SEQUENCE BASED HIERARCHICAL CONFLICT-FREE ROUTING STRATEGY OF BI-DIRECTIONAL AUTOMATED GUIDED VEHICLES

Samia MAZA and Pierre CASTAGNA,

Institut de Recherche en Communications et Cybernétique de Nantes,
1, rue de La Noë, BP 92101
Nantes Cedex 3, France

Abstract: Conflict-free routing is an important problem to be addressed in AGV Systems, in particular when they are bi-directional. This paper considers the problem of routing AGVs in the presence of contingencies. The importance of this work stems from the realization that path-planning approaches are ill suited to AGVSs subject to contingencies. In this paper, a two stage closed-loop control strategy is proposed. In the first control stage, a pre-planning method proposed by Kim and Tanchoco (1991) is used to establish the fastest conflict-free routes for AGVs. The objective of the second stage is the avoidance of deadlocks in the presence of interruptions while maintaining the planned routes.

Keywords: Automated guided vehicles, Deadlock, Robust control, traffic control, Sequences.

1. INTRODUCTION

In manufacturing systems, material transport plays a key role for production process efficiency. Because of their advantages over other material handling systems (conveyors, robots,…), AGVs are widely used in flexible manufacturing systems. The scheduling of several AGVs in a non-conflicting manner is a complicated real-time problem, especially when the AGV system is bi-directional. In fact, many conflict situations may arise, such as head-on and catching-up conflicts when the AGVs or the guide-paths are bi-directional and if no efficient control policy is used to prevent them.

Several conflict-free routing strategies have been proposed and can be classified into two categories:
(I) Predictive methods: which aim to find an optimal path for AGVs. The conflicts are predicted off-line, and an AGV's route is planned to avoid collisions and deadlocks (see Krishnamurthy, et al., 1993, Kim and Tanchoco, 1991, Oboth, et al., 1999).
(II) Reactive methods: AGVs are not planned and the decisions are taken in a real-time manner according to the system state. These methods are based on a zone dividing of the guide-path and consider them as non sharable resources (see Reveliotis, 2000, Pia Fanti, 2002, Moorthy, 2003). Predictive methods give good performances, but are not very robust since they do not take into account real time problems. However, reactive methods are very robust but the resulting performance can be poor because the decisions taken consider a very short term time horizon.

The objective of this paper is to present a mixed control strategy based on a path planning method. The objective of our control architecture is the avoidance of conflicts and deadlocks in the presence of disturbances, when the vehicles are previously scheduled.

This paper is organised as follows. Section 2 provides a brief description of the path planning method used. Section 3 is devoted to the description of the proposed method for AGVs re-scheduling. A simulation study is presented in section 4, where three deadlock avoidance algorithms are compared. Finally, this study will be concluded in section 5.

2. DESCRIPTION OF THE SHORTEST TIME PATH PLANING PROCEDURE

Kim and Tanchoco (1991) have developed an efficient algorithm called the *conflict-free shortest time procedure* (*cfstp*), to find the optimal conflict-free route for an AGV which moves on any bi-directional network, by considering the displacements of the other AGVs. The intersections and the workstations are modelled by square areas of size at least equal to that of the vehicle, called nodes. They are considered as non-sharable resources to avoid conflicts on them. A table modelling the current traffic status, i.e., scheduled entry and exit times of vehicles at each node, is then set up (fig. 1).
During a trip, each AGV will exclusively reserve some nodes on its path during an interval of time called reserved time windows. The free time windows

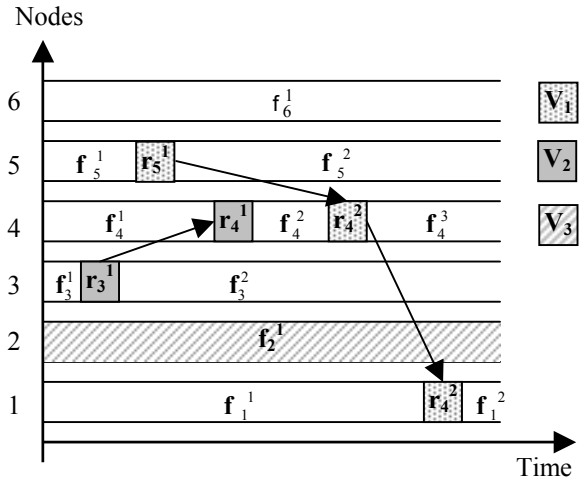


Fig. 1. Example of time-windows table
— r_i^j is the j^{th} reserved time window of node i
— f_l^k is the k^{th} free time window of node l.

between the reserved ones are available for scheduling other vehicle crossings.

