# Distributed Multi–level Motion Planning for Autonomous Vehicles in Large Scale Industrial Environments

Lorenzo Cancemi[1], Adriano Fagiolini[2], Lucia Pallottino[1]

***Abstract*—In this paper we propose a distributed coordination algorithm for safe and efficient traffic management of heterogeneous robotic agents, moving within dynamic large scale industrial environments. The algorithm consists of a distributed resource–sharing protocol involving a re–planning strategy. Once every agent is assigned with a desired motion path, the algorithm ensures ordered traffic flows of agents, that avoid inter–robot collision and system deadlock (stalls). The algorithm allows multi–level representation of the environment, i.e. large or complex rooms may be seen as a unique resource with given capacity at convenience, which makes the approach appealing for complex industrial environments. Under a suitable condition on the maximum number of agents with respect to the capacity of the environment, we prove that the algorithm correctly allows mutual access to shared resources while avoiding deadlocks. The proposed solution requires no centralized mechanism, no shared memory or ground infrastructure support. Only a local inter–robot communication is required, i.e. every agent must communicate with a limited number of other spatially adjacent robots. We finally show the effectiveness of the proposed approach by simulations, with application to an industrial scenario.**

## I. Introduction

The adoption of automated systems for transportation and warehouse storage within the logistics of a factory has become one of major leverages by which the factory itself can be competitive in the market. As a consequence, the problem of *conflict resolution* for a system of mobile vehicles moving within the same environment and sharing resources, such as corridors or rooms, has received an extensive attention in the literature. Safety (in terms of collision avoidance), robustness, and efficiency of operations in material handling systems are critical. Moreover, stalling situation resolution and fluent navigation of the agents are to be guaranteed. Stalling situations occur when agents are unable to move from a particular configuration (i.e. deadlock) or when they are constrained to move along a finite number of paths without reaching their final destinations (i.e. live–lock).

In the existing literature, coordination of multi-vehicle systems has been exhaustively discussed, especially for Autonomous Guided Vehicles (AGVs). Most of these works may be classified in two categories, centralized or decentralized control policies. Centralized control policies ensure optimal control for agents navigation and prevent collisions and stall situations, by assigning an *a-priori* determined paths to each agent. Even thought this approach is very powerful for what it concerns robustness, safety, and collision avoidance, it is strictly constrained by the computational time requested for real-time motion of the agents, that increases with the number of the agents involved. Moreover, the faster must be executed the coordination algorithm, the more expensive must be the hardware used for computing the paths, and this is strictly connected to the cost of the hardware used to run the coordination protocol. Another disadvantage of centralized control policies is that if the central control unit fails, the whole system is out of control. Many works focus on important aspects of collision avoidance and deadlock avoidance, see e.g. [1], [2], [3]. In many cases Petri net–based control policies are adopted [4], [5], to avoid deadlocks through a path re–planning or using a master–slave approach [6]. Other strategies [7], [8] avoid deadlocks by trying to detect a cyclic–waiting situation, using graph theory for planning paths such that deadlock is a-priori avoided [9] or using a matrix–based deadlock detection algorithm [10].

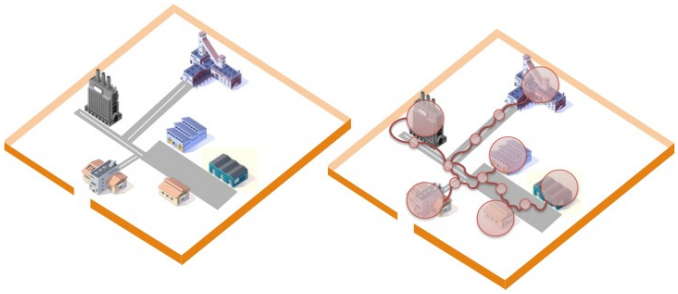

Fig. 1. Large scale industrial environment with its multi–level graph representation. Larger circles represent macro nodes that consist of a sub–level graph.

Decentralized control policies resolve the issue of high computational costs, by considering the problem as divided into two phases: during the first phase, an optimum path is assigned to each agent, by minimizing a suitable cost functional; during the second phase, every agent locally resolve any arising conflicts with other neighboring agents, according to a shared coordination policy. Among the family of decentralized solutions there is a large variety of approaches: hybrid centralized–decentralized control schemes [11], [12]; decentralized shared control policy [13]; discretization of the operative space into a finite spatial resource space [14], [15], which are "contended"

[1] Interdepart. Research Center "E. Piaggio" and Department of Information Engineering, University of Pisa, email: l.pallottino@centropiaggio.unipi.it

[2] Department of Energy, Information Engineering, and Mathematical Models, University of Palermo, email: fagiolini@unipa.it

This work was supported by E.C. contracts n. 257462 HYCON2 (Network of Excellence) and n.2577649 PLANET.

among a set of agents using a distributed mutual exclusion policy [16], [17], [18], [19]; optimal path assignment and re-planning strategies in order to recover from deadlocks [20].

The contribution of this paper w.r.t. these works is many-fold: a multi–level approach is used to handle complex and heterogeneous large scale scenario; a rule–based protocol for coordination shared among heterogeneous agents is designed to handle different goals and cost functions of different agents and that is applicable with different collision avoidance policies; a smart re–planning algorithm ensuring safe and fluid navigation of agents, avoiding deadlocks is presented; a simple and easy to implement communication system, that causes less network overheads w.r.t. message passing and that is more efficient for communication among agents. Finally, in our approach agents operate autonomously based on local communication with other agents inside a defined communication radius: no centralized mechanism, shared memory or ground support is needed.

