Journal of the Korean Institute
of Industrial Engineers
Vol.23, N0.4, December, 1997

A methodology for the flexible AGV routing*

김지표**

Jipyo Kim**

Abstract

In the AGV routing a vehicle can select the shortest route on which no conflicts are anticipated. The procedure includes time constraint and the node/arc occupation times of vehicles in order to locate the conflicting paths. The slight differences among AGV routing algorithms exist in identifying the conflict regions encountered while reaching out to the destination node. In this paper, a novel approach to the use of the re-routing scheme is presented. It will be used to minimize the travel distance of vehicles in a regular vehicle routing process rather than to cope with emergency situations. The proposed algorithm provides existing active vehicles with the ability of changing their current paths for a new vehicle whenever the equal-distance paths exist, in an attempt to optimize the AGV transportation system. This ability is possible because of the flexibility of an AGV system controlled by a computer system.

1. Introduction

As a flexible computer-controlled factory-wide transporter, an automated guided vehicle(AGV) has become a popular material handling tool. Because its inherent flexibility fits so well with current trends in manufacturing, it is likely to become an increasingly appropriate choice. However, the use of AGVs raises several issues before the full implementation of an AGV system and one of them is a routing problem.

Once the destination of a vehicle is determined, the vehicle will move to the designated point as fast as possible according to the predetermined path. If there is only one vehicle serving in the system, the problem of finding a path to the destination is very simple and it will be the shortest one. However, if more than one vehicle are engaged in the system, a conflict in taking the shortest route may occur among the vehicles and subsequently, it yields the vehicle routing problem, i.e., the problem of how to select the specific paths for a fleet of AGVs in a non-conflicting manner.

One solution for the AGV routing problem is to allow a vehicle to take the shortest route according to the rule of first-come first-served. The vehicle can select the shortest route with the path on which no conflicts are anticipated. In general, the procedure involves a scheduling process which considers time constraints and the node/arc occupation times of vehicles in order to locate

* 이 논문은 서울산업대학교 교내 학술연구비에 의하여 연구되었음.

** Dept. of Industrial Engineering, Seoul National Polytechnic University

the conflicting paths. Basically, all algorithms developed so far employ this approach to route and schedule a new vehicle. The slight differences among algorithms exist in identifying the conflict regions encountered while reaching out to the destination node.

The assumption the AGV routing methods have in common is that the path of a vehicle should be respected once the vehicle has started its journey with a non-conflicting route and timetable. In other words, it is assumed that the vehicles perform the routes correctly. However, in reality the need of re-routing a vehicle may occur due to unforeseen events such as vehicle break-downs or urgent transportation. In such cases the vehicle generally has to be re-routed from the current location to its destination using the same algorithm.

In this paper, a novel approach to the use of the re-routing scheme is proposed. It will be used to minimize the travel distance of vehicles in a regular vehicle routing process rather than to cope with emergency situations. The proposed algorithm provides busy vehicles with the ability of changing their current paths for a new vehicle whenever the equal-distance paths exist, in an attempt to optimize the AGV transportation system. This ability is possible because of the flexibility of an AGV system controlled by a computer system. Also, the interactive routing techniques can be easily incorporated into existing AGV routing algorithms.

2. An AGV Routing System

The vehicle routing problem in an AGV system involves the design of a set of minimum distance routes from one location to another for a fleet of vehicles which serve a set of pickup/delivery stations. The difficulty in dealing with the AGV routing problem is that all the vehicles moving on a factory floor should be taken into account when planning a route for a new vehicle. This issue has to be dealt with under the added complexity of time windows, stemming from the fact that some arcs and nodes of a flow path network impose time intervals during which only certain vehicles can pass them. Therefore, the spatial aspect of routing is mixed with the temporal aspect of scheduling which must be carried out to ensure the satisfaction of the time window constraints.

In the AGV routing problem time windows are used to prevent vehicle collisions, particularly head-on collision, on a floor. Because the head-on collision takes place when two vehicles are traveling along the same path in opposite direction, the bi-directional flow path is naturally assumed in that case. Figure 1 illustrates the general scheme used to route AGVs.

