A decentralized control strategy for the coordination of AGV systems

Version with DOI: https://doi.org/10.1016/j.conengprac.2017.10.001

A Decentralized Control Strategy for the Coordination of AGV Systems

Maria Pia Fanti[a], Agostino M. Mangini[a], Giovanni Pedroncelli[a], Walter Ukovich[b]

[a] *Polytechnic of Bari, Italy*
[b] *University of Trieste, Italy*

Abstract

This paper proposes a decentralized control strategy to assign destinations to Autonomous Guided Vehicles (AGV) and let them autonomously coordinate their paths to avoid deadlock and collisions. The AGVs move inside a zone-controlled guidepath network and solve a distributed assignment problem by performing a discrete consensus algorithm in order to locally minimize the global cost for reaching the destination zone. Successively, the AGVs coordinate their paths by a decentralized coordination protocol that is based on a zone-controlled approach which guarantees the avoidance of deadlock and collisions. Moreover, the proposed decentralized strategy is implemented by means of a simulation software, which allows the user to define the guidepath network and runs the two algorithms showing the behavior of the AGVs starting from their initial position to the destination.

Keywords: Autonomous Guided Vehicles, Consensus Algorithms, Decentralized Control, Deadlock Avoidance

1. Introduction

The problem of coordinating and controlling Autonomous Guided Vehicle (AGV) systems has received a large attention in the last years. First, it is important to provide some basic definitions concerning Guided Path Networks (GPNs): they are systems where the paths are enforced through a control software. In these systems, it is possible to assign a set of trips to each vehicle between various locations: if the vehicles follow a pre-assigned route, the assigned path is static, or, if the path is determined in real time,

the path is dynamic [1]. Moreover, in some cases there is a docking station in the network where the vehicles return after concluding their path and wait for the next mission (*open system*). On the other hand, in more complex systems the vehicles remain in the guided-path during the idling period (*closed system*).

In the related literature, the AGVs are coordinated by using either a centralized or decentralized control architecture. In centralized systems, a central unit (i.e. a controller) performs either some or all of the following jobs: destination assignment, path planning and coordination of vehicles in order to avoid collisions and deadlock. The central unit can send control actions to perform to all the agents in the network, also having a complete knowledge of the whole system [2–5].

Furthermore, in order to avoid the physical collision of the vehicles, the entire GPN is partitioned into a number of segments, and only one vehicle is allowed in any segment at any point in time; in the relevant terminology, these segments are known as zones and the resulting guidepath-based traffic system is said to be zone-controlled.

In particular, [2] and [3] propose policies and control strategies to avoid deadlock and collisions in zone-controlled networks with AGVs modeled by Coloured Petri Time nets. The authors in [4] deal with Resource Allocation Systems by developing a liveness-enforcing supervisor in order to avoid forbidden states; on the other hand, [5] presents an efficient AGV deadlock prediction and avoidance algorithm to be used in container port operations.

In centralized approaches, the control paradigm is robust but suffers from complexity which increases with the number of the agents resulting in too conservative policies.

On the other hand, decentralized strategies typically manage the AGVs in two phases: i) the definition of the optimum path for each vehicle by minimization of a cost function; ii) a distributed coordination of each agent with its neighbors in order to avoid conflict and deadlocks. However, the path assignment is performed by a centralized approach (i.e., a supervisor), that assigns the tasks to the AGVs. Moreover, the applied deadlock strategies resolve the deadlock by applying deadlock recovery procedures and do not prevent these blocking scenarios.

In particular, in the algorithm of [6], vehicles autonomously plan and execute their motions, detect and solve collisions and deadlock with other vehicles in communication range by conducting decision making. Nevertheless, this approach does not deal with the distributed assignment of destina-

tions. In [7] the decentralized algorithm provides strategies to detect but not prevent local deadlocks and livelocks. Authors in [8] present a distributed approach which solves the task allocation, path planning and feedback control of robots, but does not deal with deadlock situations. Moreover, in [9] starting from an initially deadlock-free condition, blockings are avoided by replanning strategies. In addition, in [10] the collision avoidance is implemented in a decentralized way and replanning strategies are proposed to recover from deadlocks. The works [11, 12] propose a two-layers control architecture to manage the traffic with hybrid path planning (partly centralized, partly decentralized) and a fully decentralized coordination. In addition, paper [13] uses a multi-layered hierarchical roadmap constructed using sonar data. Other methods include resource allocation systems [14] and rule-based approaches [15].