## II. Problem definition

The problem we intend to solve is the management of a large number of autonomous vehicular robots in an industrial environment, while ensuring safety of the system (i.e. the absence of collisions) and the accomplishment of robots' tasks. Other main requirements are the scalability of the system in the number of autonomous robots, the reconfigurability of the system upon possible environmental changes, the management of heterogeneity of robots involved, see e.g. [21].

### A. Industrial environment

In this paper we consider a wide industrial environment consisting of different (macro) areas with different scopes such as warehouses, production or assembly site, open air areas, logistics sites etc. In turn, each building may then consist of several floors, corridors, rooms, narrow passages etc. Different industrial areas have also different peculiarities. Large open air areas may be less populated with respect to the production site area and autonomous vehicles may move without being constrained to follow predefined path. Indoor environments may be covered with wired or wireless sensor network that may support autonomous vehicles in their task accomplishment. In warehouse or outdoor areas the obstacles may change the environment for a consistent amount of time and an up–to–date information may guarantee a more efficient motion planning.

In general, the environmental constraints for safety purposes may change based on the area characteristic. Constraining the whole environment for the presence of congested or small regions is a penalizing approach. Heterogeneity in the environment specification and characteristics must hence be taken into account. Indeed, our goal is to have autonomous agents that are able to plan their motion and to coordinate its motion with other agents. A motion planning in a large environment may be based on a coarse representation of the environment and refinements may be used only when necessary. Such approach is typical of approximate cells decompositions methods, see e.g. [22]. Similarly to this approach we model the environment based on the concept of micro and macro areas where macro areas consist, in turn, of smaller micro and macro areas. Such a nested approach allows a streamlined management of the world representation. Micro areas adjacent to macro areas are corridors or doorways and play the role of entrance/exit gateway for the macro area.

A dedicated supervising unit may be present at each of these gateways and be responsible for guaranteeing correct access control only to authorized vehicles and to dispatch them with the necessary initialization commands and data while moving within the supervised area. The unit may authorize the vehicle access to the area upon the download of the map of the area and possibly useful services (e.g. localization service, rekeying protocol for secure communication etc, see e.g. [23]). In case of multiple gateways for a single macro area, these gateways require synchronization and can provide the entering agents with the correct information on the associated macro area. Notice that thanks to the gateways different communication protocols or collision avoidance rules may be defined in different macro area and communicated to vehicles when entering the area. For example, in outdoor environments the communication may require higher levels of security to avoid eavesdropping, see e.g. [23]. For what it concerns the collision avoidance, for example, the policy described in [24] may used in outdoor areas while another policy may be used in structured indoor environments, such as with corridors, rooms etc.. With the proposed approach is possible to adapt and reconfigure the autonomous vehicles depending on the area it intends to reach or cross (e.g. an outdoor area versus a indoor structured area).
A simple example of an industrial layout discretized in resources is reported in Fig. 1.

### B. Nested model of the industrial environment

Based on the environmental subdivision in adjacent macro and micro areas, a graph is straightforwardly obtained. Each micro node represents a micro area of the environment and is characterized by the position $(x, y)$ of a given point of the associated area (e.g. the center) in the global reference system. An edge between nodes exists if the associated regions are adjacent. A macro node represents a macro complex area and hence consists of a set of micro and macro nodes (and associated edges) and it is connected to other micro or macro nodes through micro nodes that will play the role of entrance/exit *gateways*. Each macro node is associated to a *level* represented by a graph of micro (possibly gateways) and macro (of lower levels) nodes. Once in a gateway, each agent obtains, through the network infrastructure, the graph associated to the macro node with its arcs and costs. Notice that in a graph, the cost from a macro to a (gateway) micro node will take into account the cost of crossing the whole sub–level associated to the macro node.

In the first level, named 0–level, all nodes $n$ are identified by a triple $(UID_n, 0, M)$ where $UID_n$ is a unique identifier, $0$ represents that the node belongs to the graph of the 0–level and $M = 1$ if it is a macro node, $M = 0$ otherwise. For any sub–level each node $k$ is identified by a triple $(UID_k, l_k, M)$

where $l_k = UID_h$ is the unique identifier of the macro node $h = (UID_h, l_h, 1)$ whose graph contains $k$. Also in this case, $UID_k$ is a unique identifier and $M = 1$ if it is a macro node, $M = 0$ otherwise.

To each micro and macro node $k$ a *maximum capacity* $MaxCap(k)$ is assigned a priory. The maximum capacity of a macro node can be changed only when the node its not at its maximum capacity. This may be useful when, for several reasons, we are interested in limiting access to a plant zone to fewer agents. The capacity is communicated to the agents by the gateways together with the graph. A micro node capacity larger than one will allow more than one agent to have access to the same region associated to the node. In this case, the collision avoidance protocol, associated to the macro node the micro node belongs to, will be used to manage the potential conflict as explained in Section IV-B.

## III. The heterogeneous autonomous agents

One of the main contribution of this paper is that the protocol is designed to handle heterogenous agents. Agents may differ in their kinematic constraints (holonomy, maximum speed, limited turning radius, etc.), in their mission specification (crossing given regions, in reaching a final destination, patrolling among different points of the environment), in their personal cost functions (minimize time to complete the mission, minimize traveled distance, minimize energy consumption, etc.). In particular, each agents plans its own trajectory based on its personal costs through an appropriate algorithm. For example, a simple Dijkstra algorithm can be used to determine a minimum path on the given graph toward the final destination, a salesman travel problem can be solved in case of agents that must patrol different areas in the environments or an $A^*$ algorithm can be used to minimize the time to reach the final destination, see e.g. [22] for details on such algorithms.