The *cfstp* is used to find the shortest path on a directed time windows graph, in which the vertices represent the free time windows and the links model the reachability between these time windows. The ability to reach a time window from another one is established by calling another algorithm called *the reachability test procedure*. For two free time windows f_n^p and f_m^q associated respectively to the nodes n and m, this last procedure makes the following reachability tests between them:

(1) Check for space feasibility, i.e., the existence of a physical link relating m to n.
(2) Check for time feasibility, i.e., the node m is reachable from the node n within its free time window f_m^q.
(3) Check for potential conflicts.

3. REAL-TIME RESCHEDULING METHOD FOR CONFLICTS RESOLUTION

There are two types of contingencies: temporary and permanent. This work deals with the first type. This consists of a slowing down in front of a fixed or a moving obstacle, or a temporary stop on a lane or a node to charge the battery, etc. In that case, the scheduled arrival and exit times will not be respected and consequently, there is no security guarantee for the AGVs since collisions can occur.

3.1 Principle of the method

To ensure the reliability of an AGV system in the presence of interruptions while maintaining the scheduled trajectories, a control architecture was proposed in (Maza and Castagna, 2001, 2002). Indeed, a second level of real-time control was added to the AGVs scheduling level which uses the *cfstp*, in order to avoid deadlocks and conflicts when needed (Fig. 2). First, the AGVs are scheduled on the nodes and lanes of the guide-path in a non conflicting manner while optimising the mission's duration. Then, the scheduled entry times to each node are used to establish for each vehicle of the fleet, its own priority to cross these nodes (see Maza and Castagna, 2001). Two algorithms were proposed. The first one, called *RVWA* (or *Robust Vehicle Waiting Algorithm*), is based on a theorem that says that if each AGV respects its node crossing order, the property of non-conflict is conserved.
The second algorithm, called *RVRAA* (or *Robust Vehicle Routing Ahead Algorithm*) allows the rescheduling of the AGVs on some nodes in order to improve the *RVWA* which always induces unnecessary waiting of vehicles to respect their crossing priorities.

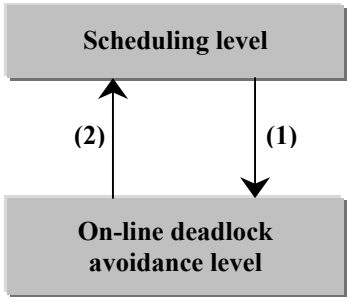


Fig. 2. The AGV control architecture.
(1) the scheduling level delivers for each node i, an ordered list, O_i, of AGVs having to cross it in a growing order of their arrival dates.
(2) the deadlock avoidance level operates in presence of contingencies by respecting the predicted node's crossing order (*RVWA*) or by re-ordering the AGVs (*RVRAA*). It informs the 1^{st} level about the current changes.

3.2 The robust vehicle delaying method

In order to improve the *RVRAA* which is based on a restrictive and simplifying hypothesis, another procedure for AGVs' re-ordering is presented here.

3.2.1 Description of the method

Instead of giving a vehicle, which is trying to cross some node, a highest priority on some predefined path, a new algorithm called *RVDA* (or *Robust Vehicle Delaying Algorithm*) is proposed. It is applied when an AGV V tries to cross any node N where it does not have priority. This algorithm first identifies the late AGV, say U, which has more priority than the actual AGV V on N. Then, it will try to penalize the vehicle U, by giving it a lower priority, on a path that will be calculated. This action can take place if and only if it will not induce an unsafe state.

Definition 1. an AGV state is called *unsafe* if it can conduct the AGV system to a deadlock state, i.e., it satisfies the necessary condition for the occurrence of conflicts.

In all what follows, the AGV which calls the *RVDA* and the late one will be called, respectively, V and U.

The figure 3 shows an example of unsafe state which is an imminent deadlock if the rescheduling action of U and V takes place.