This paper considers an industrial zone-controlled GPN: the problem consists in assigning a set of tasks to be executed in some zones to a set of AGVs and to control the moves of the vehicles in order to avoid deadlocks and collisions. Such assignment problem can be formulated as an Integer Linear Programming (ILP) problem that is NP-hard and requires large computational resources when the system dimension increases. Hence, some distributed approaches are proposed in literature, which provide optimal or good suboptimal solutions [16], [17], in reasonable time by exploiting the new challenges of the Information and Communication Technology tools.

In this paper, the problem of assigning tasks to AGVs, and routing them, is solved with distributed algorithms.

First, the "intelligent" AGVs assign themselves the destinations (tasks) to be reached by means of a distributed discrete consensus algorithm. Such an assignment is autonomously performed by the vehicles that iteratively solve some Local Integer Linear Programming (L-ILP) problems [16] with a set of vehicles with which they can communicate. The global objective is minimizing the total completion time of all the tasks in the systems. The main advantage of the decentralized assignment approach is that the task assignment problem, which is an NP complete problem, can be solved in reasonable time among a subset of AGVs.

Second, after the completion of the assignment phase, the vehicles move across the guidepath network to reach their destination using a decentralized zone-control based coordination algorithm. We prove that the proposed coordination protocol guarantees collision and deadlock avoidance, if operating in a suitable class of network topology.

A first version of the proposed algorithm is presented in [18] and this paper improves the decentralized control strategy as follows: i) the formalism and notations are revised and improved; ii) the liveness of the system guaranteed by the decentralized deadlock avoidance policy is proved; iii) the assignment phase is simplified by considering a different initial assignment (i.e., a feasible solution of the defined L-ILP): the consequence is that it is not necessary to define a different programming problem to be applied when the L-ILP is not feasible, like in [18]; iv) the coordination algorithm is modified by reducing the communications among the agents and clarifying in more details their actions; v) a simulation software is implemented in order to allow users to describe a GPN and the behavior of the decentralized controlled AGVs.

Finally, the proposed algorithms are applied to GPN inspired by a company located in the Southern Italy. The company designs AGV systems and the relative control software: a decentralized management strategy is requested in order to deal with large fleets of AGVs. Here, with the aim of clearly explaining the proposed control strategy, the management of a simplified network topology is analyzed.

The rest of the paper is organized as follows. Section 2 states the problem and describes the system model also characterizing the liveness of the system. Sections 3 and 4 present the assignment algorithm and the coordination strategy, respectively. Finally, Section 5 discusses an application of the algorithms by a simulation software and Section 6 draws the conclusions.

2. System Description

2.1. Problem Specification

We consider an industrial Guided Path Network (GPN) composed of Z zones and a set of N AGVs $A = \{n | n = 1, 2, \ldots, N\}$ travelling in the system: only one AGV is allowed in each zone, which is exclusively allocated to it. The set of AGVs has to complete a set of $K \leq N$ tasks $U = \{j | j = 1, 2, \ldots, K\}$ that have to be assigned to the AGVs. The tasks are considered as zones to be reached by the AGVs starting from their initial position. Note that $n \in A$ and $j \in U$ can be considered the id numbers to identify respectively each AGV and task.

The vehicles have the following information about the system: i) the GPN layout, i.e., the network topology and the location of the Z zones; ii) the set of tasks to be completed and the relative position in the GPN.

The assignment of the tasks in U is performed autonomously by the vehicles in A and the assigned destinations have to be reached in a distributed manner, avoiding collisions and deadlocks.

Therefore, we decompose this problem into two sub-problems to be solved by two distributed algorithms executed in sequence by the vehicles:

- *the decentralized assignment problem* that the AGVs solve by Algorithm 1; after the algorithm execution each AGV knows its final destination task (zone) and the route to reach it;
- *the decentralized coordination problem* that is solved by Algorithm 2 performed by the AGVs in order to reach the assigned destinations, also guaranteeing collision and deadlock avoidance.

2.2. System Model

We describe the system dynamics inspired by the Discrete Event model presented in [1]: each event is described by the advance of vehicles in A from a zone to the next one. Moreover, the system state s is represented by a labeled partially directed graph $G(s)$: the graph topology is determined by the layout and the zoning of the GPN. The labeling function l is defined on the set of edges and it labels an edge with the identification number of the vehicle lying on the corresponding zone.