### A. Agent internal state

Depending on the personal costs and goals, each agent determines the path on the graph that it intends to use. Furthermore, each agent $i$ has the possibility to set its own speed $v_i$ in a given known interval $[0, v_{max}]$ especially for collision avoidance purposes.

Each agent $i$ is characterized by its own $UID_i$ unique identifier and a priority $P_i$ that is preassigned to the agent based on the task criticalities or time constraints and that can be changed through the gateways if necessary. For example, in case of urgent requirement of a certain type of material, the vehicle handling the good can be set as a higher priority agent. The agent has also an internal clock $t$ that is synchronized with other agents clocks.

To manage the presence of different levels, i.e. macro nodes in macro nodes, each agent must keep in memory the node it is occupying and the final desired node for each involved level. An identifier of the currently occupying level is also useful to manage the passages between levels. We denote the micro node that agent $i$ intend to reach in the plan as $FN_i$.

### B. Inter agents communication

Each agent sends and receives a communication packet that is a quintuple $CP_i = (UID_i, P_i, CN_i, PN_i, SN_i)$ with the following information: its own unique identifier $UID_i$, the current priority $P_i$, the node currently occupied $CN_i$, the first node in the desired path (that follows the currently occupied node) named *primary node* $PN_i$, a *secondary* desired nodes $SN_i$ different from $PN_i$ whose role and determination will be cleared in the following.

*Definition 1:* Agents $i$ and $j$ are *neighbours* if $\{CN_i, PN_i, SN_i\} \cap \{CN_j, PN_j, SN_j\} \neq \emptyset$. We will denote with $\mathcal{N}_i$ the set of neighbours of agent $i$. Node $k \in \{CN_i, PN_i, SN_i\} \cap \{CN_j, PN_j, SN_j\}$ are *shared nodes* between agents $i$ and $j$.

Based on the current capacity of node $PN_i$ and the communication packets received from neighbours, each agent $i$ evaluate the access to node $PN_i$ as described in the following. Notice that the communication is required among agents that are interested in the same nodes and hence are close to each other.

### C. Access to a micro node

To access a micro node, each agent $i$ checks the communication package received by the neighbours. Based on those packages each agent may reconstruct the following information:

- *Current capacity* of node $CurCap(N_i) = \#\{j \in \mathcal{N}_i | CN_j = PN_i\}$[1].
- Current *number of requests* of access to node $Req(N_i) = \#\{j \in \mathcal{N}_i | PN_j = PN_i\} + \#\{j \in \mathcal{N}_i | SN_j = PN_i\}$.

If agent $i$ has a next node $PN_i$ such that $CurCap(N_i) + Req(N_i) \leq MaxCap(N_i)$, i.e. the number of requesting agents plus the number of agents currently on the node does not exceed the maximum capacity, the agent may have access to the node. Otherwise, a priority–based ranking is determined. On a FIFO–like (First–In–First–Out) flavour, the ranking is first based on the time of request. The oldest requests will be processed based on the priority value. Furthermore, in case of equal priority the order is decided based on the $UID$'s values. With this FIFO–like approach we will be able to ensure that agents will reach $PN_i$ (or $SN_i$) in a finite number of steps. Moreover, we penalize agents to request the same node after few time steps.

All agents requesting access to $n$ with ranking value $rank_i(n) \leq MaxCap(n) - CurCap(n)$ may have immediate access to the node. Hence, the effective ranking (in the following, for simplicity referred to as ranking) of other agents is $ER_i(n) = rank_i(n) - MaxCap(n) + CurCap(n)$. Notice that each agent can easily compute the effective ranking position of all its neighbours.

Each agent will require access to the $PN_i$ only when it is close enough to the region associated to $PN_i$ in order to have access to it and release node $CN_i$ shortly after. On the other hand, the agent must also be able to stop its motion if needed before entering $PN_i$. This strictly depends on the

[1]We denote with $\#$ the cardinality of the set.

kinematics and dynamics model of the agent and hence each agent decides the correct time to request the next node based on its model, the dimension of the region associated to $CN_i$, its position and current speed.

### D. Planning alternative paths

Based on the available graph, each agent $i$ plans its most convenient trajectory toward the desired node $FN_i$, or the desired exit/entrance gateway of current level. Once such *primary path* has been found, the first node of the path is removed by agents' graph together with all the incoming arcs. A new optimal path planning is run and an alternative *secondary path* is found.

Such alternative path is introduced to try to reduce the waiting time spent by agents to have access to node. This occurs when the agent's ranking implies a foreseen long waiting time before agent will be able to have access to the first node of the primary path. Moreover, it may be possible to authorize the possibility to choose between primary and secondary paths to a limited number of agents or only to a specified subset of specialized agents based on their ability in accomplishing critical tasks. This is done to manage high traffic plant sectors and to avoid that all agents, with a common primary and secondary next node, choose the secondary one. For simplicity, in our implementation we chose to authorize only the first half of ranked agents to use their secondary paths.

Whenever authorized, agent $i$ requests access to both $PN_i$ and $SN_i$. Notice that also for the secondary node $k = SN_i$ a ranking is required in case $CurCap(SN_i) + Req(SN_i) > MaxCap(SN_i)$. In this case, agents with $SN_j = k$ may be ranked based on a scaled priority to possibly penalize the choice of the secondary path with respect to the primary one. The *effective total cost* $ETC_i(PN_i)$ ($ETC_i(SN_i)$) of each agent $i$ to reach the final desired node through the primary (secondary) path is hence a function of the cost of the primary (secondary) path and the position $ER_i(PN_i)$ ($ER_i(SN_i)$) in the ranking list associated to $PN_i$ ($SN_i$). Based on that total costs, the agent will chose which path to follow between primary and secondary. Indeed, in case of a ranking on $PN_i$ and the possibility to access to $SN_i$, the secondary path can have such a larger cost with respect to the primary one that $ETC_i(PN_i) < ETC_i(SN_i)$ and the agent prefers to wait for $PN_i$ while avoiding to have access to $SN_i$.