The conceptual foundations of the AGV routing problem were first laid by Walker et al.(1985). They proposed an AGV scheduler that uses Dijkestra's shortest path algorithm and that generates a timetable which contains the node occupation times for each vehicle. The possible conflicts between a new vehicle and existing active ones are detected by comparing the node occupation times.

Taghaboni and Tanchoco(1988) presented a vehicle controller for free-ranging AGVs. Its task includes determining a route for the selected vehicle in a non-conflicting manner. In order to avoid the possible head-on collision, the vehicle controller uses the concept of virtual tunnels which exist only imaginarily between two nodes on a floor. The number of tunnels and their flow direction could be changed at any time so as to eliminate head-to-head collisions. The drawback of this method is in the assumption that each aisle between two nodes should have enough space to accommodate the tunnels required to avoid the possible conflicts.

Huang et al.(1989) modeled a labeling algorithm which employs multiple time windows on arcs and at nodes of a network. Assuming that no other vehicle is allowed to use the arc on which a vehicle is moving, a new network is generated by converting the arcs in the original network to nodes. Also, the travel time on an arc of the original network is replaced by dwell time on the corresponding

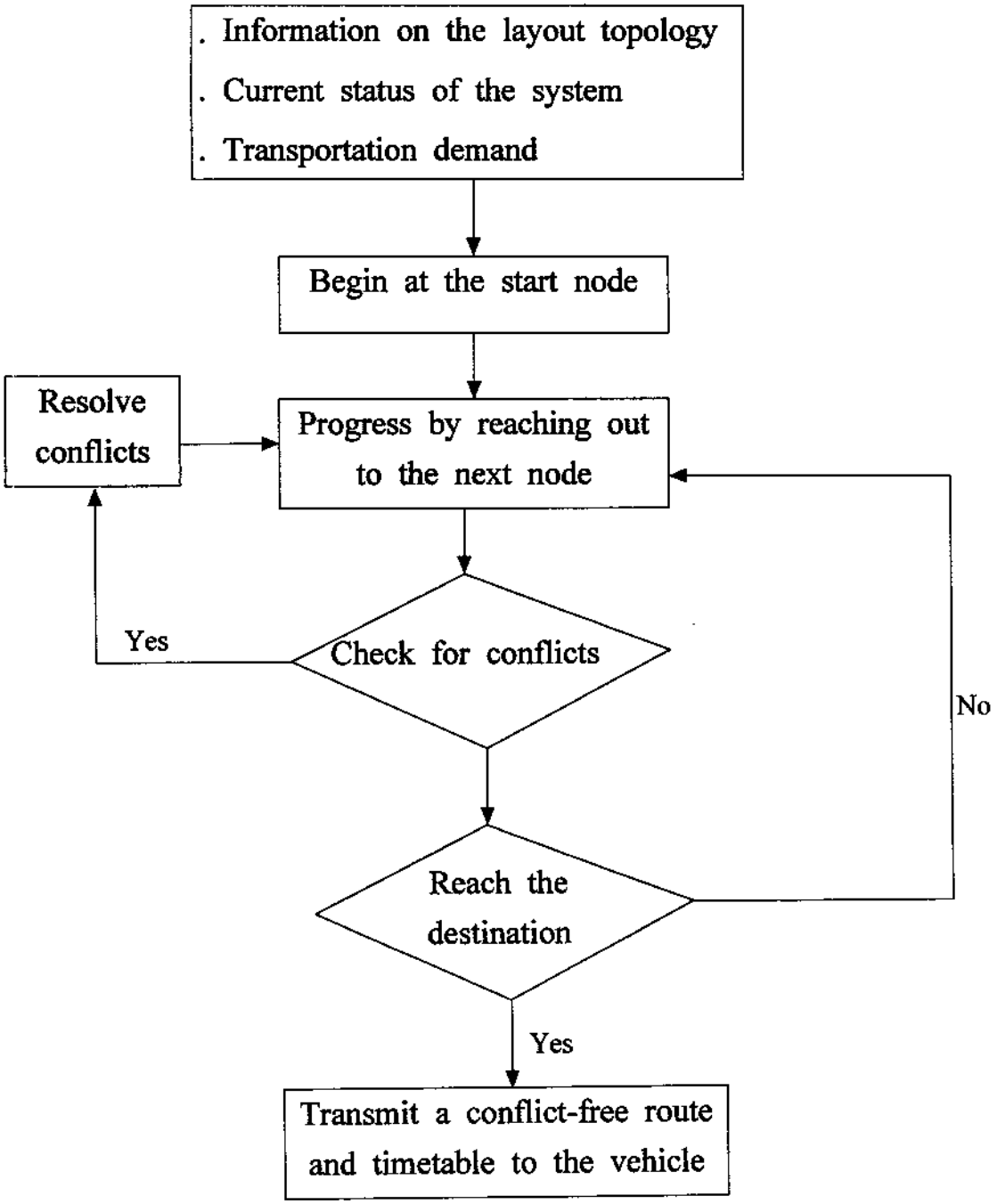

Figure 1. General scheme for routing an AGV

node of the new network. Therefore, the travel time on an arc of the new network is zero and the arcs just indicate the reachability between two nodes. When a new route is constructed, the time windows are imposed on nodes according to the node occupation times. Hence, the head-on collision on a track can be avoided by checking the time windows of nodes.

Kim and Tanchoco(1991) proposed an algorithm in which loops are allowed so that a vehicle can wait at the other available lane until the conflicting situation expires and restart its journey. They used the concept of a time window graph that is a directed graph of the free time windows which are distinguished from the reserved time windows. Dijkestra's shortest path method is used to determine a route in the time window graph. Potential conflicts in the lane (i,j) can be detected by comparing the vehicle crossing schedules at its end nodes i and j. Because a loop is allowed and a lane can be occupied simultaneously by more than one vehicle, it is claimed that their algorithm generates a better solution than the other ones.