A partially directed graph is described by a triple $G = (\mathcal{V}, \mathcal{Z}, D)$ where:

- $\mathcal{V}$ is the set of the nodes representing vertices to cross in order to go from a zone to another one;
- $\mathcal{Z}$ is the set of the graph edges with $\mathcal{Z} \in (\mathcal{V} \times \mathcal{V})$ representing the zones of the GPN: edge $z_{x,y}$ between the two vertices v_x and v_y is directed if an AGV is moving from node v_x to node v_y or viceversa while it is undirected if there is no vehicle on that edge;
- D is a partial function on $\mathcal{Z}$ such that $D(z_{x,y}) = v_y$ if there is an AGV on $z_{x,y}$ moving towards v_y. If z is undirected then $D(z)$ is not defined.

Moreover, given the graph G it is possible to state the following definitions:

- A path $\pi = \{v_0, z_{0,1}, v_1, \ldots, v_{n-1}, z_{n-1,n}, v_n\}$ with $n \geq 0$ is a sequence of vertices connected by edges such that the generic edge $z_{x,y}$ points from a generic vertex to the following of the sequence. A path is simple if all the vertices are distinct, except, possibly, for the first and the last.

- A cycle c is a simple path such that $n > 0$ and $v_0 = v_n$.
- A joint between two cycles c and c' is a simple path that is a sub-path of both c and c'.
- A bridge is an edge whose removal disconnects the remaining subgraph. Therefore a bridge is an edge not contained in any cycle.
- A pass p between two cycles c and c' is a simple path such that its first vertex lies on c, its last vertex on c', and all its edges are undirected and are not components of either c or c'. We denote with $P = \{p_1, \ldots, p_m\}$ the set of all the passes in the network.
- A chain in G is a subgraph defined by a sequence $ch =< c_1, \pi_2, c_2, \ldots, \pi_n, c_n >, n \geq 1$ such that c_i are cycles and each path π_i is a joint or a pass that connects two or more cycles.

Example 1. Figure 1 shows an example of graph belonging to the class of the proposed GPN layouts. Figure 1(a) shows an example of labelled graph $G(s)$ describing the state s: there is a pass of two-bridges length between vertices v_3, v_4 and v_5, i.e. $p_1 = \{v_3, z_{3,4}, v_4, z_{4,5}, v_5\}$. Moreover, AGV 1 lies in the zone $z_{2,3}$, while AGV 2 is in edge $z_{3,9}$: therefore $z_{2,3}$ and $z_{3,9}$ are directed edges. Since the bridges $z_{3,4}$ and $z_{4,5}$ form a pass, the graph constitutes a chain.

Moreover, Fig. 1(b) describes the same network at a different state: AGV 3 is on edge $z_{4,5}$ and therefore this is now a directed edge; in this configuration there is not a pass between v_3, v_4 and v_5 anymore. There are two chains ch_1 and ch_2 and both contain paths that are either cycles or joints. When an event occurs the new state s' is described by the updated $G(s')$: e.g., if AGV 3 occupies edge $z_{5,6}$ then the edge $z_{4,5}$ becomes a bridge again and $z_{3,4}$ and $z_{4,5}$ constitute a pass.

2.3. Characterization of the System Liveness

In this section we prove in the framework of [1] that graphs with a topology belonging to a specific class are deadlock-free.

Definition 2.1. *In the guided path network, state s is chained iff $G(s)$ is chained.*

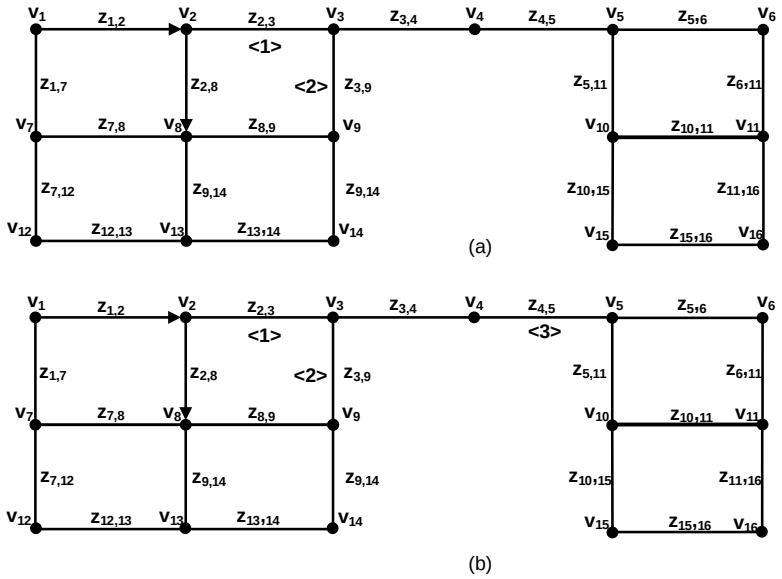

Figure 1: Example of a partially directed graph

Definition 2.2. *In a guided path network, state s is semi-chained if it is generated from chained states by transferring a single vehicle between two cycles c and c' connected by a path in some given chain.*

More clearly, the definition of the class of semi-chained states characterizes the state from which it is possible reach a chained state through an event sequence that concerns the advancement of a single vehicle in a network simple path.