Moreover, in case of long waiting an agent $i$ may prefer to try another path that does not cross $PN_i$ or $SN_i$. This can be handled updating at each time step the cost $c(CN_i, n)$ of the arc from $CN_i$ to $n = PN_i$ or $n = SN_i$ based on the ranking position as follows

$$c(CN_i, n) := c(CN_i, n) + d_i(ER_i(n) + 1) \qquad (1)$$

where $d_i$ is a parameter representing the inverse of the agent waiting tolerance. Higher is the value $d_i$ more the path is penalized. When a long waiting is foreseen another path may hence been found by the agent's planning algorithm.

We are now able to provide the final state machine that describes the proposed coordination protocol. For simplicity of description we suppose that each macro node consists only of micro nodes and hence two levels are available in the plant model. In this case, for each agent $i$ we consider the vector $l_i = [c_0, f_0, c_1, f_1]$ where $c_0$ and $c_1$ are currently occupied nodes, $f_0$ and $f_1$ are the final nodes in the 0–level and the current level respectively.

## IV. The coordination protocol

A well know approach to model the agent motion, based on environmental information and intra–agent information exchange, is a finite state machine. Each state of the machine is associated to the agent state while arcs represent events occurrence that imply a modification of the agent state. More formally consider three agent states:

1) **Waiting (W)** can not have access to neither $PN_i$ nor $SN_i$.
2) **Requesting (R)** the agent requests access to $PN_i$ and possibly to $SN_i$.
3) **Moving (M)** the agent has gained the access to $PN_i$ or $SN_i$ and is moving to reach the new node, or the agent is moving in $CN_i$.

For space limitations is is not possible to report all the switching conditions between agent states as logical proposition of variables defined above. However, the state flow for the three states is reported in Fig. 2 and Fig. 3. Notice that each state flow is computed in a time step. In the $W$ and $M$ state flows, the speed and the position in the motion space are modified accordingly to the motion protocol associated to the node level. In case of the $W$ state, this is done taking into account the presence of other agents and the fact that no other node has been authorized to the agent yet.

As already mentioned, agent will require the access to $PN$ and $SN$ based on the $CN$ physical dimension and on the time foreseen to reach $PN$. Finally, notice that $CN$ is updated and the resource released when the agent will not occupy the physical region associated to it while moving toward $PN$ or $SN$.

### A. Deadlock free properties

For the sake of simplicity in the exposition of this section, we suppose that when an agent enters its desired final node it is excluded from the coordination protocol and not taken into account anymore in the system evolution.

A basic hypothesis is that the collision avoidance policy in each node of capacity larger than 1 and the dimension of the region are such that the agent is able to exit the node in finite time. Those aspect will be further analyzed in section IV-B.

Notice that the state flow is such that for agents in states $W$ and $R$ after a time step the agent switches to a different state, $R$ and $W$ or $M$ respectively. On the other hand, agents in state $M$ may stay on the state $M$ for more than a time step. However, the following holds.

*Proposition 1:* Given an agent in state $M$, after a finite number of steps the guards toward state $R$ are verified and the agent changes its state.

*Proof:* An agent in state $M$ may be moving within $CN$ (i.e. $CN = n$) and hence, based on the hypothesis on the collision avoidance policy, after a finite time it is able to

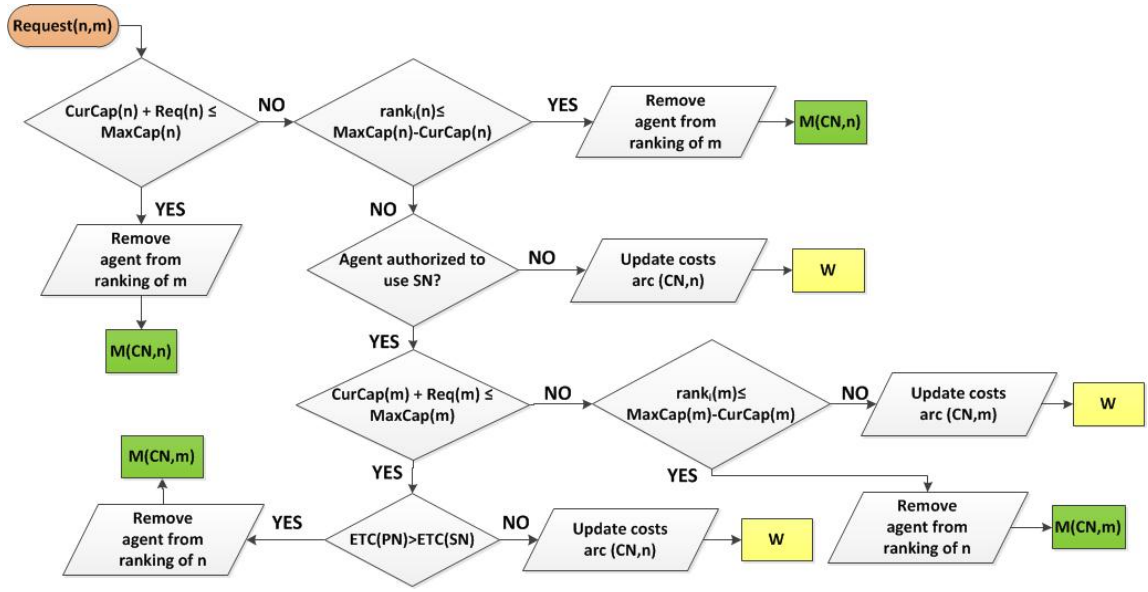


Fig. 2. State flow for agent $i$ with $PN_i = n$ and $SC_i = m$ when requesting access to $n$ and $m$.