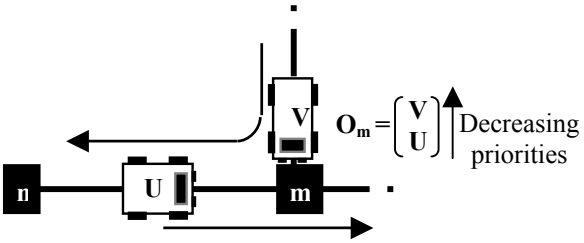


Fig. 3. Example of an unsafe state that may result if the AGV U is delayed in front of V on the node m (rescheduling action by the *RVDA*)

3.2.2 Safe AGVs rescheduling

Theorem 1. The first necessary condition for the re-ordering (delay action of the late AGV U in front of V) to be possible without inducing conflicts is that the late AGV is outside its common-path with V (see definition 2).

Before proving this theorem, let's first introduce some useful definitions.

Definition 2. If U is the late AGV, we define the common-path between the vehicles U and V as the set of common nodes to be visited by them, where U is more priority than V.

Definition 3. The guide-path is defined by a graph G=(N, E), where N is the set of nodes and E the set of edges relating each pair of nodes.

Definition 4. A sequence or a word is a finite character string defined on an alphabet Σ. If $v=v^1v^2\ldots v^n$ is a sequence of characters $v^i \in \Sigma$, then the integer n is the length of v denoted by |v|. The i^{th} element of a sequence s will be denoted s^i. The empty sequence does not contain any character and is denoted by 1, such that |1|=0.

Definition 5. Let u and x be two sequences defined on the alphabet Σ. The sequence u is a continuous sub-sequence of x, if there exist two sequences v and w on Σ such that: x=vuw.
The sequence u is called a *factor* of the sequence x. For example the sequence $u_1=x^2x^3x^4$ is a factor of $x=x^1x^2x^3x^4x^5x^6$, while $u_2=x^2x^5x^6$ is a sub-sequence of x but not a factor. u is called a *proper factor* of x if u≠x, and it is called a *left factor* of x if there is a sequence w on Σ such that x=uw. We write $u \in LF(x)$, where LF(x) is the set of the left factors of x. w is then a right factor of x. u is a common factor to the sequences x and w if u is a factor of x and factor of w, and we write $u \in CF(x,w)$.

Definition 6. The sequence $b=b^1b^2\ldots b^m$ is the reverse sequence of $q=q^1q^2\ldots q^n$ if and only if :
1. m=n (i.e., |b|=|q|),
2. $\forall k \in \mathbb{N}^*$ such that $1 \le k \le n$: $b^k = q^{n-k+1}$,

$\mathbb{N}$ is the set of naturals and $\mathbb{N}^* = \mathbb{N} - \{0\}$.

The operator Inv denotes the reverse of a sequence, i.e., b= Inv(q). It has the two following properties:
1. If b=Inv(q) then q=Inv(b).
2. If s=uv then: Inv(s)=Inv(v)Inv(u).

When the path of an AGV is known (for example planned by the *cfstp*), it can be described by a sequence defined on N, the alphabet of nodes.
As an AGV advances in its mission trip, the preceding sequence describing its path should be reduced and updated. For this, 3 sequences S(V), D(V) and R(V) will be associated to each operational vehicle V, where:
- S(V) is the static sequence that describes the whole path of V relating its source and destination nodes.
- D(V) is the dynamic sequence that describes the whole set of the remaining nodes to be visited by V at the actual time t_{now}. Notice that D(V) is a right factor of S(V) with $D^m(V) = S^n(V)$, where m and n are the sequences length of D(V) and S(V) respectively.
- R(V) is the realised sequence which describes all the nodes that have been already visited by V at time $t<t_{now}$. R(V) is a left factor of S(V), and S(V)=R(V)D(V).

Let D(V) and D(U) be the dynamic sequences associated to the vehicles V and U respectively.

The common-path of the vehicles V and U at the time t_{now} is given by the longest common sub-sequence (LCS) of :
- D(V) and D(U) if U and V are traveling this common-path into the same direction,
- D(V) and Inv(D(U)) if V and U cross the common-path into opposite directions.