Now, in order to characterize the liveness of the system, we recall the results proved in [1].

Definition 2.3. *The guided path network system is live if every vehicle $i \in A$ maintains its capability to reach any zone of the guidepath network throughout a presumably infinite horizon of the system operation.*

Let $R(s)$ be the set of states that can be reached from a given state s through an event sequence. Moreover, the following theorem proved in [1] characterizes the state liveness of a guided path network of Z zones and b bridges.

Theorem 2.4. *In a guided path network with $N < Z - b - 1$ vehicles, state s is live iff the reachability set $R(s)$ contains a chained state s'.*

Assumption 2.5. *Let us consider a class of guided path networks in which the sense of the vehicle motion is imposed by the following rules on the partially directed graph G:*

a) delete each pass from G;

b) determine the strongly connected components of the graph;

c) impose one sense of the vehicle motion in one of the shortest cycles in each strongly connected component;

d) in each strongly connected component force the sense of the vehicle motion in the remaining shortest cycles according to the sense of vehicle motion imposed by rule c).

The application of these rules on the graph of Fig. 1 is shown in Fig. 2.

The following proposition guarantees the liveness of the system in the proposed class of GPN layouts for each reachable state.

Proposition 2.6. *Let us consider a guided path network where the sense of the vehicle motion is established by Assumption 2.5. If in the live state s there are $N < Z - b - 1$ vehicles and only one vehicle is admitted in each pass, then each state s' $\in R(s)$ is live.*

Proof: In order to prove the proposition, if we show that each state $s' \in R(s)$ is a chained or a semi-chained state, then the sufficient condition of Theorem 2.4 is verified.

Indeed, by Definition 2.2 a semi-chained state is a state that reaches a chained state by a sequence of events involving the advance of a vehicle. Let the system be in a live state s and let $i \in A$ be a vehicle which has to move from zone $z_{x,y}$ to zone $z_{y,w}$ that is not occupied. By Assumption 2.5, four cases are possible:

1. zones $z_{x,y}$ and $z_{y,w}$ are not bridges. Then $z_{x,y}$ and $z_{y,w}$ are edges belonging to a strong component. By Assumption 2.5 if s is chained then the state s' reached by the advance of vehicle i is chained;

2. zone $z_{x,y}$ is not in a cycle and zone $z_{y,w}$ is an edge of a cycle (by hypothesis $z_{y,w}$ can not be a bridge because only one vehicle is admitted in each pass). Then s is semi-chained and by the layout of Assumption 2.5, s' is chained;

3. zone $z_{x,y}$ is an edge of a cycle and zone $z_{y,w}$ is a bridge. In this case the state s is chained and state s' is semi-chained by Assumption 2.5;

4. both $z_{x,y}$ and $z_{y,w}$ are bridges. Then states s and s' are semi chained but in a finite number of steps the state will result in case 2.

Summing up all the reached states are chained or semi-chained. This proves the proposition. □

3. Assignment Problem

This section briefly describes the assignment problem and introduces the consensus algorithm that the vehicles apply in a decentralized approach in order to distribute themselves the tasks. The objective is to minimize the total completion time, i.e. the time necessary to complete all the tasks in the system. Starting from the knowledge of the costs to reach each destination zone in the network, the AGVs at time $t = 0$, in a decentralized manner, reach a consensus about the task that each of them has to perform.

3.1. Assignment problem specification

Let us consider the set A of the AGVs in the system and the set U of tasks that should be assigned to the AGVs at time $t = 0$. The following constraints are considered:

1. each task can be assigned to only one vehicle that can perform only one task (*capacity constraint*);

2. each vehicle can communicate only with the vehicles lying within a communication radius ρ equal to a number of edges of the graph: $\mathcal{N}_n(t) \subset A$ is the set of the AGVs with which AGV $n \in A$ can communicate at time $t \geq 0$ (*communication constraint*);

3. the time is discretized in time units and $t \in \mathbb{N}$.