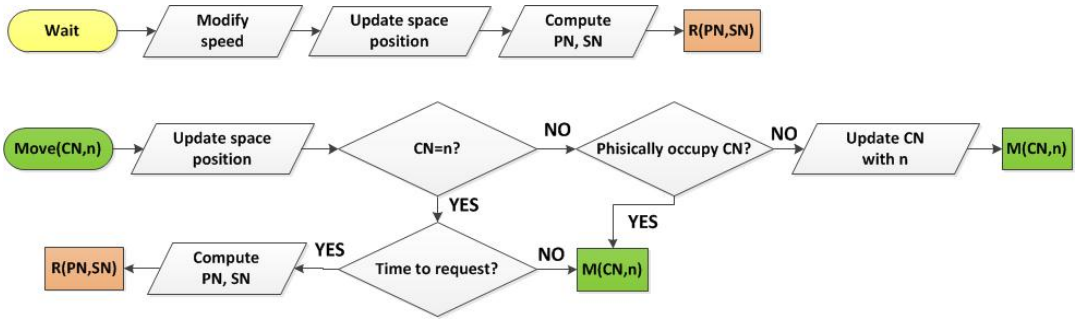


Fig. 3. State flow for agent $i$ with $PN_i = n$ and $SC_i = m$ when waiting for having access to a node or moving toward a new node.

request access to $PN$ and $SN$. Hence a transition to state $R$ occurs. Otherwise, the agent is moving from $CN$ toward another node ($PN$ or $SN$, i.e. $CN \neq n$). For hypothesis on the collision avoidance policy, agent will then reach $n$ and release $CN$ in a finite time, i.e. $CN$ is updated to $n$. Concluding, from $M$ a switch to $R$ occurs in finite time and hence the thesis. ■

We now introduce a condition on the total number of agents in the environment to guarantee that no deadlock occurs. A deadlock is a condition in which no agent will ever change its current node $CN$ and the system is in a stall configuration.

Condition 1: The number $N$ of agents in the environment is smaller than the sum of the capacity of macro and micro nodes of the 0-level. In other words, given $L_0 = \{j | j = (k,0,1) \wedge j = (k,0,0), k \in \mathbb{Z}\}$

$$N < \sum_{j \in L_0} MaxCap(j) \tag{2}$$

*Proposition 2:* If condition 1, i.e. (2) holds, there always exists at least an agent that will reach its desired next node $PN$ or $SN$ in a finite number of time steps.

*Proof:* Condition (2) ensures that at any time $T$ there exists at least a node $j$ with $CurCap(j) < MaxCap(j)$. Hence, there exists an agent $i$ with $PN_i = j$ or $SN_i = j$ that obtains the access to the new node in a finite time. Indeed, even if no agent is interested in $j$ at time $T$, after a finite number of steps the costs of arcs of agents requiring other occupied resources will be such there exists an agent $i$ with $PN_i = j$ or $SN_i = j$. Moreover, after another finite number of steps, see proof of Proposition 1, the node $k = CN_i$ is released and the same reasoning can be applied to agents requesting $k$ that is not at its maximum capacity. ■

From above proposition we may conclude that no deadlock occurs with the proposed approach. Obviously we are not able to guarantee livelock occurrence, i.e. agents move on the graph without reaching the final destination. However, in the conducted simulations such livelock occurs in small ad-hoc crowded environments with a small total capacity with respect to agents.

### B. Collision Avoidance

To manage the heterogeneity of a large scale industrial environment, every macro node is assigned with a possibly different collision avoidance strategy, being developed to efficiently handle the operating conditions of the node itself. While moving within all the micro nodes comprised in a higher level macro node, agents are supposed to adopt the associated

macro–node collision avoidance policy. Such protocol must ensure that every vehicle can reach in finite time their next nodes without colliding agains other vehicles.

The definition of each macro–node collision avoidance protocol, which is out of the scope of the paper, depends on many aspects, including the nature of the environment represented by the node, the type of coexisting vehicles, some special safety super requirements due to e.g. coexistence with humans in specific corridors or rooms. Furthermore, to correctly handle the motion of vehicle from one node to another, while allowing to use possibly different motion protocols, we introduce the notion of a macro node *capacity* as follows.

A suboptimal yet conservative choice is to define the capacity of a macro node based on the maximum capacity of the correspondent region's dimension. Suppose e.g. that heterogeneous agents are able to stop (otherwise we may use the idea of reserved region proposed in [24]) and have a maximum safety disk (representing e.g. the dimension of the robot) of radius $R$. In worst case we must need to stop agents and move them only one at a time (in FIFO order) toward the desired node. The maximum number of agents that can be handled with this very conservative approach depends on $R$ and the region under consideration and can be set as the maximum capacity of the node associated to the region.

Note that an agent of dimension $R$ moving in a region may overlap part of another adjacent region. Based on a similar approach to the one proposed in [25], two adjacent regions are characterized by the *overlapping region*, in which the agent's safety disk partially overlaps both regions (represented by the grey region in Fig.4). The dimension of the region used to determine the maximum capacity is at net of the overlapping regions and this is the region in which agent must move when waiting to access to a new node. When an agent $i$ in a node $n$ has gained access to node $m$ starts moving toward the region associated to $m$. The capacity of node $n$ is decreased only when the agent lays in the region associated to $m$ and its safety disk does not overlap with the region associated to $n$, see Fig.4. It is true that, with this approach, the node that is currently occupied can be represented by a couple of nodes but this is omitted in the state representation of agents to simplify the protocol explanation.