The continuous common-paths of V and U are defined by the longest common factors (CF) of the

sequences D(V) and D(U) or Inv(D(U)). The first continuous common-path between V and U is given by the first common factor of D(V) and D(U) (or Inv(DU)), which will be denoted by $W(V \wedge U)$. It is calculated by scanning the two sequences from left to right. If N=D^1(V) is a common node to V and U, then $W(V \wedge U)$ is the left factor of D(V) and a factor of D(U) (or Inv(D(U)).

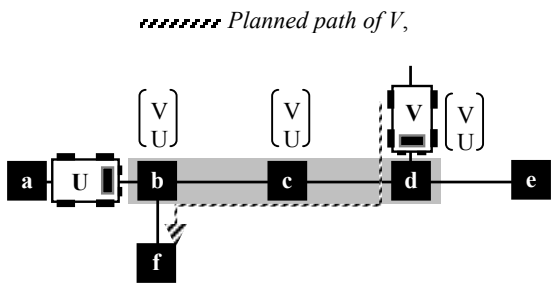


Fig. 4. Illustrative example.

(Fig. 4.) shows an example of a common path. If U is on [a,b] then Inv(D(U))=edcb and Inv(S(U))=edcba, while D(V)=dcbf. Then $W(V \wedge U) = dcb = \mu(V \wedge U)$ However, if U is on the common path [b,d], say between the nodes b and c, then Inv(D(U))=edc and thus $W(V \wedge U) = dc$ is a proper factor of $\mu(V \wedge U) = dcb$.

In what follows, only the case where V and U have opposite directions on their common path will be considered. Let's denote by $\mu(V \wedge U)$ the first common factor of the sequences D(V) and Inv(S(U)).

Corollary 1. The AGV U is outside its common-path with V if and only if :

$$1^{st}LCF(D(V), Inv(D(U)) = 1^{st}LCF(D(V), Inv(S(U)))$$

Where 1^{st}LCF is the first Longest Common Factor, and N=D^1(V) is the first common node of V and U.

We start by demonstrating the corollary, then a proof of the theorem will be given.

Proof of corollary 1:
It is sufficient to show that if an AGV is inside the preceding common-path then $W(V \wedge U)$ is a proper factor of $\mu(V \wedge U)$ and inversely. Two cases are possible :

The vehicle U is on a :

$$\begin{cases} \text{a link } [n_1, n_2] \text{ and is going toward the node } n_2 \text{...(1)} \\ \text{a node say } n_2 \text{.....(2)} \end{cases}$$

The nodes n_1 and n_2 are given by two elements of the sequences R(U) and D(U) :

$$\begin{cases} n_1 = R^{|R(U)|}(U) = Inv(R(U))^1 \ \& \ n_2 = D^1(U) = Inv(D(U))^{|D(U)|}, \\ \quad \text{for case 1.... (3)} \\ n_2 = R^{|R(U)|}(U) = Inv(R(U))^1, \text{ for case 2 ..... (4)} \end{cases}$$

If the AGV U is on its common-path with AGV V, then we will have :

$$\begin{cases} n_1, n_2 \in \text{ common path of U and V, Case 1 ..... (5)} \\ n_2 \in \text{ common path of U and V, Case 2 ..... (6)} \end{cases}$$

From (3) and (5) (case 1) :
$n_1 = R^{|R(U)|}(U)$ and $n_1 \in$ common path , and from (4) and (6) (case 2) :
$n_2 = R^{|R(U)|}(U)$ and $n_2 \in$ common path .
Thus, there is a sequence Q, $Q \neq 1$ (at least equal to n_1 for case 1, or n_2 for case 2), such that Q is a factor of D(V) and right factor of R(U) (i.e., left factor of Inv(R(U)), and $\mu(V \wedge U) = W(V \wedge U)Q$. That implies that the sequence W is a proper factor of μ.
For the reverse implication, we can proceed in the same manner. If $W(V \wedge U)$ is a factor of $\mu(V \wedge U)$ then there is a non empty sequence Q such that $\mu(V \wedge U) = W(V \wedge U)Q$. Thus, Q is a factor of D(V), of Inv(S(U)), but not Inv(D(U)). It means that Q is a right factor of R(U), and it describes the set of nodes already visited by AGV U. These nodes are common to AGV V, which means that U is inside its common-path (defined previously) with V.

