibility of deadlock formation in each period.

In this study, we focus on the deadlock prediction problem in an AGV system used in the container terminal environment; thus, treating other aspects of the system as given inputs. The objective is to develop an efficient deadlock prediction algorithm and an avoidance strategy. Section 2 reviews the relevant work on deadlock prediction. A new prediction algorithm along with an efficient avoidance strategy is presented in Section 3. Section 4 reports details of the numerical results and the last section concludes the paper.

![Fig. 2. Physical constraints in AGVS leading to interference.](image)

![Fig. 3. Illustration of blocking phenomenon in zone controlled AGVS.](image)
2. Problem statement and literature review

They are several forms of deadlock that can be formed in an AGVS. Given the layout and the routes of the vehicles, there can be many forms of deadlocks depending on the control logic of the AGVS. In this section, the three most common kinds of deadlock will be discussed including the generic form of deadlock, i.e. cyclic deadlock in which, the vehicles form a cycle of request for the zones.

2.1. Cross lane deadlock

The first kind of deadlock would be when two of the vehicles wanted to switch lanes. An illustration of this is shown in Fig. 4.

For example, say AGV1 wants to go to the working lane and AGV2 wants to go to the traveling lane. A situation may occur such that AGV1 might not be able to turn to the working lane if AGV2 is too near to the intersection, i.e. AGV1 might collide into AGV2 when it makes the turn into the working lane. The navigation system on the vehicle will tell the AGV1 to stop. This will be the same for AGV2, which results in two vehicles waiting for each other to move, but nobody can move.

This kind of deadlock would be resolved by having a smart navigation system, i.e. decide where the vehicles should stop when they are waiting. Installing more sensors to tell the vehicle where to stop can do this.

2.2. Shop deadlock

Another kind of deadlock termed shop deadlock (Chang et al., 1997) can occur when the vehicles are dispatched carelessly in an overloaded shop as illustrated in Fig. 5. In the container yard, there is only finite amount of capacity in each storage area. If all the storage space is used up, a situation as shown in Fig. 5 may occur. It shows a shop deadlock whereby AGV1 wants to unload its container but the storage space has reached the limit and AGV2 is waiting to pick up the container resulting in both vehicles waiting. The crane is lifting a container and cannot be dispatched to lift the container from AGV1.

To prevent the shop deadlock, three things can be implemented concurrently. The first is routing. In Fig. 5 there is only one lane in which the vehicle...
can move. Hence additional lanes and cross over between the lanes can be provided as in Fig. 6.

These additional lanes together with a good method of distribution of the containers throughout the storage yard and Banker's algorithm as explained by Chang et al. (1997) could prevent the shop deadlock from happening. For this paper, prediction of this form of deadlock was not implemented as the terminal operator based in Singapore has its own distribution algorithm that assures that the space constraint will not be tight.

2.3. Cyclic deadlock

The last deadlock, which is the most generic, is the cyclic deadlock shown in Fig. 7. This form of deadlock occurs when there is a chain of vehicles requesting for the zones (resources) in such a way that these form a cyclic request of zones. This form of deadlock prediction was not available in the control systems of the vehicles. A prediction and avoidance algorithm is needed to tackle this situation.

2.4. Methods in predicting or detecting cyclic deadlock

In the literature available, Hyuenbo et al. (1995), Lee and Lin (1995), Viswanadham et al. (1990), Yeh and Yeh (1998) have discussed the methods of predicting and detecting cyclic deadlocks. The works by Coffman et al. (1971) and Gold (1978) contained arguably the major theoretical results for formation of deadlock in communication network. The methods mentioned in the literature in predicting the deadlock are either complicated or computationally expensive. For our case, the computation time to predict such a deadlock must be small as every sample time for the control system is in the range of 1.5–2 s. In the following, a discussion on why the methods mentioned in the literature are not feasible for implementation is provided.

Lee and Lin (1995) and Viswanadham et al. (1990) use the petri-net theory to predict and avoid cyclic deadlocks in the FMS and AGVS. The entire network must be represented in the form of a matrix. The AGVS layout in our implementation consists of 1370 nodes and several thousand arcs. The dimension of the matrix would be in the order
of a few million elements, which takes up huge memory space. To detect a cyclic deadlock, matrix vector operation needs to be done. For our implementation, the matrix is too large and the computation per iteration is \( O(nn) \) where ‘\( n \)’ is the number of nodes and ‘\( m \)’ is the number of arcs of the network. Therefore it is not feasible to implement this method for this problem.