## V. Simulation Results

For simplicity in the simulation reported in this section we considered the case of nodes of maximum capacity equal to 1. The environment considered in this set of simulation is reported in Fig. 5 where a node is associated to each corridor, each room and door. Nodes associated to regions in which a dashed map is represented are macro nodes. The graph associated to the environment considered is reported in Fig. 7 with its arc costs that take into account the length of corridors or dimension of rooms.

In the simulation reported in Fig. 6, two robots want to reach the point on the other side of the plan with a crossing point corresponding to the node 13 in the center of the environment. The robot navigating from East to West (from 11 to 15) has higher priority and the robot navigating from South to North (from 3 to 23) waits until the resource is made available.

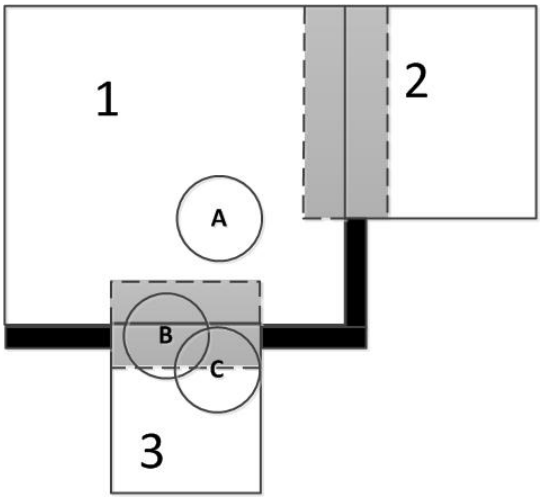


Fig. 4. Consider agent in position $A$ its current node $CN$ is node 1, when gained access to 3 and moving toward it, e.g. in position $B$ the agent is occupying both 1 and 3 in terms of capacity. When an agent reaches position $C$ it releases node 1.

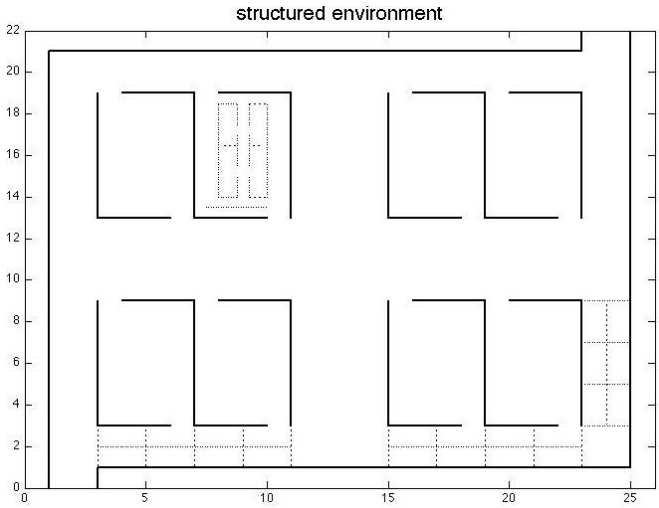


Fig. 5. Environment representation, dashed areas are macro node

The second simulation, reported in Fig. 8, is similar to the first one. However, in this case a fixed robot has been considered in node 13 while a robot in 3 would like to reach node 23 (from South to North). When the algorithm starts, the non–fixed robot chooses its best path toward the goal and this crosses node 13 (the center of the plant). The robot starts moving toward 13 but, since the node is at its maximum capacity, the robot can not have access to it. After a waiting time the increased arcs cost, as in (1), allows the robot to find another path that eventually bring it toward its destination, i.e. node 23 without crossing node 13.

In the last simulation we show an agent moving form 0-level to 1-level. In Fig. 9 on the left the evolution in the 0-level is reported while on the right the evolution in the macro node 20 1-level is represented. Circled points in the top left image correspond to robot fixed on nodes 11 and 40 that will force the other robots to find alternative paths.

More complex simulations have been conducted with up to

Fig. 6. Two robots in a crossing scenario the lower priority robot waits until the resource is made available. In this scenario for the lower priority robot is not convenient to look for another longer path.

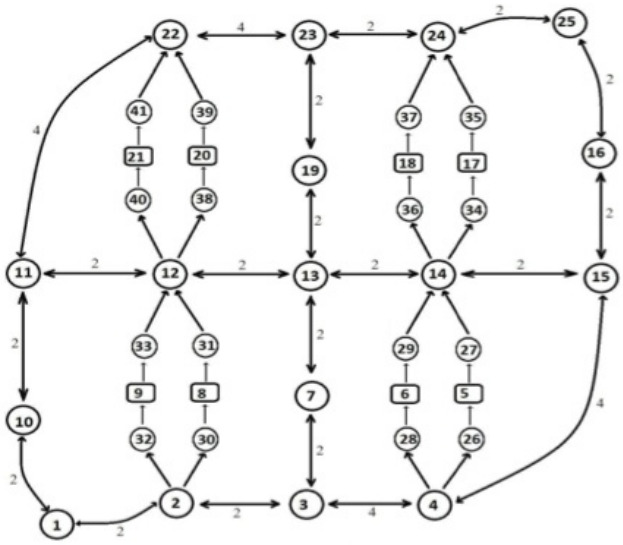


Fig. 7. Graph associated to the environment of Fig. 5

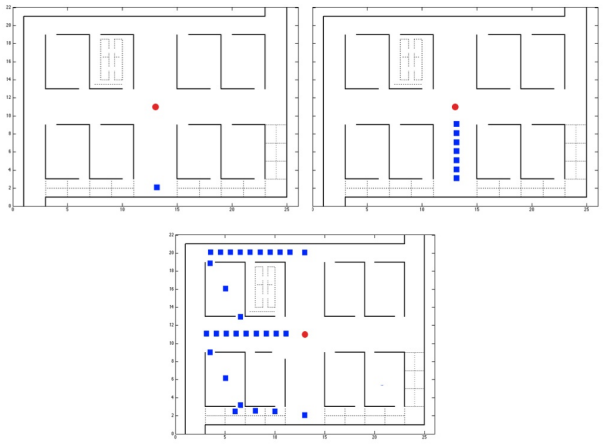

Fig. 8. Evolution of an agent that encounters another agent blocked on a node and after waiting to access that node decides to go toward the goal by replanning.

