Time Windows Based Dynamic Routing in Multi-AGV Systems

Nenad Smolic-Rocak, *Student Member, IEEE*, Stjepan Bogdan, *Member, IEEE*, Zdenko Kovacic, *Member, IEEE*, and Tamara Petrovic, *Student Member, IEEE*

***Abstract*—This paper presents a dynamic routing method for supervisory control of multiple automated guided vehicles (AGVs) that are traveling within a layout of a given warehouse. In dynamic routing a calculated path particularly depends on the number of currently active AGVs' missions and their priorities. In order to solve the shortest path problem dynamically, the proposed routing method uses time windows in a vector form. For each mission requested by the supervisor, predefined candidate paths are checked if they are feasible. The feasibility of a particular path is evaluated by insertion of appropriate time windows and by performing the windows overlapping tests. The use of time windows makes the algorithm apt for other scheduling and routing problems. Presented simulation results demonstrate efficiency of the proposed dynamic routing. The proposed method has been successfully implemented in the industrial environment in a form of a multiple AGV control system.**

***Note To Practitioners*—The automated warehouses are the facilities which can be equipped with stationary and mobile robotic sections (e.g. palletization workcells and unmanned forklifts, respectively). The goals, such as the maximal production throughput, efficient utilization of all robotic subsystems and collision avoidance with accommodation to dynamic changes within the manufacturing process itself have been achieved by designing and implementing a supervisory controller whose main characteristics are the ability of dynamic routing and scheduling in accordance with the current status of a given shop floor layout. The maximal throughput is provided by applying a shortest path algorithm, while potential conflicts and deadlocks are resolved by using the time windows insertion and time windows overlapping check algorithms. Because of the safety reasons, the proposed supervisory controller is somewhat restrictive, allowing only one vehicle to reside at one arc at the time. The described multi-AGV control system was successfully installed and put in operation at several European factories.**

***Index Terms*—Automated guided vehicles, dynamic routing, path feasibility, time windows.**

I. Introduction

Introduction of automated guided vehicles (AGVs) into automated materials handling systems, flexible manufacturing systems and containers handling systems has brought many notable advantages (increased system throughput, reduced operational costs, and provision of prerequisites for consistent execution of predetermined tasks), but it also requires the use of effective on-line supervisory control strategies able to solve conflict and deadlock problems in the system layout.

Manuscript received February 15, 2008; revised September 24, 2008. First published May 02, 2009; current version published January 08, 2010. This paper was recommended for publication by Associate Editor T.-E. Lee and Editor Y. Narahari upon evaluation of the reviewers' comments. This work was supported by a grant from Sitek S.p.A., Verona, Italy, and by a grant from the Croatian Ministry of Science, Education and Sports.

N. Smolic-Rocak is with Visor d.o.o., 10000 Zagreb, Croatia (e-mail: nenad.smolic-rocak@visor.hr).

S. Bogdan, Z. Kovacic, and T. Petrovic are with the Laboratory for Robotics and Intelligent Control Systems, Department of Control and Computer Engineering in Automation, Faculty of Electrical Engineering and Computing, University of Zagreb, HR-10002 Zagreb, Croatia (e-mail: stjepan.bogdan@fer.hr; zdenko.kovacic@fer.hr; tamara.petrovic@fer.hr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2009.2016350

One can notice that various methods for AGV routing and scheduling are currently in use [1]–[4], but new methods, which employ faster and computationally more efficient algorithms, are still the subject of intensive research. Very good reviews of the field can be found in [5]–[7]. Many modern AGVs are free-ranging, equipped with a decentralized control systems based on heuristic-behavior and fuzzy algorithms [8]–[10]. However, due to very restrictive security issues, traditional centralized decision-making structures still dominate in the industrial implementations.

AGVs routing and scheduling can be performed either on-line or off-line. Off-line planning requires the knowledge of all tasks prior to execution of an algorithm, while on-line planning allows new tasks to appear although the planning has been already done. Generally, two approaches to on-line routing, static and dynamic, are prevailing in the literature. While the static routing is focused on the spatial dimension of the routing problem (determination of paths in the space domain), the dynamic routing treats the routing as a time-space problem dealing with determination of paths feasible both in space and time (feasible in the sense of conflict and deadlock free execution of the path).

In [11] authors propose a deadlock resolution by using a technique frequently adopted for vehicle management in AGV systems, so called the zone control [12], [13]. The guide paths are subdivided in disjoint zones representing workstations, intersections of several paths or simple parts of a straight lane, and the admittance of a vehicle to any zone must be previously authorized by the control, in order to avoid collisions and deadlocks.

Petri nets, as a powerful tool for DES analysis and synthesis, are often used in investigation of AGV control systems properties [14], [15]. In [16] two types of AGVs are modeled by Petri nets: centralized—an AGV is exclusively assigned to a job until the job is completed, and decentralized—an AGV is shared by multiple jobs. In [17] authors also use Petri nets in order to decouple the upper level planning from the lower level logical control in an AGV system.