Hyuenbo et al. (1995) use graph theory to detect impending deadlock. In order to do that, bounded circuits defined in Reveliotis have to be found. The number of bounded circuits for our network is very large since the network in too complex and hence this method is also dropped.

Yeh and Yeh (1998) discuss a very efficient deadlock prediction strategy by identifying cycles in dynamic directed Graph. Though the motivation of the current work stems from the proposed algorithm in Yeh and Yeh (1998), do not provide a robust avoidance strategy. In the discussion below, we show that the performance of the prediction algorithm depends on the Network structure and for complex AGVS networks, a good Avoidance strategy is vital for improved throughput.

2.5. Multi-cyclic deadlock

In the course of the simulation study, a modified form of cyclic deadlock was discovered. Multi-cycle deadlock is formed due to congestion and confluence of many cycles. This condition cannot be predicted by any cycle detection algorithms since each cycle by itself is not complete. More details on the specifics of the multi-cycle deadlock are presented in Section 3.

As far as the authors are aware, none of the methods described above have been implemented in practice, let alone one that operate in real container terminal environment. Further, there has been no known work in detecting *multi-cycle deadlock situations* which if let undetected would stall the entire AGV system.

3. Deadlock prediction algorithm

In view of the shortcomings of the existing method, we develop a simple and effective approach in this paper to satisfactorily resolve this problem. The main idea is to dynamically project the position of each vehicle after one zone step, and to detect cycle forming from the new positions. The main advantage offered by this approach is that the cycle detection algorithm depends on the number of AGVs in the system, which in this case, is much smaller than the number of zones in the terminal layout.

3.1. Phase 1: cyclic deadlock prediction

The flow chart depicted in Fig. 8 illustrates how phase 1 of the algorithm works:

*Step 1:* Extract the location \((L_p)\) (i.e. the control points) of its next zone of the selected vehicle \((V_i)\) that is about to enter a new zone. For every sampling time, i.e. 1.5–2 s, 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.

*Step 2:* Check whether this next zone \((L_p)\) is occupied by another vehicle. If it is occupied,
return “vehicle is blocked” (A), otherwise, go to Step 3.

Step 3: Extract the location ($L_q$) of $V_i$’s next 2 zones (i.e. the “next next” zone).

Step 4: Check whether any other vehicle occupies $L_q$. If it is not occupied, return “vehicle is safe to proceed, deadlock is not predicted” (B), otherwise, go to Step 5.

Step 5: Extract next zone location ($L_r$) of the vehicle that is occupying $L_q$ and update $L_q$ to the location $L_r$.

Step 6: Check whether $L_r$ is equal to $L_q$. If they are equal, return “vehicle is not safe to proceed, deadlock is predicted” (C), otherwise, go back to Step 4.

The example in Fig. 9 shows four vehicles AGV1–AGV4 and the shaded nodes are the locations of the vehicles. The arcs in each of the node are pointing to the next location node of the vehicle’s route for e.g. AGV4’s next location will be node 2. Following the given algorithm for the one zone step deadlock prediction, say AGV1 is about to enter a new zone, i.e. node 2. It checks whether node 2 is occupied (Here, the node is free). It then checks its ‘next next’ node, which is node 3. It finds that AGV2 is occupying the node and hence, AGV2’s next node is checked, which is node 4. AGV3 is found to be occupying node 4 and then it continues to check AGV3’s next node, i.e. node 5 and finds that AGV4 is occupying it. Finally it checks that AGV4’s next node, i.e. node 2 is the same node that AGV1 wants to enter to. If AGV1 is allowed to enter node 2, there will be a cyclic request of resources, which implies a cyclic deadlock. The algorithm will thus return a value saying that a deadlock is predicted at the next zone step. When a deadlock is predicted, the system can choose to reroute the AGVs involved or choose to wait until the deadlock is cleared. In this paper, we focused on the latter as the primary objective of the paper is to study the effectiveness of the deadlock prediction algorithm, leaving the route selection procedure as a black box procedure that the port container operator can change any time.