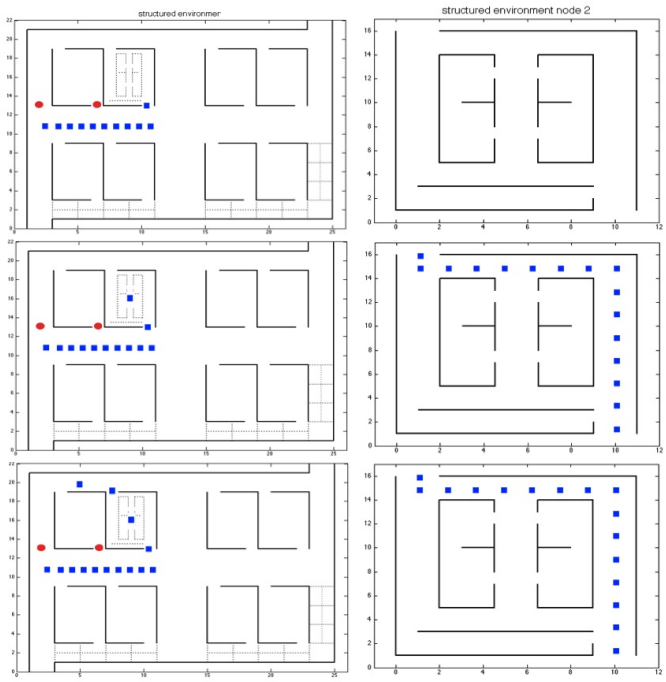


Fig. 9. Evolution of an agent through macro node 20, left: evolution in the 0-level, right: evolution in the 1-level.

10 agents and with different node capacities and arc costs. A simulation with 5 agents is reported in Fig. 10 where node 11 has capacity 2 and indeed two agents cross it simultaneously. Moreover such agents have different speed and exit the node in different steps.

## VI. Conclusions and Future work

In this paper a decentralized protocol for multi–robot coordination has been proposed. The main characteristic of the protocol is that it can be adapted to large scale environments (structured and unstructured, e.g. open space environment with few obstacles) to a non fixed number of heterogeneous robots with different kinematic or dynamic constraints. Moreover, the protocol is able to handle different collision avoidance policies to be used in different part of the environment. More simulation will be conducted in more complex environments taking into account also different policies. A probabilistic approach will be used to determine under which conditions all the vehicles eventually reach their final destination.

## References

[1] M. P. Fanti. *Event–based controller to avoid deadlock and collisions in zone–control AGVS*. Int. J. of Production Res., vol. 40, no. 6, pp. 1453–1478, 2002.

[2] S.A. Reveliotis. *Conflict resolution in AGV systems*. IIE Trans., vol. 32(7), pp. 647–659, 2000.