A different approach to routing and scheduling is used in [18]. The proposed method is based on a well known Banker's algorithm. Even though conflict and deadlock are avoided, the described technique requires that, at any state, at least one AGV can finish its mission with no movement of other AGVs in the system.

The aim of the dynamic routing method, proposed in this paper, is to determine the shortest path (in terms of a traveling time) by solving a shortest path problem with time windows (SPPTW) [19], [20]. The results presented herein are extension of the work presented in [21] and [22]. Since the number of active vehicles and the corresponding missions change in time dynamically, the proposed routing method makes the determined shortest path to become feasible by means of a time windows elongation, resulting in conflict-free and deadlock-free paths for all active vehicles. In the implemented automated warehouse control system, we have had to deal with the fact that any path allocated to the active AGV could be changed during the mission execution [23].

II. A Multi-AGV Plant Layout Representation Review Stage

Usually, when it comes to the mathematical analysis and supervisory control algorithm design, a layout of a considered multi-AGV plant can be represented by a graph whose intersections and ends of paths are referred to as graph nodes, while paths themselves are represented by arcs.

Let us use for this purpose a directed graph $G = (N, A)$, which contains a set of nodes, $N = \{n_1, n_2, \ldots, n_n\}$, and a set of weighted arcs, $A = \{a_1, a_2, \ldots, a_m\}$. The meaning of arc weight may vary from one application to the other depending on the system requirements and supervisory control algorithm design. Respecting the fact that the proposed dynamic routing is based on time windows, the arc traverse time is a variable chosen to define the arc weight.

A minimal traverse time of arc a_j for vehicle r_i, corresponding to the nominal arc weight $\hat{w}_{ij}$, is defined as

$$\hat{w}_{ij} = \frac{l_j}{v_{ij}} \tag{1}$$

where l_j is the length of arc a_j, and v_{ij} is the maximal admissible speed for vehicle r_i on arc a_j.

In a directed graph $G = (N, A)$, node n_1 (arc a_1) is called the *upstream node* (*arc*) to node n_2 (arc a_2), if there exists arc $(n_1, n_2) \in A$ (node $(a_1, a_2) \in N$). The upstream node (arc) is denoted as $n_1 \rightarrow n_2 (a_1 \rightarrow a_2)$. Consequently, node n_2 (arc a_2) is called the *downstream node* (*arc*) to node n_1 (arc a_1).

Having a directed graph $G = (N, A)$, an *arc adjacency matrix* $\mathbf{G}$ is defined as a matrix with a number of rows and columns equal to the number of arcs in G, with element g_{ij} equal to 1 if arc a_i is upstream to arc a_j, otherwise it is equal to 0. A set of active vehicles (vehicles with assigned missions) is denoted as $R_a = \{r_i : r_i \in R\}$, where $R = \{r_1, r_2, \ldots, r_r\}$ is a set of vehicles operating in the plant layout. A set of active missions is defined as $M_a = \{m_i : m_i \in M\}$, where M is a set of all possible missions that can be requested by the supervisor.

A path is a set of arcs, $\sigma = \{a_j : a_j \in A\}$. The *weight of the path* σ is equal to the release time of its destination arc d_σ, that is, $W(\sigma) = {}^{\text{out}}t_{d_\sigma}$. A set of all paths that connect origin arc o_i and destination arc d_i of mission m_i is $\mathbf{\Sigma}_i = \{{}^i\sigma_1, {}^i\sigma_2, \ldots, {}^i\sigma_q\}$.

Taking into account that the path and the mission priority of vehicle r_i can be changed during mission execution, mission m_i is defined as

$$m_i(t) = \left(o_i, d_i, {}^i\hat{\sigma}(t), P_i(t), r_i\right) \tag{2}$$

where ${}^i\hat{\sigma}(t) \in \mathbf{\Sigma}_i$ is the shortest *feasible* path between o_i and d_i, $P_i(t)$ is a changeable mission priority (higher priority mission corresponds with lower value).

Mission priority $P_i(t)$ is calculated according to the relation

$$P_i(t) = \begin{cases} \min\left[\frac{t_{di} - t}{W({}^i\hat{\sigma}) - t}, P_{i0}\right] & for\ W({}^i\hat{\sigma}) \neq \infty \\ -\infty & for\ W({}^i\hat{\sigma}) = \infty \end{cases} \tag{3}$$

where t_{di} is a due time of mission m_i, and P_{i0} is an initial mission priority.

Initially assigned missions' priorities $\{P_{i0}\}$ are recalculated by the dispatching system each time the request for a new mission arrives or current paths become unviable.

When m_i is finished and there are no new mission assignments for the vehicle, an *idle positioning mission* is activated and vehicle r_i is driven to the nearest *idle positioning arc* (a dead-end arc which is treated only as o_i or d_i). Being on the idle positioning arc, the vehicle is waiting for a new mission assignment.