Let $V$ denote the number of vehicles in the entire AGV system. In the worst case, each cycle detection phase performs at most $O(V)$ operations (for example, in a huge cyclic chain involving most of the vehicles), hence the computational complexity of the algorithm is in the worst-case $O(V^2)$ ($O(V)$ for each vehicle).

In a control system, it will be difficult to detect when an AGV exactly enters new zone, i.e. just crosses the boundary line of the previous zone and the new zone. This is due to the sampling time, every 1.5-s data is sent to the central control system such that the detection of crossing a zone boundary exactly cannot be detected. In order to detect whether the vehicle has entered a new zone, a comparison of the previous sampled position and the newly sampled position of the vehicle is done. If there is a change in the zone, the vehicle has entered a new zone. Here, it is assumed that a vehicle cannot traverse an entire zone within 1.5 s.

It is possible to extend this idea to the two-zone step deadlock prediction. The purpose of this is basically to facilitate for a better performance of the system. If deadlock is predicted earlier, then a
better avoidance measure can be taken. For example, when a vehicle’s destination is not in the predicted deadlock region, then the vehicle can be rerouted in order to avoid congestion in the potential deadlock region. A disadvantage of this form of prediction will be that 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 taken creates an error. This error of prediction gets larger as more zone steps are predicted in advance.

3.2. Phase 2: multi-cycle deadlock resolution

The current study showed the existence of a modified form of cyclic deadlock situation. This situation is termed as multi-cycle deadlock. It is formed when multiple impending cycles compete for a specific resource. Fig. 10 illustrates an example situation.

Existing methods look only for impending cycles and clearly, any cycle detection algorithm will let AGV1, AGV2, and AGV3 wait forever.

In Fig. 10, we see that the AGV1 wants to enter the empty node. It’s ‘next next’ node the same as that of the ‘next next’ node of AGV3. Now, AGV1 runs the deadlock prediction algorithm and finds an impending deadlock. According to ‘wait and proceed’, AGV1 waits for the deadlock to clear. This is the same for the AGV2 and AGV3 and their respective ‘next next’ position is as shown. AGV2 and AGV3 will thus wait for the deadlock to clear before proceeding. Here, the cyclic deadlock is avoided by the ‘wait and proceed’ measure but it in turn creates another deadlock, which is not cyclic, but has many cycles that share a single empty resource.

For this kind of a deadlock to occur, the following must be true:

- There must be more than one “cycle”.
- There must be a common empty node shared by the cycles.
- The cycles must be unique, i.e. different start node and end node (excluding the empty node).
- The ‘next next’ position of the vehicle at the end of each cycle must be the start of another cycle.

Discovery of such a hybrid deadlock situation basically led to the development of the prediction strategy described above. The prediction mechanism looks forward in time and hence, there is more than one way for resolving the multi-cycle deadlock case.

Fig. 10. An illustration of the multi-cycle deadlock situation.
3.3. Phase 3: multi-cycle deadlock resolution

In the second phase of the algorithm, we resolve these deadlocks in the following way (see Fig. 11):

- Take one of the AGV at the end of each cycle involved in this deadlock. For example, AGV2.
- Check the forward stars of the common node (i.e., check all nodes adjacent to the common node).
- For each forward star, check whether AGV2 will enter a cyclic deadlock if it enters the common node.
- If no impending deadlock is detected, then reroute it through this forward star.
- Otherwise, check other forward stars.

Note that new AGV routes need to be determined whenever such situations arise. This method is not a complete resolution of the multiple cycle deadlocks, it fails when all the forward stars of the common node is an impending cyclic deadlock. Fortunately, for the container port AGV layout, the multi-cycle deadlock occurs mainly in the ship loading/unloading area where there are plenty of small cycles interlinking with one another. In this area, each node has plenty of forward stars (i.e. each node can go to a lot of other nodes). Thus, the probability of entering a situation of all the forward stars having an impending cyclic deadlock is slim, provided that the number of vehicles in the berth area is kept at a reasonable value.

3.4. Phase 4: alternative multi-cycle deadlock resolution strategy

Another way of avoiding deadlock is by using a dynamic routing system. With a routing strategy that determines the shortest path for an AGV from its current location to its destination, and a smart arc-weight updating policy, such a system is realizable. Since the movement decision of an AGV needs to be given in real time, it is advisable to use a semi-dynamic approach. The following procedure gives guidelines for such a system.