Proof of theorem 1:
We will show that if the preceding condition is not satisfied, a conflict may occur. If $W(V \wedge U)$ (W for short) is a proper factor of $\mu(V \wedge U)$ (μ for short), then there is a non-empty sequence Q such that : μ=WQ. We have the followings:

$$W = 1^{st}CF(D(V), Inv(D(U))) \text{ while } W^1 = D^1(V), \quad (1)$$

$$\mu = 1^{st}CF(D(V), Inv(S(U))) = WQ, \quad (2)$$

$$Inv(S(U)) = Inv(D(U))Inv(S(U)) \quad (3)$$

(1) and (2) imply that Q is a factor of Inv(S(U)) but not of Inv(D(U)). It means that Q is a left factor of Inv(R(U)), i.e., ∃Q` such that Inv(R(U))=QQ`.
Since there are common nodes between U and V outside the path described by sequence W and given by the elements of Q, it is easy to proof that:

$$W^1 = D^1(V) = \mu^1, \quad (4)$$

$$W^{|W|} = Inv(D(U))^{|D(U)|} = D^1(U), \quad (5)$$

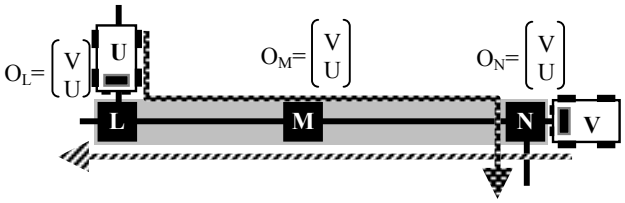
$$Q^1 = Inv(R(U))^1 = R(U)^{|R(U)|} = D(V)^{|W|+1}, \quad (6)$$

and,
U is actually between the nodes given by Q^1 and $W^{|W|}$, (7)

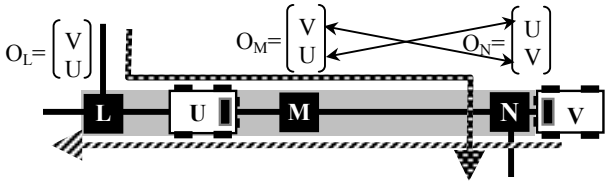
If the AGV U is delayed in front of V on the common-path given by the elements of the sequence W, the resulting node's crossing order is as follows:
(A) $\forall i \in \{1,\ldots, |W|\}$, n=$W^i$: $o_n(U) > o_n(V)$, where $o_n(X)$ is the crossing order of the vehicle X at the node n, and
(B) $\forall i \in \{1,\ldots, |Q|\}$, n=$Q^i$: $o_n(V) > o_n(U)$, because the nodes n have already been crossed by U.

Since μ is the concatenation of the sequences W and Q, and the node's crossing order (or the AGVs' priorities) is reversed between W and Q, then an unavoidable conflict may result between U and V along this path.
Example:

Planned path of V, *Planned path of U*



(a) U outside its common path with V



(b) U inside its common path with V

Fig. 4. Illustrative example of the theorem 1.
Case a: U can be delayed in front of V on N, and should be also delayed on their common path.
Case b: If U is delayed in front of V on N, an unsafe state will be reached. It leads to an unavoidable deadlock state if V starts crossing N.

The preceding common path can also been crossed by other vehicles, which can be affected by the delay action of the AGV U (see fig. 5). Thus, additional tests are necessary to avoid unsafe states.

Definition 7. We call intermediate vehicle of V and U, each AGV V_x having to cross some nodes belonging to the common-path of U and V, and which has a priority comprised between that of V and U, i.e., priority(V)<priority(V_x)<priority(U).

In what follows, $\Im$ denotes the intermediate vehicles of V and U on their common-path. It contains also V_c, medium vehicles of U and V_x ($V_x \in \Im$) outside the common-path of U and V.