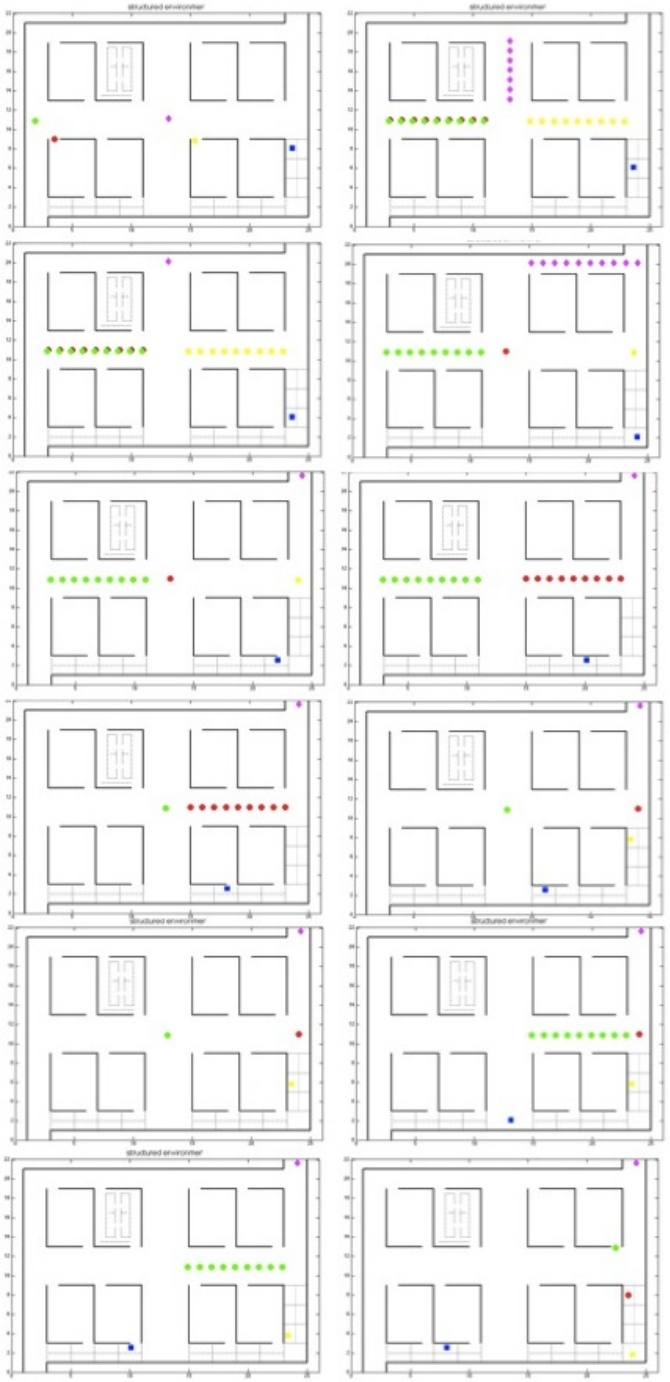

Fig. 10. Simulation with 5 agents.


[3] S.A. Reveliotis and P.M. Ferreira. *Deadlock avoidance policies for automated manufacturing cells*. IEEE Transactions on Robotics and Automation, vol.12, n.6, pp. 845–857, 2002.

[4] N. Wu and M. Zhou. *Shortest routing of bidirectional automated guided vehicles avoiding deadlock and blocking*. IEEE/ASME Transactions on Mechatronics, vol. 12, no. 1, pp. 63–72, Feb. 2007.

[5] M. Fanti. *A deadlock avoidance strategy for agv systems modelled by coloured petri nets*. Sixth International Workshop on Discrete Event Systems, pp. 61–66, 2002.

[6] S. Yuta and S. Premvuti. *Coordinating autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots*. IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 1566–1574, Jul 1992.

[7] R.L. Moorthy, W.Hock-Guan, Ng Wing-Cheong and T. Chung-Piaw. *Cyclic deadlock prediction and avoidance for zone–controlled AGV system*. International journal of Production Economics 83 (2003) 309–324.

[8] M. Singhal. *Deadlock Detection in Distributed Systems*. Computer Vol. 22, Numb. 11, Pages 37-48, 1989.

[9] J.W. Yoo, E. S. Sim, C. Cao and J. W. Park. *An algorithm for deadlock avoidance in an AGV system*. Int. J. Adv. Manuf. Technology 26: 659-668, 2005.

[10] M. Lehmann, M. Grunow and H. O. Gunther. *Deadlock handling for real–time control of AGVs at automated container terminals*. OR Spectrum, Springer, 28:631-657, 2006.

[11] S. LaValle and S. Hutchinson. *Path selection and coordination for multiple robots via nash equilibria*. IEEE International Conference on Robotics and Automation, vol. 3, pp. 1847–1852, May 1994.

[12] S. LaValle and S. Hutchinson. *Optimal motion planning for multiple robots having independent goals*. IEEE Transactions on Robotics and Automation, vol. 14, n. 6, pp. 912–925, 1998.

[13] S. Kato, S. Nishiyama and J. Takeno. *Coordinating mobile robots by applying traffic rules*. IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 1535–1541, Jul 1992.

[14] J. Wang. *Operating Primitives Supporting Traffic Regulation and Control of Mobile Robots under Distributed Robotic Systems*. IEEE International Conference on Robotics and Automation, vol. 2, pp. 1613–1618, 1995.

[15] J.Wang and S.Premvuti. *Distributed Traffic Regulation and Control for Multiple Autonomous Mobile Robots Operating in Discrete Space*. IEEE International Conference on Robotics and Automation 1995, pp. 1619-1624.

[16] J. Wang. *DRS Primitives based on Distributed Mutual Exclusion*. IEEE/RSJ International Conference on Intelligent Robots and Systems, June 26-30, Yokohama, Japan, pp. 1085-1090, 1993.

[17] Leslie Lamport. *The Mutual Exclusion Problem: Part I - A theory of interprocess communication*. Journal of the ACM, Vol. 33 n. 2, pp. 313–326, April 1986

[18] Leslie Lamport *The Mutual Exclusion Problem: Part II - Statement and solutions*. Journal of the ACM, Vol. 33 n. 2, pp. 327–348, April 1986.

[19] S. Manca, A. Fagiolini, L. Pallottino. *Decentralized Coordination System for Multiple AGVs in a Structured Environment*. 18th World Congress of the International Federation of Automatic Control (IFAC 2011), vol.18, n. 1, pp. 6005–6010, 2011.

[20] N. Wu and M. C. Zhou. *AGV routing for conflict resolution in AGV systems* IEEE International Conference on Robotics and Automation, Vol. 1, pp. 1428–1433, Taipei-Taiwan, Sept. 14-19, 2003.

[21] A. Bicchi, A. Fagiolini, L. Pallottino. Towards a Society of Robots IEEE Robotics & Automation Magazine. doi: 10.1109/MRA.2010.938839, vol.17, no.4, pp.26,36, Dec. 2010.

[22] S. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[23] A. Bicchi, A. Danesi, G. Dini, S. La Porta, L. Pallottino, I. M Savino, R. Schiavi. Heterogeneous wireless multirobot system IEEE Robotics & Automation Magazine, vol.15 n.1, pp.62-70, 2008.

[24] L. Pallottino, V.G. Scordio, A. Bicchi and E. Frazzoli. *Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems*. IEEE Transactions on Robotics, vol. 23, n. 6, pp. 1170–1183, 2007, doi 10.1109/TRO.2007.909810.

[25] E. Roszkowska and S. Reveliotis. A Distributed Protocol for Motion Coordination in Free-Range Vehicular Systems- Automatica, to appear.