We assume that a vehicle can reside only on arcs, as well as we assume that only one vehicle at the time is allowed to occupy one arc. Assigned to mission m_i, vehicle r_i occupies particular arc a_j during a *time window* w_{ij} defined as

$$w_{ij} = {}^{\text{out}}t_{ij} - {}^{\text{in}}t_{ij}, \quad w_{ij} \geq \hat{w}_{ij} \tag{4}$$

where ${}^{\text{out}}t_{ij}$ is a release time of arc a_j from mission m_i, and ${}^{\text{in}}t_{ij}$ is a time of entry on arc a_j by mission m_i.

Time windows, as well as release times and entry times of arc a_j, can be represented in a form of *time vectors*

$$\mathbf{w}_j = [w_{ij}], \ {}^{\mathbf{in}}\mathbf{t}_j = \left[{}^{\text{in}}t_{ij}\right], \ {}^{\mathbf{out}}\mathbf{t}_j = \left[{}^{\text{out}}t_{ij}\right] \tag{5}$$

where the first component corresponds to the highest priority mission, nth component to the lowest priority mission and $n = |M_a|$, i.e. the dimension of all three vectors is equal to the number of active missions.

Dimension n varies in time, since the number of active missions is changing dynamically. The components of vector $\mathbf{w}_j$, related to active missions that do not use arc a_j, are set to zero, while the related components of vectors ${}^{\mathbf{in}}\mathbf{t}_j$ and ${}^{\mathbf{out}}\mathbf{t}_j$ are set to ∞.

From time vectors defined as in (5), we know which missions visit which arcs, but we cannot tell directly in which order. For the purpose of time windows insertion, we must sort components of time vectors in a chronological order.

In this way, vector $\mathbf{x} = [x_i]$ can be converted into a *sorted vector* $\langle\mathbf{x}\rangle = [x_i]$, where $\langle x\rangle_i = x_i \leq \langle x\rangle_{i+1} = x_{i+1}$.

III. The Feasible Path Determination

The method described herein offers a solution to the Shortest Path Problem with Time Windows (SPPTW). The proposed method checks the mission candidate paths (from the set of off-line calculated paths $\mathbf{\Sigma}_i$) by using time windows to check if determined paths are feasible (herein we are not concerned with calculation of candidate paths—they could be obtained off-line by one of the shortest path algorithms, such as Dijkstra [24], string algebra [25], Floyd–Warshal, etc.). Viability of a particular path is evaluated by a time windows insertion followed by a time windows overlap (conflict) test. In the case of conflict, the algorithm iteratively reinserts time windows until conflicts disappear or an overlap is present only on the path's origin arc, indicating that the candidate path is not feasible. The procedure is repeated for all candidate paths and the result is a set of executable paths. The final task of the algorithm is to choose the shortest one among executable paths in terms of a time required for a vehicle in mission to get from the origin to the destination arc.

When a new mission m_m is requested at the moment t_m, the supervisor is looking for an idle vehicle r_{m} to assign it to that mission (with initial mission priority P_{m0}). As a goal of dynamic routing is to determine the shortest path for mission m_m under current state of the system, all candidate paths should be checked for feasibility.

A. Initialization of Time Vectors (Step 1)

The first step in the iterative procedure for a feasibility test and a release time determination of paths in $\mathbf{\Sigma}_m$, is an initialization of time vectors. Let us choose a candidate path $\sigma_i \in \mathbf{\Sigma}_m$. For each arc $a_j \in \sigma_i$, its time vectors are initialized as

$$\begin{aligned} \mathbf{w_j} &= [w_{1j} \quad w_{2j} \quad \ldots \quad \hat{w}_{mj} \quad 0 \quad 0]^T \\ {}^{\mathbf{in}}\mathbf{t_j} &= \left[{}^{\text{in}}t_{1j} \quad {}^{\text{in}}t_{2j} \quad \ldots \quad {}^{\text{in}}t_{mj} \quad \infty \quad \infty\right]^T \\ {}^{\mathbf{out}}\mathbf{t_j} &= \left[{}^{\text{out}}t_{1j} \quad {}^{\text{out}}t_{2j} \quad \ldots \quad {}^{\text{out}}t_{mj} \quad \infty \quad \infty\right]^T. \end{aligned}$$

During initialization, the components of ${}^{\mathbf{out}}\mathbf{t_j}$ whose values were less than t_m, are set to ∞ (as well as their counterparts in ${}^{\mathbf{in}}\mathbf{t_j}$). Since they correspond to the missions that occupied arc a_j before a new mission was requested, they do not influence time windows settings. In the case ${}^{\text{in}}t_{ij} \leq t_m$ and ${}^{\text{out}}t_{ij} > t_m$, vehicle r_i occupies arc a_j at the moment of request t_m, and these components of time vectors remain unchanged. Components of vector $\mathbf{w}_j$ that belong to the missions with lower priorities than mission m_m are set to 0. In the same time, all components of vectors ${}^{\mathbf{in}}\mathbf{t_j}$ and ${}^{\mathbf{out}}\mathbf{t_j}$ that correspond to those missions are set to ∞. Components that belong to mission m_m, ${}^{\text{in}}t_{mj}$ and ${}^{\text{out}}t_{mj}$, are unknown values that have to be determined by the dynamic routing.