3. Flexible Routing for an AGV

One of the main advantages of using AGVs is their flexibility. The route, number of vehicles, stops, and the quantity of components per vehicle can be varied. For instance, the routing of vehicles can be easily changed through software modification in a central computer without physical interruption. In this section, a simple but efficient algorithm to add more flexibility to the AGV system through re-routing is presented.

Re-routing implies that a set path is revised due to certain significant changes in operation requirements. The changes may be the events such as vehicle breakdowns, the occurrence of urgent journeys, and the sudden appearance of obstacles on a path, which can not be taken into account in the normal routing process. Thus far, re-routing has been discussed to resolve those problems.

Another possible use of the re-routing method is to include the re-routing module on a regular AGV routing algorithm in an attempt to produce a shorter path for a new AGV whenever possible. In the existing vehicle routing algorithms, the scheduled path for a busy vehicle is assumed to be fixed so that the new vehicle should wait or bypass conflict regions if conflicts between the existing tours and a new tour are anticipated. However, sometimes, the travel distance of a new vehicle could be reduced without significant change of delivery times of them if the busy vehicles are allowed to change their routes for the new vehicle and can find another path equal to the original one in terms of travel distance/time.

A typical situation is exemplified in Figure 2 which depicts the AGV floor system as a network G(N,A) where N represents a set of nodes and A a set of arcs. In the network the decision points for an AGV system are generally represented as nodes, and the tracks are represented as arcs. In this example, the nodes are numbered in a square. The number beside an arc implies the travel distance/time on the arc; here, the travel speed of a vehicle is assumed constant. The position of the nodes can be identified using their Cartesian coordinates in reference to the origin. Each node has a value for the x-coordinate and for the y-coordinate. Currently, two vehicles V_1 and V_2 are performing journeys according to the following paths:

V_1 : A→ [2]→ [3]→ [7]→ [11]→ [15]→ [16]

V_2 : B→ [11]→ [10]→ [9]→ [8]

(Note: A and B are the current positions of V_1 and V_2, and [16] and [8] are their destination nodes, respectively)