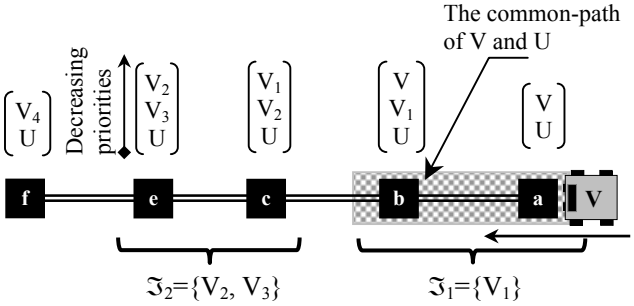


Fig. 5. Definition of the set $\Im$ of intermediate AGVs with $\Im=\Im_1 \cup \Im_2$

From (fig. 6.), it is clear that if U is delayed in front of V on their common path given by nodes {a, b}, conflicting priorities may result between U and V_1 on nodes b and c. Even if U is delayed in front of V_1 on node c, another unsafe state will be reached because of the conflicting priorities between U and V_2 on nodes c and e.

Theorem 2. The second necessary condition for the delay-action of the late AGV U to be possible without inducing conflicts is that U is outside each of its common-path with $V_x \in \Im$. The delay-action must be done on each common-path between U and the vehicles of $\{V\} \cup \Im$, to avoid unsafe states.

Remarks: theorem 2 can be proved by applying theorem 1 to the vehicles U and V_x (instead of V), where $V_x \in \Im$.
The two preceding conditions taken together constitute a necessary and sufficient condition for a safe AGVs rescheduling.

3.2.3 The Robust Vehicle Delaying Algorithm

Suppose that the common path of V and U has been calculated. Let $\Im$ be the set of medium AGVs of U and V on their common path. Suppose that U is outside this path. In what follows, only the case where V and U have to cross it into opposite directions will be considered here.
The dynamic sequence of the late AGV U can be expressed as a concatenation of three sequences: $D(U) = d_1(U) W(U \wedge V) d_2(U)$, where the sequence $W(U \wedge V)$ describes the common path of U and V.
the sequence $d_1(U)$ describes the set of nodes to be visited by U before reaching the extreme node of the common path described by $W(U \wedge V)$.

START
M= $W(U \wedge V)^{|W(U \wedge V)|}$
for i=$|d_1(U)|$ to 1, step (-1)
n= $d_1^i(U)$
Is there any AGV V_m in O_n which is medium between U and any AGV V_x in the set $\Im$?
If yes then Add V_m to $\Im$
M=n
End if
Next i
If $M = d_1^1(U)$ then
G=$|R(U)|^{|R(U)|}$
Let $\Im_G$ be the set of AGVs having to cross G (after U)
If $\Im_{F_U} \cap \Im \neq \varnothing$ then
the AGV U is inside all its common-paths with $V_x \in \Im \cap \Im_G$ and the delay action is prohibited. Return.
Else
U should be delayed in front of all the AGVs of $\Im$ on each node belonging to the path [N, M], where N is the node where is called the *RVDA* and M is the preceding calculated node.
END

Example: consider the (fig. 5) and suppose that U is on the link [f,e]. The path [N,M] where the delay action of U should take place if possible is [a,e].Here $\Im=\{V_1,V_2,V_3\}$ and $\Im_f=\{V_4\}$ so $\Im_f \cap \Im=\varnothing$ and U is outside each common path with $V_x \in \Im$. However, if instead of V_4, the node f is to be crossed by V_3 then

U is inside its common path with V_3, and an unsafe state will be reached if U is delayed in front of $V_x \in \Im$ on the path [a,e] (see theorem 2).

4. SIMULATION STUDY

To check for the efficiency of our algorithms *RVWA*, *RVRAA* and *RVDA*, a simulation study has been conducted with ARENA package. The studied AGVS is composed of bi-directional mesh-like guide-path of 45 nodes and 60 links and a fleet of 8 AGVs. Each simulation essay is a chain of at least 10 replications. In one replication, each AGV has to realise a set of 100 missions randomly generated.
In order to approach reality, random failures of AGVs are generated in the simulation model. They are characterised by two parameters: the failure rate τ and the mean time between failures MTBF.
Different simulations are done with various system parameters in order to compare the preceding algorithms and bring out the situations where the use of one algorithm is more appropriate than another.

4.1. The influence of the failure rate τ

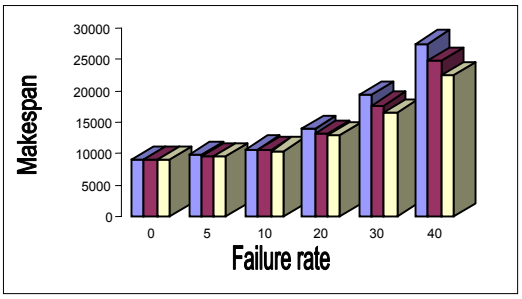


Fig. 6. The evolution of the makespan D according to the failure rate (τ=0, 5%, 10%, 20%, 30%, 40%).