The vehicle r_m assigned to the new mission m_m occupies the origin arc o_{m} at the moment of request. Therefore, the entry time of o_{m} is set to be equal to the request time t_m. The release time of o_{m} depends on

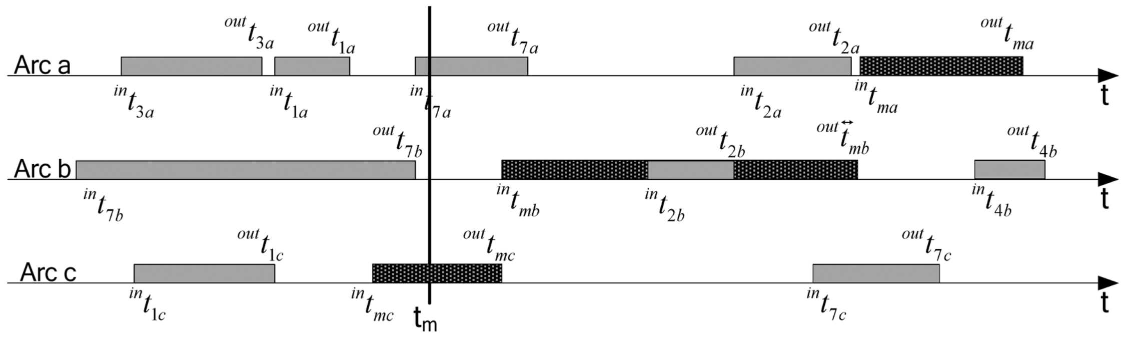

Fig. 1. Time windows overlap.

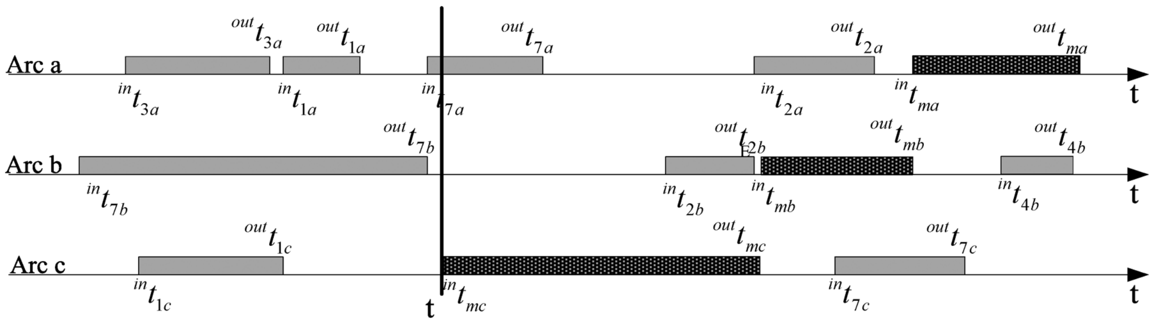

Fig. 2. Reinserted time windows with no overlap.

the average vehicle speed v_{mo_m}, and the remaining length of the arc at the moment of request, $\tilde{l}_{o_m}$, that is, $\tilde{w}_{mo_m} = \tilde{l}_{o_m}/v_{mo_m}$.

Accordingly, for the origin arc $o_m \in \sigma_i$ we set

$$\mathbf{^{in}t_{o_m}} = \left[{}^{\text{in}}t_{1o_m} \;\; {}^{\text{in}}t_{2o_m} \;\; \ldots \;\; t_m \;\; \infty \;\; \infty\right]^T$$
$$\mathbf{^{out}t_{o_m}} = \left[{}^{\text{out}}t_{1o_m} \;\; {}^{\text{out}}t_{2o_m} \;\; \ldots \;\; t_m + \tilde{w}_{mo_m} \;\; \infty \;\; \infty\right]^T.$$

B. Insertion of Time Windows (Step 2)

Having time windows of all arcs that belong to the candidate path initialized, starting from the second arc on the path, we are looking on each arc $a_j \in \sigma_i$ for the first available time window that fulfils two requirements: a) it is wide enough to accommodate vehicle r_m for a predetermined period, and b) its entry time ${}^{\text{in}}t_{mj}$ is set after the release time of the upstream arc ${}^{\text{out}}t_{mi}$, $i \to j$.