Now, the vehicle V_3 is picked to transport a unit load from the pickup point, node 14 to the delivery point, node 4. It is obvious that the shortest path from the pickup point to the delivery point is the following route: [14]→ [15]→[11]→[7]→[3]→[4]. However, if the route of V_1 and V_3 and the travel time of each vehicle are carefully

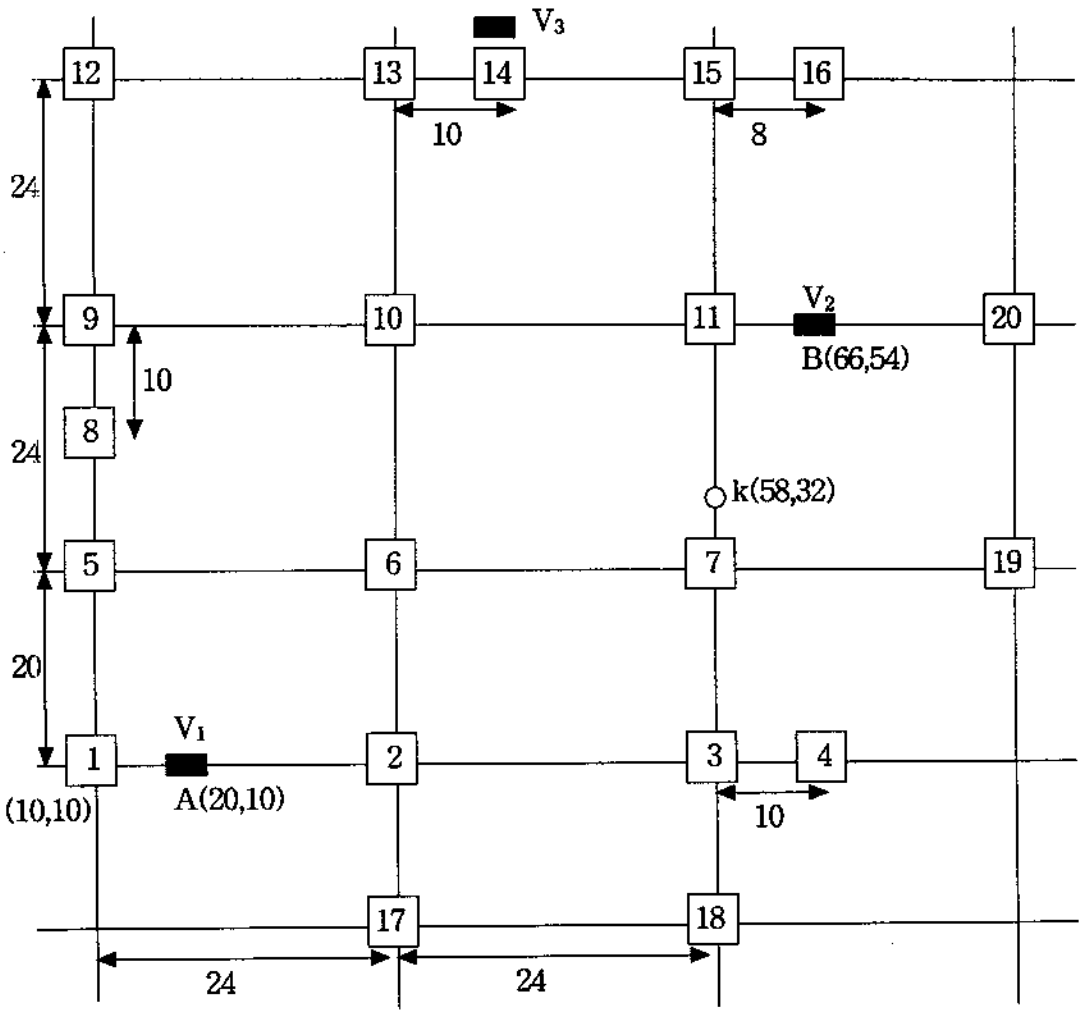

Figure 2. The layout of an AGV floor system

reviewed, it is not difficult to see a potential conflict somewhere on the arc (7-11).