- The 'work' is assigned to a particular vehicle and the origin and destination information is passed to the routing system.
- The travel time on the lanes is updated based on the projected congestion (Leighton et al., 1988). Here, projected congestion means that the traffic information is projected in time and the congestion in future is calculated assuming that the AGV’s proceed according to expectation.
- A shortest path is calculated with the projected congestion effects.
- The vehicles move from the origin to the destination along the calculated route.

Fig. 11. Resolution of multi-cycle deadlock.
The routes need to be recalculated only when a deadlock is predicted in the vehicles path. The procedure is called semi-dynamic because the calculated routes are stored and the vehicles use the stored routes. The following are the important components of the routing:

3.5. Calculation and storage of routes

Gallo and Pallottino (1988) have discussed in detail the various algorithms used for finding the shortest path. They also show that the performance of the algorithms depend mainly on the structure of the underlying network. Based on the Network under study, appropriate algorithm can be chosen. Also, in the proposed semi-dynamic system, vehicle path needs to be re-calculated only when one of the following conditions holds:

- The AGV reaches the destination and a new job is assigned to that particular vehicle.
- The deadlock prediction algorithm predicts the formation of a deadlock and requests for a new route for the AGV.

3.6. Updating the arc weights

The arc weights in the system are dynamically computed so that the vehicles can use a path of least congestion and at the same time try to travel the minimum distance to reach the destination. It has been shown by Leighton et al. (1988) that a routing schedule in \( O(c + d) \) can be constructed for a given layout. Here, ‘c’ is the maximum congestion and ‘d’ is dilation, i.e. the length of the longest path. Hence, reduction in the values of ‘c’ and ‘d’ has a direct impact on the schedule length. Using this result, one can construct an arc weight updating policy that can model these effects. Also, this means that a vehicle route so calculated will take a least congested path and hence the formation of multi-cycle deadlock situation will be remote.

4. Implementation

The simulation of the deadlock prediction algorithm and the AGV operations are done using the AutoMod simulation software\(^1\). Before we described how the workload is being modeled, we first provide definitions of some important terms:

- **Yard cranes** are cranes that carry containers from the vehicles to the storage yard or vice versa.
- **Bridge cranes** are yard cranes that run on concrete platform.
- **Quay cranes** are cranes that carry containers from the vehicles to the ship or vice versa.
- A **discharging process** is the discharging of containers from the ship to the container storage yards.
- A **loading process** is the loading of containers from the container storage yard to the ship.

The workload of the terminal is generated in the following manner:
- A berth is randomly assigned to an arriving ship.
- A ship arrives at every \( \frac{1}{2} \) h interval.
- Each container storage yard is made 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 purposes. It is possible to move the quay cranes but this movement is not simulated here.
- Four quay cranes are assigned to each vessel.
- Percentage of container handled by each quay crane:
  - 1st quay crane: 18%, 2nd quay crane: 25%, 3rd quay crane: 27%, 4th quay crane: 30%.
- Percentage of discharge containers follows a triangular distribution of \((0,50,100)\).
- Amount of loading containers is the number of discharge containers.
- Each batch (number of containers) per cluster that is handled follows a triangular distribution of \((1, 12, 24)\).

\(^1\)AutoMod V 9.0 reference manual.
The time taken for a quay crane to load and unload a container follows a triangular distribution (1.25, 1.63, 2.00) min.

The time taken for a bridge crane to load and unload a container follows a triangular distribution (2.00, 2.50, 3.00) min.

We will not touch on the implementation of the one-zone step deadlock prediction and avoidance (wait and proceed) algorithm in details, but to briefly sketch the idea of the implementation by an example. Consider the situation as shown in Fig. 12. Suppose that AGV1 wants to move into its next control point and is currently stopping at its current control point. When AGV1 is about to move into its next control point, it calls the deadlock prediction algorithm. In this example, the deadlock prediction algorithm will return that a cyclic deadlock is predicted if AGV1 moves into its next control point. At this point in time, the leaving station procedure for AGV1 will repeatedly call the deadlock prediction algorithm in discrete time steps of 0.2 s (you can always change the time step) until no deadlock is predicted or no vehicle is blocking its next control point. Meanwhile, the rest of the vehicles are going about their assigned task.