In the case that

$$\left[\left\langle {}^{\text{in}}t_j \right\rangle_1 - t_m\right] > \hat{w}_{mj} + \varepsilon_{mj}$$

and

$$\left[\left\langle {}^{\text{in}}t_j \right\rangle_1 - (\hat{w}_{mj} + \varepsilon_{mj})\right] > {}^{\text{out}}t_{mi} \text{ for } i \to j \tag{6}$$

then ${}^{\text{in}}t_{mj} = {}^{\text{out}}t_{mi}$ and ${}^{\text{out}}t_{mj} = {}^{\text{in}}t_{mj} + \hat{w}_{mj}$; otherwise, the index of the first available time window is determined by

$$p = \arg\min_{\ell} \left\{ \left\langle {}^{\text{in}}t_j \right\rangle_\ell : \left[\left\langle {}^{\text{in}}t_j \right\rangle_{\ell+1} - \max\left(\left\langle {}^{\text{out}}t_j \right\rangle_\ell, {}^{\text{out}}t_{mi}\right)\right] > \hat{w}_{mj} + 2\varepsilon_{mj}, i \to j, \ell = 1, n-1 \right\} \tag{7}$$

where n is a number of time vector components that are $\neq \infty$ and ε_{mj} is a *safety time* of mission m_m on arc a_j. The safety time, which depends on the AGV speed, the AGV construction and the warehouse layout, should be added between two consecutive time windows, in order to prevent a collision of vehicles in the node. Its value is usually set to 1–5% of $\hat{w}_{mj}$. Once p is determined, entry and release times of mission m_m on arc a_j are calculated as

$$\begin{aligned} {}^{\text{in}}t_{mj} &= \max\left(\left\langle {}^{\text{out}}t_j \right\rangle_{p-1} + \varepsilon_{mj}, {}^{\text{out}}t_{mi}\right) \\ {}^{\text{out}}t_{mj} &= {}^{\text{in}}t_{mj} + \hat{w}_{mj}. \end{aligned} \tag{8}$$

It may happen that a time windows distribution on arc a_j is so dense that ${}^{\text{in}}t_{mj}$ cannot be determined, i.e., none of relations (10) and (11) give an answer where to insert a time window for a new mission. In that case a new time window is set after the last time window on arc a_j, that is

$${}^{\text{in}}t_{mj} = \left\langle {}^{\text{out}}t_j \right\rangle_n + \varepsilon_{mj}, {}^{\text{out}}t_{mj} = {}^{\text{in}}t_{mj} + \hat{w}_{mj}. \tag{9}$$

C. Time Windows Elongation and Overlaps (Step 3)

As mentioned before, by leaving one arc, a vehicle enters another one, which means that it can only reside on arcs (it cannot occupy nodes). The following equation should be fulfilled for all arcs on the path

$${}^{\text{in}}t_{mj} = {}^{\text{out}}t_{mi}, \quad i \to j. \tag{10}$$

In order to check if path σ_i is feasible, first we have to extend inserted time windows to meet requirement (10). The time window elongation on arc a_j yields

$$w_{mj} = \hat{w}_{mj} + {}^{\text{in}}t_{mi} - {}^{\text{out}}t_{mj}, \quad j => i. \tag{11}$$

As arc length is defined by the warehouse layout, a time window can be extended only by changing a vehicle speed. As a consequence, an arc release time is changed

$${}^{\text{out}}\overleftrightarrow{t}_{mj} = w_{mj} + {}^{\text{in}}t_{mj}. \tag{12}$$

Time vectors of arc a_j become

$$\mathbf{w_j} = [w_{1j} \quad w_{2j} \quad w_{3j} \quad \ldots \quad w_{mj} \quad 0 \quad 0]^T$$
$$\mathbf{{}^{in}t_j} = \left[{}^{\mathrm{in}}t_{1j} \quad {}^{\mathrm{in}}t_{2j} \quad {}^{\mathrm{in}}t_{3j} \quad \ldots \quad {}^{\mathrm{in}}t_{mj} \quad \infty \quad \infty\right]^T$$
$$\mathbf{{}^{out}\overleftrightarrow{t}_j} = \left[{}^{\mathrm{out}}t_{1j} \quad {}^{\mathrm{out}}t_{2j} \quad {}^{\mathrm{out}}t_{3j} \quad \ldots \quad {}^{\mathrm{out}}\overleftrightarrow{t}_{mj} \quad \infty \quad \infty\right]^T.$$

The time window elongation can cause an *overlap*, also called a *conflict*; a situation when two (or more) vehicles request an arc over the same time period. The situation when an overlap takes place after applying time window extension is shown in Fig. 1. Since ${}^{\mathrm{in}}t_{ma} \neq {}^{\mathrm{out}}t_{mb}$, a newly inserted time window on arc b, which belongs to mission m_m, has been widen. This action caused an overlap with the time window of mission m_2, which indicates that if the speed of vehicle m_m is reduced in order to release arc b just in the moment when it enters arc a, then it will collide with vehicle r_2.