Let (10,10) be the (x,y) coordinate of the node 1. Then the coordinate of the other nodes can be determined referring to the travel distance/time between two adjacent nodes. Let the current clock time be t=20. If the vehicles should move according to the paths described earlier, their paths and schedules are constructed as follows:

S(V_1) : A→ [2,(34,10)/34]→ [3,(58,10)/58]→ [7,(58,30)]/78] → [11,(58,54)/102]→ [15,(58,78)/126]→ [16,(66,78)/134],

S(V_2) : B→ [11,(58,54)/28]→ [10,(34,54)/52]→ [9,(10,54)/76]→ [8,(10,44)/86],

S(V_3) : [14,(44,78)/20]→ [15,(58,78)/34]→ [11,(58,54)/58] → [7,(58,30)/82]→ [3,(58,10)/102]→ [4,(68,10)/112],

where [a,(x,y)/b] denotes [node name,(x,y) coordinate/

node passing time].

By inspection, the time and the location of collision between V_1 and V_3 can be easily identified. They will collide head to head at t=80 and at the point k whose (x,y) coordinate is (58,32). Therefore, the route and schedule of V_3, $S(V_3)$, should be revised to avoid the conflict. A possible way for that is either to bypass the arc (11-7) or to have the vehicle V_3 wait at the available neighboring arc until the arc (11-7) is cleared if the waiting option is allowed. If the vehicle V_3 waited at the arc (11-10) or (11-20) until V_1 passes through the arc (7-11), the waiting time would be 44 unit times. It would take extra 20 travel unit times to reach the destination node if V_3 bypassed and moved according to the new path {[14]→ [13]→ [10]→ [6]→ [2]→ [3]→ [4]}.

Consider the possibility of re-routing the busy vehicle V_1. If the re-routing of the vehicle V_1 is allowed and the alternative paths with the same travel time as the original one can be found, the extra travel time will be saved. Considering the total travel time 92 of V_3 when no conflict occurs, the saving of 44 or 20 travel time is substantial. In the example the alternative routes of the V_1 could be $S^*(V_1)$ = {A→ [2]→ [6]→ [10]→ [11]→ [15]→ [16]} or $S^*(V_1)$ = {A→ [2]→ [6]→ [10]→ [13]→ [14]→ [15] → [16]}, and all of which render the same travel time as $S(V_1)$ and are free of conflicts.

One assumption regarding a facility layout configuration for this approach is that a set of aisles is arranged in a rectangular pattern parallel to the wall of a building or departments so that AGVs move along a straight line around departments/workstations. This assumption is realistic and reasonable particularly for an AGV system because an AGV is most efficient while moving along a straight line[5].

Another assumption used in this approach is that the total travel time originally set for each existing journey should be honored when alternative paths are sought. In other words, the alternative path candidates are limited to the paths in which a vehicle can carry components to a delivery point according to the first scheduled time because an arbitrary modification of the tour of busy vehicles, particularly when it results in the delivery delays, may cause a serious scheduling problem to the production process of departments. Therefore, the objective of the proposed re-routing method is to discover alternative paths of busy vehicles to minimize the travel time of a new vehicle while respecting the previously scheduled delivery time of busy vehicles.

3.1 Re-routing Method

Consider a situation in which a number of vehicles are currently serving the demands of departments on a factory floor. A new transportation demand is received, a vehicle, say V_m, is selected, and a routing algorithm is invoked. The algorithm finds the shortest path by progressively reaching out from the start node, which is generally done by iteratively picking out a base node and scanning all nodes connected to the base node in a network. In the process of reaching out from one node to another, the reachability should be checked. If a head-on conflict is detected, the re-routing algorithm is invoked to see the possibility of re-routing the busy vehicle, say V_o, which would be in conflict with V_m.

Reviewing the path of the vehicle V_o, first, the procedure identifies the start node s and the destination node t for the re-routing. In this case, the start node is the first node that V_o will reach after the re-routing is considered. Then the re-routing space and the path search direction are defined using the start node s and the destination node t. The procedure will explore a subset of alternative paths only in the re-routing space using the path search direction. For instance, if (x_s,y_s) and (x_t,y_t) are the coordinates of the node s and t, respectively, then the re-routing space S will be defined as $S = \{(x,y) | a \le x \le b$ and $c \le y \le d\}$ where $a=\min(x_s,x_t)$, $b=\max(x_s,x_t)$, $c=\min(y_s,y_t)$, $d=\max(y_s,y_t)$.