The same thing applies even if AGV1 is approaching the current control point (the control point shown in Fig. 12). When it detects that a deadlock is predicted, the ‘decelerate ok’ function instructs the vehicle to stop. When the vehicle stops at the current control point as shown in Fig. 9, the next thing it looks for is whether a process is associated with it. In this example, there are no processes associated with this control point and the leaving station procedure is called, as the vehicle is about to leave its current control point. After this, it does the same thing as described previously.

4.1. Sleeping vehicles

A problem encountered during the implementation is that if no work is assigned to the vehicle after it has delivered the container either to the...
storage yard or ship, it will “sleep” or “park” at that location. This results in a form of “deadlock” where the vehicle will not move off until a new job is assigned to it. Thus if there is another vehicle which has its destination point where the “sleeping” vehicle is, it will have to wait until the “sleeping” vehicle is assigned a new job and move away from the destination point. This results in wastage of time. To resolve this situation, an algorithm is presented that performs a “bumping” action. This “bumping action” is done when a vehicle wants to move into the location where the “sleeping” vehicle is positioned. The “bumping” vehicle will signal the “sleeping” vehicle to wake up and either “sleep” elsewhere or find a job to do.

4.2. Virtual control points

So far we have only focused on deadlock prediction. Even though the prediction phase is theoretically fool proof, a deadlock may occur due to the uncertainty of the vehicle route during the loading and unloading process. Take for example, a vehicle carrying a container reaches its destination and the unloading process (crane lifting the container) begins. During this period, the next job of the vehicle is still unknown. As it was not known how long the unloading process would take, without a next job, there would be no start and destination points and thus no route for the vehicle. The simulation environment requires that all vehicles should have a starting and destination point for proper working of the algorithm.

In order to resolve this kind of situation, a virtual “next control point” will have to be created for the vehicle during its loading and unloading process. With this virtual point, it is sure that if the actual route of the vehicle’s next job was through this virtual point, a deadlock would not occur. If the actual route of the vehicle’s next job is not through this virtual point, a deadlock might occur. If a deadlock is detected, a new route will be scheduled that route the vehicle through the virtual control point.

We illustrate this phenomenon with Fig. 13. Let us assume that AGV1 is unloading its container at the present control point as shown. It does not have a new job and therefore has no associated route. A virtual control point is created for AGV1 at that position when other vehicles are running their deadlock prediction algorithm. Thus if AGV1 is to actually move to the virtual control point when it receive its new route, a deadlock will not occur. This is because every move has been expected when each vehicle runs its deadlock prediction function. If AGV1 does not move to the virtual control point and instead the next control point is the position of AGV2, then a

![Diagram](image-url)  

**Fig. 13.** An illustration of the problem encountered.
deadlock occurs if there is a cyclic request of their next control points shown in the above figure. The cyclic request occurs between AGV 1, 2, 3, and 4. Thus in this case, AGV1 detects a deadlock and it will then re-route through the virtual control point. This will ensure that no deadlock will occur.

5. Results and discussion

The one-zone step deadlock prediction and avoidance algorithm was implemented first in a small-scale model for testing and later scaled to the actual model. The actual model layout consists of four berths and 80 AGVs. The workload generated follows the simplified version of the load distribution mentioned previously. The effectiveness of the algorithm on the small-scale algorithm is discussed first followed by the actual model.

The testing of the algorithm was done in a path layout shown in Fig. 14 with 8 vehicles. The workload distribution is random. The avoidance measure used in the model was the ‘wait and proceed’ method. The model was simulated for 52 weeks and no incident of a cyclic deadlock occurred. If the deadlock prediction and avoidance algorithm was not imposed, the model quickly runs into deadlock in 20 min.

The layout for the actual model used in the simulation is shown in Fig. 15. There are 80 AGVs in the actual model. The workload distribution was simulated according to the information given.

Without imposing the prediction algorithm, the simulation runs into a cyclic deadlock situation in 40 min and a typical deadlock is as shown in Fig. 16.