Having in mind the previous example, once all time windows that belong to a particular path are extended according to (12), new time vectors should be checked for overlaps, starting from the origin arc on the path. If

$$\left\{\left\langle {}^{\mathrm{in}}t_j \right\rangle_\ell : \left[\left\langle {}^{\mathrm{in}}t_j \right\rangle_{\ell+1} - \left\langle {}^{\mathrm{out}}\overleftrightarrow{t}_j \right\rangle_\ell\right] < 0, \quad \ell = 1, n-1\right\} = \emptyset \tag{13}$$

then there are no overlaps on arc a_j.

In case (13) is not satisfied, the first arc with an overlap should be detected and the time windows should be reinserted, starting from the arc with an overlap all the way to the last arc on the path. A new time window is inserted on the arc with an overlap by using (8), only this time index p is calculated according to the following relation:

$$p = \arg\min_\ell \left\{\left\langle {}^{\mathrm{in}}t_j \right\rangle_\ell : \left[\left\langle {}^{\mathrm{in}}t_j \right\rangle_{\ell+1} - \max\left(\left\langle {}^{\mathrm{out}}t_j \right\rangle_\ell, {}^{\mathrm{out}}t_{mi}\right)\right] > \hat{w}_{mj} + 2\varepsilon_{mj}, i \rightarrow j, \ell = q, n-1\right\} \tag{14}$$

where q corresponds to the last time window involved in the overlap, that is

$$q = \arg\max_\ell \left\{\left\langle {}^{\mathrm{in}}t_j \right\rangle_\ell : \left[\left\langle {}^{\mathrm{in}}t_j \right\rangle_{\ell+1} - \left\langle {}^{\mathrm{out}}\overleftrightarrow{t}_j \right\rangle_\ell\right] < 0, \ell = 1, n-1\right\}. \tag{15}$$

Since a new time window cannot be placed upstream of the time window q, in (14) only time windows that follow after q are checked. After time windows are reinserted on all arcs, they should be checked for overlaps repeatedly until a) there are no overlaps or b) overlap occurs on the origin arc. In case a) the path σ_i is feasible and its weight is equal to the destination arc release time. In case b) the path σ_i is not feasible and its weight is set to. The final form of time windows for path $\sigma_m = \{c, b, a\}$ is shown in Fig. 2. Its weight is $W(\sigma_m) = {}^{\mathrm{out}}t_{ma}$.

In case situation on the shop floor is such that the feasible path cannot be determined, the algorithm is waiting for an event triggered by the release of some arc (i.e. the change in time vectors) and then restarts routing.

Deadlock prevention is a direct result of mission planning. A mission is planned in the way that occurrence of deadlock is not permitted. In case of deadlock at least one of the currently active missions would have weight (completion time) $W(\sigma_m) = {}^{\mathrm{out}}t_{mdest} = \infty$, i.e. one of the missions can not be completed. However, according to the algorithm only missions with $W(\sigma_m) < \infty$ are asserted as feasible. Since only feasible missions are downloaded to the AGVs' controllers, the vehicles actually do not make decisions whether to enter particular arc or not; whole mission is know in advance, i.e. deadlock occurrence is not possible since downloaded missions are deadlock free. ${}^{\mathrm{out}}t_{ma}$.