The path search direction consists of the following two

strategies in order to determine the direction of vehicles at each decision point:

Local direction strategy: The strategy, used for branch selection, encourages a vehicle to continue to move on in the current direction. For instance, if a vehicle just arrives at the node n, (p_1,q_1) from the node m, (p_2,q_2), then the vector $(\triangle p,\triangle q)$ will be the direction of next movement of the vehicle where $\triangle p= p_1-p_2$ and $\triangle q= q_1-q_2$.

Global direction strategy: The strategy, used for pruning branches, defines the possible directions for a vehicle. Let (x_s,y_s) and (x_t,y_t) be the coordinates of the start node s and the destination node t. Let $\triangle x= x_t-x_s$ and $\triangle y= y_t-y_s$. Then the vectors, $(\triangle x,0)$ and $(0,\triangle y)$, are defined as the global direction of movement of a vehicle. The direction of the vehicle should be either of the two vectors.

A tree search procedure based on the branch and bound method with depth-first search and backtracking is used to find the alternative path because it arrives very quickly at the optimal solution without jumping one tree to another. In the re-routing process any path which reaches the destination node first can be the optimal path that becomes equivalent to the original path of V_n in terms of travel time. Also, it is not necessary to compute the lower and upper bound at each branching step because the branching process proceeds only through the reachability checking.

Building on the observation made above, the formal description of the re-routing algorithm is presented below.

Re-routing Algorithm

The following notations and definitions are used in the description of the algorithm.

s : the start node of the vehicle to be re-routed,

t : the destination node of the vehicle to be re-routed,

S : the re-routing space,

G(N, A) : the network which represents an AGV floor system; N denotes a set of nodes and A a set of arcs,

G*(N, A) : the network created by the re-routing space S; then $G^* \leq G$,

B : the set of nodes which are generated by branching process; initially B={s},

C : the set of arcs which are pruned,

e : the node selected for the next branching process using the local direction strategy; initially e = s,

P : the array which records a parent node of the node selected for the branching.

(Initialization)

The procedure is initiated by determining G* and identifying the start node s, its immediate past node s*, and the destination node t.

Compute the global directions, $(\triangle x,0)$ and $(0,\triangle y)$, using nodes s and t.

Compute the local direction using nodes s and s*.

(Branching)

The branching process is started using the arcs which are directly connected to node e and follow the global direction strategy. It is a binary branching because the global direction strategy allows only two direction each time.

Update the set B.

(Pruning)

A branch i pruned if

i) head-on collision is expected on the arc which connects the terminal node and its parent node.

ii) the terminal node is out of the re-routing space.

iii) the terminal node has no child nodes to branch out.

The set C is updated by the result of the pruning step.

(Branch Selection)

Apply the local direction strategy to the new branches and determine new branching direction. Update node e and local direction. If all branches are pruned, the backtracking procedure is invoked; otherwise, go to the branching stage.

(Backtracking)

The backtracking step is invoked whenever all the branches are pruned so that the further branching process is blocked. If the sibling branch which is not pruned and has not been chosen before is available, then the procedure continues through the branch. Otherwise, the backtracking is conducted again.

(Termination)

The algorithm terminates when the destination node t is added to the set B or when the backtracking reaches the start node s and its child nodes are no longer available. The former case indicates that the alternative path for the vehicle is identified. The path can be constructed tracing back the array P. The latter case implies that the algorithm fails to obtain an alternative path.

3.2 Numerical Example

A numerical example is developed to illustrate the algorithm using the situation of V_1 in Figure 2.