By varying the failure rate and fixing other parameters, it can be concluded that more the failure rate is important, better will be the makespan achieved by *RVRAA* and *RVDA* and that the *RVDA* gives the best results. That confirms the theory since the *RVDA* is more permissive than the two other algorithms.

4.2. The influence of the MTBF

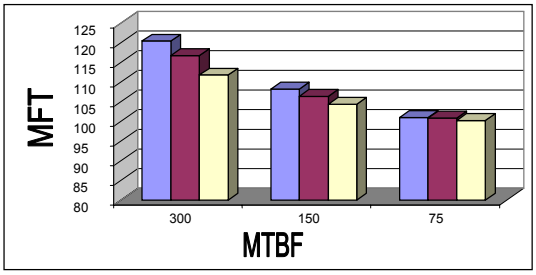


Fig. 7. Evolution of the Mean Flow Time (Mission duration) according to the MTBF (MTBF= 150, 150, 75)

The same conclusion can be made about the influence of the failures frequency on the efficiency of our algorithms. For long and spaced failures, the *RVDA* gives the shortest makespan. However, when the failures are frequent and short, the three algorithms have the same behaviour. That means that is more appropriate to use the simplest one i.e., *RVWA*.

4.3. The influence of number of AGVs on the circuit

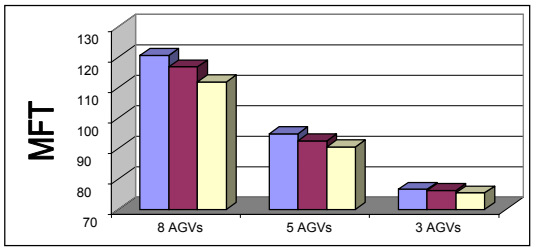


Fig. 8. Evolution of the Mean Flow Time (Mission duration) according to the Number of AGVs on the circuit (Number= 8, 5, 3)

We have conducted other simulations while varying the fleet size and fixing other parameters. From (Fig. 8), it can be seen that the *RVDA* gives the best results when the guide-path is overloaded.

5. CONCLUSION

In this paper, a two-stage robust control for conflict-free routing of bi-directional AGVs has been presented. In order to combine the advantages of the planning and real-time methods, a path planning method to establish the conflict-free fastest routes for AGVs is used in the first stage. However, three algorithms called *RVWA*, *RVRAA* and *RVDA* are used in the second stage to avoid conflicts in real-time. The simulations carried out allowed us to study the efficiency of the preceding algorithms as well as the influence of certain parameters on the realized performances.

REFERENCES

Kim, C.W., and J.M.A. Tanchoco (1991). Conflict-free shortest-time bi-directional AGV routing. In: *International Journal of Production Research,* **29**, 2377-2391.

Krishnamurthy, N.N., R. Batta., and H. Karwan (1993). Developing conflict-free routes for automated guided vehicles. In: *Operation research,* **41**, 1077-1090.

Maza, S., and P. Castagna (2001). Conflict-free AGV routing in bi-directional networks. In: *IEEE International conference on Emerging Technologies and Factory Automation*. 761-764, France.

Maza, S., and P. Castagna (2002). Robust conflict-free routing of bi-directional AGVs. In: *IEEE International conference on systems, man and cybernetics*. Tunisia.

Moorthy, R.L., W. Hock-Guan, N. Wing-Cheong, and T. Chung-Piaw (2003). Cyclic deadlock prediction and avoidance for zone-controlled AGV system. In: *International Journal of Production Economics*. **83**, 309-324.

Oboth, C., R. Batta, M. Karwan (1999). Dynamic conflict-free routing of automated guided vehicles. In: *International Journal of Production Research*. **37**, 2003-2030.

Pia Fanti, M. (2002). Event-based controller to avoid deadlock and collisions in zone-control AGVS. In: *International Journal of Production Research*. **40**, 1453-1478.

Reveliotis, S.A. (2000). Conflict resolution in AGV systems. In: *IIE Transactions*. **32**, 647-659.