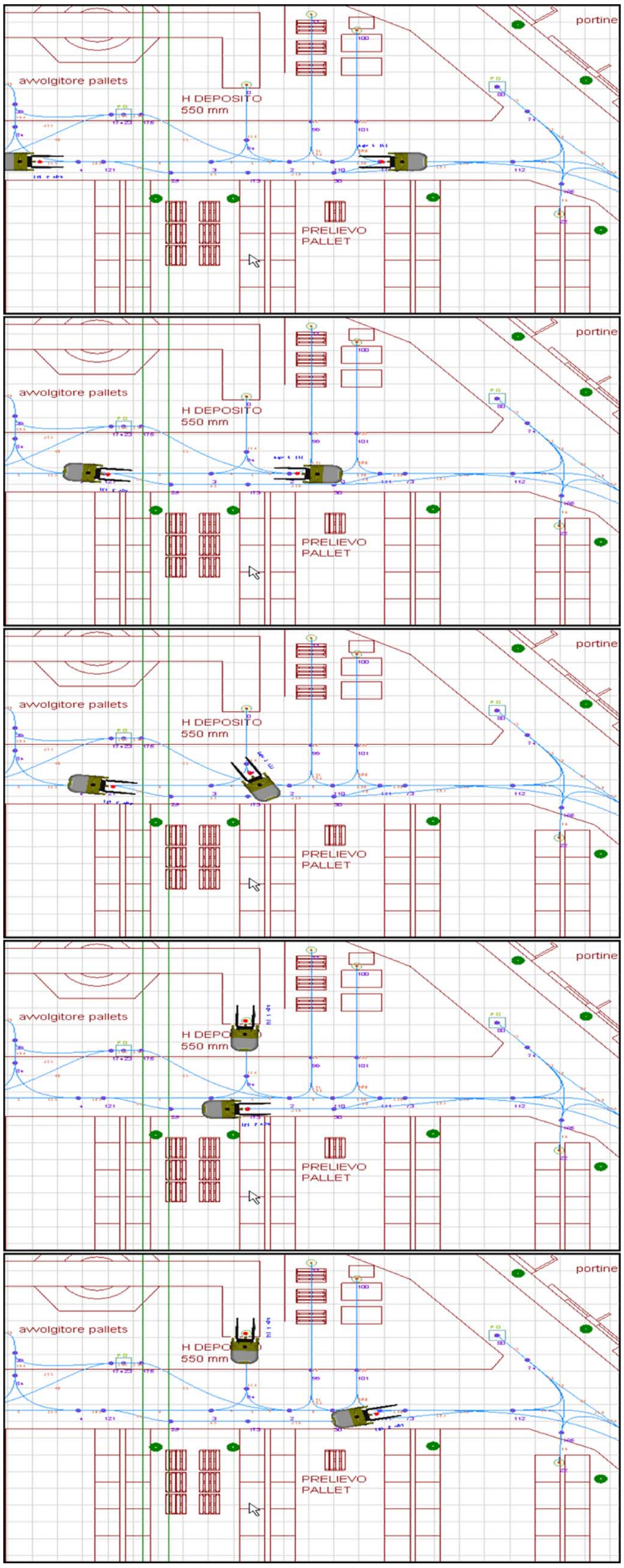

Fig. 3. Tracking of the vehicles in the system GUI.

Here should also be noted that active missions of priority lower than the priority of the mission m_m are rerouted using the same procedure as for the mission m_m.

IV. An Industrial Implementation

Prior to an industrial implementation the presented method for dynamic routing in the system with multiple AGVs was thoroughly tested by using simulators described in [26] and [27]. So far, six installations have been successfully put into practice in Italy, Spain, and Austria (movie clips can be downloaded from http://larics.rasip.fer.hr/movies). The software can be divided into the following main modules: layout design, mission definition, vehicle scheduling and routing, mission control, and communication.

Graphical user interface (GUI) is partly shown in Fig. 3. Vehicles are tracked and revealed in GUI while moving through working environment. Nodes, arcs and parts of equipment are depicted as well in order to allow easier conception of the current status of the system. The interface enables an operator to monitor the overall system performance, as well as to send requests and commands to an AGV (in case the AGV is in the manual mode of operation). There exist several measures for the system performance [28], [29]. In our case the system performance is a function of the total vehicle travel time and the total vehicle idle time (including the empty vehicle travel time). On field results demonstrate superior results compared with previously used software.

V. Conclusion

This paper presents a dynamic routing method for an industrial shop floor equipped with multiple automated guided vehicles (AGVs). The method takes into account the number of active missions and their priorities. The proposed dynamic routing method uses time windows in a vector form to find the candidate paths and then check their feasibility. The feasibility of a particular path is evaluated by time windows insertion and by performing the time windows overlapping tests. The algorithm resolves the time windows conflicts iteratively by inserting new time windows until there are no overlaps or the overlap is present on the path's origin arc, which means that the candidate path is not feasible. As a result, the set of executable paths is formed and the final task of the algorithm is to choose the shortest one in terms of time required for a vehicle to get from the origin to the destination arc.

The proposed method allows circular paths and prevents conflicts and deadlocks that can be caused by head-on situations. The proposed algorithm has been implemented in the form of an industrial laser guided vehicle (LGV) control system and successfully tested on several industrial layouts.

References

[1] A. J. Broadbent, C. B. Besant, S. K. Premi, and S. P. Walker, "Free ranging AGV systems: Promises, problems and pathways," in *Proc. 2nd Int. Conf. Automat. Mater. Handling*, Birmingham, U.K., May 1985, pp. 221–237.

[2] S. C. Daniels, "Real-time conflict resolution in Automated Guided Vehicle scheduling," Ph.D. thesis, Dept. Industr. Eng., Pennsylvania State University, Pennsylvania, 1988.

[3] G. Desaulniers, A. Langevin, and D. Riopel, "Dispatching and conflict-free routing of automated guided vehicles: An exact approach," *Int. J. Flexible Manufact. Syst.*, vol. 15, pp. 309–331, October 2003.

[4] R. H. Möhring, E. Köhler, E. Gawrilow, and B. Stenzel, "Conflict-free real-time AGV routing," in *Proc. 3rd Int. Conf. Appl. Infrastructure Res.*, Berlin, Germany, Oct. 2004, pp. 661–675.

[5] F. Taghaboni-Dutta and J. M. A. Tanchoco, "Comparison of dynamic routing techniques for automated guided vehicle systems," *Int. J. Production Res.*, vol. 33, pp. 2653–2669, 1995.

[6] T. Le-Anh and M. B. M. De Koster, "A review of design and control of Automated Guided Vehicle systems," ERIM Report Series Research in Management, ERS-2004-030-LIS, 2004.