(Initialization)

Start node : s, (34,10)

Immediate past node of s : s*, (10,10)

Destination node : t, (66,78)

$S=\{(x,y)|\ 34\leq x\leq 66 \text{ and } 10\leq y\leq 78\}$

Then $\triangle x = 66-34 = 32$

$\triangle y = 78-10 = 68$

Therefore, global directions are the vectors, $(\triangle x,0) = (32,0)$ and $(\triangle y,0) = (0,68)$.

Compute the local direction vector using the node 1, (10,10), and node 2, (34,10).

$\triangle p = 34 - 10 = 24$

$\triangle q = 10 - 10 = 0$

Therefore, the local direction vector, $(\triangle p,\triangle q)$, is (24,0), which indicates that the vehicle should, if possible, go to the direction that can increase the value of x coordinate.

$B = \{2\}$ and $C = \emptyset$

(Branching)

The two nodes 3 and 6 which are directly connected to the start node and follow the global direction are chosen. As shown in Figure 2, the corresponding branches are (2-3) and (2-6).

$B=\{3,6\}$

(Branch selection)

Consider the two nodes 3, (58,10) and 6, (34,30).

Compute the direction of each node;

for node 6, $\triangle p' = 34 - 34 = 0$

$\triangle q' = 30 - 10 = 20 \rightarrow (\triangle p',\triangle q') = (0,20)$.

for node 3, $\triangle p'' = 58 - 34 = 24$

$\triangle q'' = 10 - 10 = 0 \rightarrow (\triangle p'',\triangle q'') = (24,0)$.

Compare the direction of each node to the local direction obtained at the initialization step. Node 3 has the same direction as the local direction.

$e = 3$ and $(\triangle p,\triangle q) = (24,0)$.

(Branching)

The branching process generates the nodes 4 and 7 and the corresponding arcs are (3-4) and (3-7). $B = \{4,7\}$

(Pruning)

Arc (3-4) is pruned because node 4 is out of the re-routing space.

$C = \{(3-4)\}$

(Branch selection)

Node 7 is the only one left for branching.

$e=7$ and $(\triangle p,\triangle q) = (0,20)$

(Branching)

Nodes 11,19 are generated and their corresponding arcs are (7-11) and (7-19).

$B=\{11,19\}$

(Pruning)

Arc (7-11) is pruned because it is the conflicting arc.

Arc (7-19) is pruned because node 19 is out of the re-routing space.

$C=\{(7-11), (7-19)\}$

(Branch selection)

Because all the new branches are pruned, the backtracking procedure is invoked.

(Backtracking)

Because arc (2-6) which is the sibling of arc (2-3) is available, the branching procedure will start with the arc.

By continuing the procedure, Table 1 is obtained, which summarizes the results of each step. Figure 3 illustrates the search tree for this example. By retracing the array P, the alternative path for the vehicle V_1 can be constructed with the node passing time(node: *time*) as follows: {(2:*34*)→ (6:*54*)→ (10:*78*)→ (13:*102*)→ (14:*112*) → (15:*126*)→ (16:*134*)}.

Table 1. Results of re-routing V_1

Iteration	B	C	e	P
0	{2}	ϕ	2	-
1	{3,6}	ϕ	3	P(3)=2
2	{4,7}	{3,4}	7	P(7)=3
3	{11,19}	{7-11,7-19}	6	P(6)=2
4	{7,10}	ϕ	10	P(10)=6
5	{13,11}	ϕ	13	P(13)=10
6	{14}	ϕ	14	P(14)=13
7	{15}	ϕ	15	P(15)=14
8	{16}	ϕ	16	P(16)=15

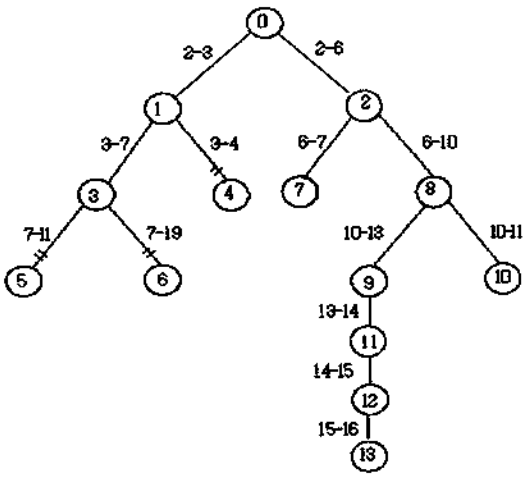

Figure 3. Search tree

3.3 Discussion

The algorithm developed is intended to reinforce the existing AGV routing algorithms by granting vehicles the capability of re-routing. It can, incorporated into existing regular vehicle routing algorithms, help to minimize the travel time of vehicles.