The simulation model was tested with the prediction algorithm. An impending deadlock situation that has been successfully detected is as shown in Fig. 17. In the figure, the cycle is not complete and no more AGVs are allowed to enter the cycle until the vehicles in the cycle move away.

The algorithm has been tested for various cases and Table 1 shows the number of deadlocks predicted in a period of 1 week. The simulation was run with 20, 40 and 80 vehicles and the avoidance measure used is ‘wait and proceed’. In the table, we can observe the relation between the number of deadlocks formed due to the number of vehicles and also due to the traffic effects. Surprisingly, deadlocks form quickly even with a small number of AGVs. Also, it is to be noted that the number of ships served reduces as the AGVs are reduced. This shows that any attempt to prevent deadlock formation by reducing the number of vehicles will not be worthwhile. Hence to improve service rate, more vehicles have to be used.

Table 1 also shows the deadlock formation rate, which is the ratio of the number of deadlocks predicted to the number of ships served. This is a
Fig. 15. Layout of the actual model.

Fig. 16. Cyclic deadlock in the actual model.

Fig. 17. Impending deadlock detected by the algorithm.
measure of the deadlock formation due to the traffic effects. As the number of vehicles is increased, the network becomes more congested and more deadlocks tend to be formed. But it is to be noted that the deadlock formation rate when 80 vehicles are used is only slightly higher than when 40 vehicles are used. This indicates that most of the deadlocks formed consists only of smaller cycles.

Fig. 18 is a bar chart showing the number of deadlocks formed in a 7-day period. The result is obtained by running the simulation with 80 vehicles. We note that this result corroborates the earlier result that the deadlock formation rate is 1.9. When 80 vehicles are used, the number of ships served in a day is 10 and the average number of deadlocks predicted in a day is 19.

The routing used in the simulation is obtained form AutoMod. AutoMod has a static routing policy and does not take congestion effects into consideration. It can also be expected that when dynamic routing is used, the probability of the multi-cycle deadlock situation arising will be minimal as we restrict the congestion.

The simulation results indicate:

- For a complicated network, deadlocks are formed irrespective of the number of vehicles used. Deadlock will not be formed though if the number of vehicles is extremely small (say 2 or 3) but the throughput is affected when too few vehicles are used.
- The number of deadlocks formed varies in direct proportion with the number of vehicles (when reasonable number of vehicles is considered). But, the rate of deadlock formation does not increase when more vehicles are added because most of the deadlocks are formed as small cycles.
- To increase throughput, more vehicles are needed but more vehicles means excessive congestion.
- Deadlock formation depends on the amount of work and distribution of the workload. More vehicles imply more ships being served and hence more frequent deadlocks.

6. Managerial insights and conclusions

Recent advances in information technology have been reshaping the modus operandi of material handling process; especially in key components of logistics supply chains like container terminals. In more advanced mega container terminals, AGVs is replacing traditional heavy-duty trucks to transport containers within terminals. An AGV system offers great flexibility to efficiently handle dynamic transportation requests arising from frequent container movements. Such a system is capital intensive and needs high productivity to make it economically viable. As with most fully automated systems, an AGV system is prone to deadlocks which, if not properly handled, can substantially reduce the productivity of the system. An effective and efficient deadlock prediction algorithm enables the system to carry out frequent predictions and to take necessary
precautionary measures to avoid deadlock formation. Numerical results presented earlier show that deadlock can be formed quite frequently in an AGV system of a mega container terminal. There is a need to carry out frequent predictions so as to avoid deadlock formation. However, all the deadlock prediction algorithms developed so far are computationally intensive and thus only applicable to small-scale AGV operations. This paper has developed an efficient deadlock prediction algorithm of complexity of $O(V^2)$, where $V$ is the number of vehicles in an AGV system. This efficient algorithm makes it possible to run the prediction in each short decision cycle of issuing new AGV movement commands. These frequent predictions are vital to prevent deadlock formation.

It is observed from the numerical results that the frequency of possible deadlock formation increases with the workload of the terminal, especially when workload distribution is highly uneven. Proper workload balance across different sections of the terminal can reduce the frequency. It is recommended that the impact of workload distribution on the likelihood of deadlock formation should be investigated.

References


Spyros, A.R., Conflict Resolution in AGV Systems. School of Industrial & Systems Engineering, Georgia Institute of Technology.