[7] L. Qiu, W. J. Hsu, S. Y. Huang, and H. Wang, "Scheduling and routing algorithms for AGVs: A survey," *Int. J. Production Res.*, vol. 40, pp. 745–760, Feb. 2002.

[8] M. B. Duinkerken, M. van der Zee, and G. Lodewijks, "Dynamic free range routing for automated guided vehicles," in *Proc. 2006 IEEE Int. Conf. Netw., Sens. Contr.*, Fort Lauderdale, FL.

[9] S. Berman and Y. Edan, "Decentralized autonomous AGV system for material handling," *Int. J. Production Res.*, vol. 40, no. 15, pp. 3995–4006, 2002.

[10] D. G. Lindeijer, "Controlling automated traffic agents," Ph.D. dissertation, Technical Univ. Delft, Delft, The Netherlands, 2003.

[11] M. P. Fanti and B. Turchiano, "Deadlock avoidance in automated guided vehicle systems," in *Proc. 2001 IEEE/ASME Int. Conf. Adv. Intell. Mechatron.*, Como, Italy, 2001, pp. 1017–1022.

[12] C.-C. Lee and J. T. Lin, "Deadlock prediction and avoidance based on Petri nets for zone-control automated guided vehicle systems," *Int. J. Production Res.*, vol. 33, pp. 3249–3265, 1995.

[13] M.-S. Yeh and W.-C. Yeh, "Deadlock prediction and avoidance for zone-control AGVS," *Int. J. Production Res.*, vol. 36, no. 10, pp. 2879–2889, 1998.

[14] N. Q. Wu and M. C. Zhou, "Deadlock modeling and control of automated guided vehicle systems," *IEEE/ASME Trans. Mechatron.*, vol. 9, no. 1, pp. 50–57, 2004.

[15] N. Q. Wu and W. Q. Zeng, "Deadlock avoidance in AGV system using colored Petri net model," *Int. J. Production Res.*, vol. 40, no. 1, pp. 223–238, 2002.

[16] D. Y. Lee and F. DiCesare, "Integrated scheduling of flexible manufacturing systems employing automated guided vehicles," *IEEE Trans. Ind. Electron.*, vol. 41, no. 6, pp. 602–610, 1994.

[17] N. Q. Wu and M. C. Zhou, "Shortest routing of bidirectional automated guided vehicles avoiding deadlock and blocking," *IEEE/ASME Trans. Mechatonr.*, vol. 12, no. 1, pp. 63–72, 2007.

[18] S. A. Reveliotis, "Conflict resolution in AGV systems," *IIE Trans.*, vol. 32, no. 7, pp. 647–659, 2000.

[19] M. Savelsbergh and M. Sol, "DRIVE: Dynamic routing of independent vehicles," *Oper. Res.*, vol. 46, no. 4, pp. 474–490, 1998.

[20] J. Yang, P. Jaillet, and H. Mahmassani, "Real-time multi-vehicle truckload pickup and delivery problems," *Transport. Sci.*, vol. 38, no. 2, pp. 135–148, 2004.

[21] N. Smolic-Rocak, S. Bogdan, Z. Kovacic, and K. Petrinec, "String Algebra-based approach to dynamic routing in multi-LGV automated warehouse systems," in *Proc. CCA06*, München, Germany, 2006, on CD-ROM.

[22] S. Bogdan, M. Puncec, and Z. Kovacic, "The shortest path determination in AGV systems by using string composition," in *Proc. ICIT03*, Maribor, Slovenia, 2003, on CD-ROM.

[23] R. W. Seifert, M. G. Kay, and J. R. Wilson, "Evaluation of AGV routing strategies using hierarchical simulation," *Int. J. Production Res.*, vol. 36, pp. 1961–1976, Jul. 1998.

[24] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[25] R. A. Wysk, N. S. Yang, and S. Joshi, "Detection of deadlocks in flexible manufacturing cells," *IEEE Trans. Robot. Automat.*, vol. 7, no. 6, pp. 853–859, 1991.

[26] K. Petrinec, Z. Kovacic, and A. Marozin, "Simulator of multi-AGV robotic industrial environments," in *Proc. ICIT03*, Maribor, Slovenia, 2003, on CD-ROM.

[27] S. Bogdan, Z. Kovacic, N. Smolic-Rocak, and B. Birgmajer, "A matrix approach to an FMS control design: From virtual modelling to a practical design," *IEEE Robot. Automat. Mag.*, vol. 11, no. 4, pp. 92–109, 2004.

[28] J. K. Lim, J. M. Lim, K. Yoshimoto, K. H. Kim, and T. Takahashi, "A construction algorithm for designing guide paths of automated guided vehicle systems," *Int. J. Production Res.*, vol. 40, no. 15, pp. 3981–3994, 2002.

[29] K. H. Kim and J. M. A. Tanchoco, "Economical design of material flow paths," *Int. J. Production Res.*, vol. 31, no. 6, pp. 1387–1407, 1993.