Because the requirement of computation of a lower and upper bound for each branch is eliminated, the algorithm can reach an optimal solution very quickly. Any path which reaches the destination node first from the newly defined start node can be the solution; i.e., a feasible solution obtained first becomes the optimal solution in this algorithm. Therefore, the reachability from one node to another becomes the major concern and the reachability test for conflicts on aisles takes the most computational time of the algorithm. However, the additional computational time by implementing the re-routing algorithm does not affect the performance of the whole system significantly. For example, Kim and Tanchoco (1991) reported that the complexity of their AGV routing algorithm is $O(v^4n^2)$ where v is the number of vehicles and n the number of nodes in the layout. In this case, the computation time due to the reachability test is $O(v^2)$. Therefore, the complexity of the algorithm is still $O(v^4n^2)$ when the re-routing algorithm is used with Kim and Tanchoco's routing algorithm.

The effectiveness of the re-routing algorithm depends much on the type of a department layout. The algorithm particularly works well on layouts where aisles are orthogonal to each other and intersections have two aisles crossed each other because there are more chances to find alternative paths. The algorithm, however, may have difficulty identifying alternative paths for vehicles whose path involves a loop or detour because it is almost impossible to find the alternative path equivalent to the original one with a loop or detour in a given layout.

4. Conclusion

In this paper, we discussed an AGV re-routing technique that is able to check the feasibility of re-routing a vehicle and renew a schedule whenever possible. The re-routing approach is defined as a procedure to revise an initial path of existing active vehicles whenever it triggers conflicts with a vehicle to be newly introduced. In the literature the re-routing method has been mentioned as a tool to cope with the unforeseen events such as vehicle breakdown and lane blockage. In this research the re-routing method is used, as a part of a regular vehicle routing system, to utilize the flexibility of an AGV system. The feasibility of this approach was also discussed and it was demonstrated in an example that the travel distance of vehicles can be reduced considerably through the re-routing process. Because the re-routing process takes place in a computer software, a vehicle can continue to move without any interruption even though its original path is modified.

References

[1] Egbelu, P.J., and Tanchoco, J.M.A., "Potential for bi-directional guide path for automated guided vehicle based systems," *International Journal of Production Research*, Vol. 24, pp.1075-1098, 1986.

[2] Huang, J., Palekar, U.S., and Kapoor, S.G., "A labelling algorithm for the navigation of automated guided vehicles," *Advances in Manufacturing Systems Engineering*, edited by M. Anjanappa and D.K. Anand, pp.181-193, 1989.

[3] Kim, C.W. and Tanchoco, J.M.A., "Conflict-free shortest time bidirectional AGV routeing," *International Journal of Production Research*, Vol. 29, pp. 2377-2391, 1991.

[4] Kim, J. and Klein, C.M., "Location of departmental pickup and delivery points for an AGV system," *International Journal of Production Research*, Vol. 34, pp.407-420, 1996.

[5] Muller, T., *Automated Guided Vehicles*(Bedford, UK: IFS Ltd.), 1983.

[6] Taghaboni, F. and Tanchoco, J.M.A., "A LISP-based controller for free-ranging automated guided vehicle systems," *International Journal of Production Research*, Vol. 26, pp.173-188, 1988.

[7] Walker, S.P., Premi, S.K., Besant, C.B., and Broadbent, A.J., "The Imperial College free-ranging AGV (ICAVG) and scheduling system," *Proceedings of the 3rd International conference on AGVSs*, edited by S. E. Anderson(IFS Ltd and North-Holland), 1985.

96년 4월 최초 접수, 97년 10월 최종 수정