The problem is defining a discrete consensus algorithm that the AGVs have to perform for autonomously assigning to each of them a destination zone by minimizing the total task completion time.

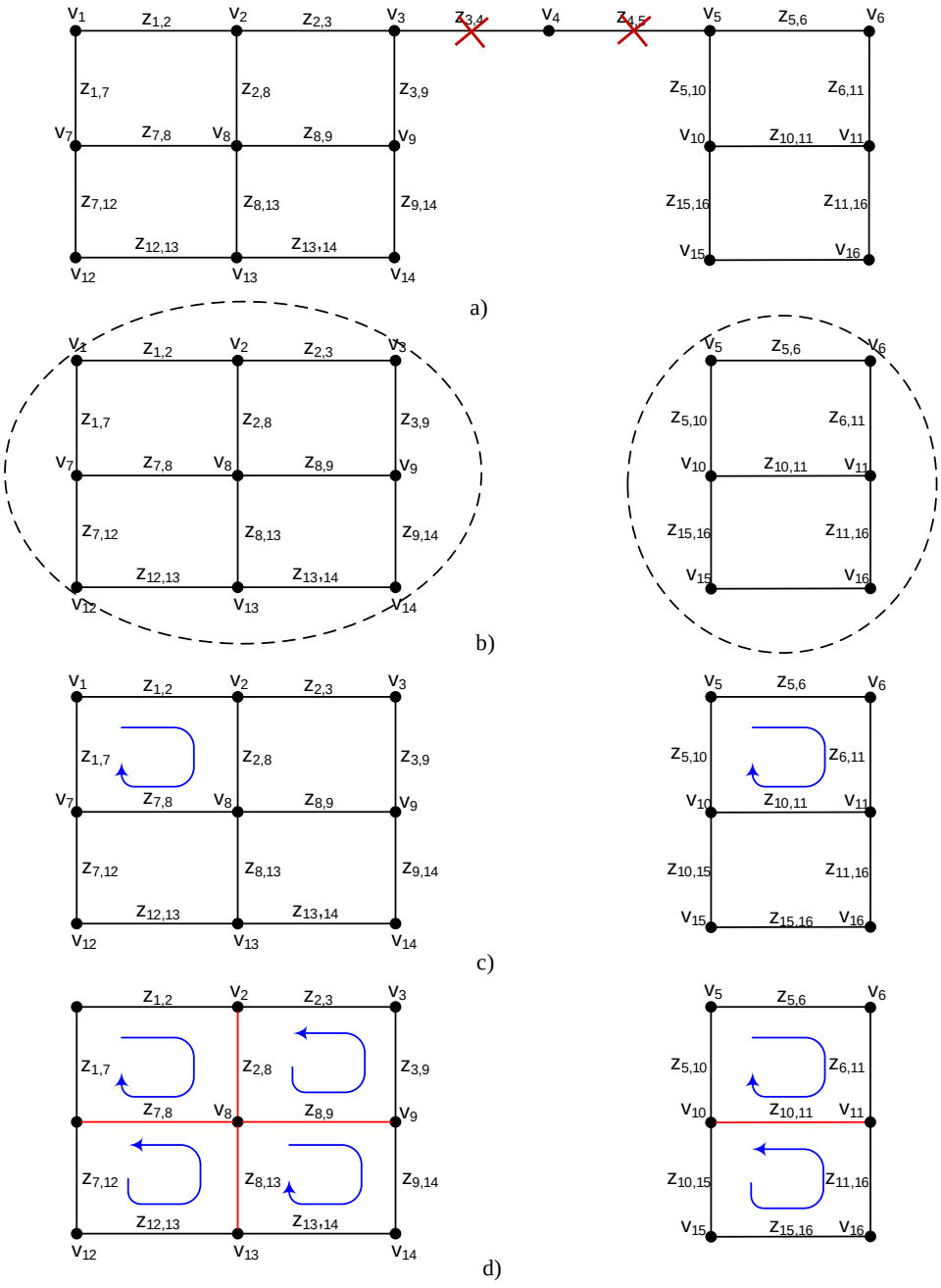

Figure 2: Application of rules of Assumption 1

The communication among the AGVs is modelled by a undirected communication graph $G_C = (A, E)$ where A is the set of nodes (vehicles) and $E \subseteq \{A \times A\}$ is the set of edges: there is an edge from node n to node i if and only if n can transmit information to i and vice versa.

We introduce the following assumptions:

A. each AGV has a full knowledge of the GPN layout but does not know the position of the other AGVs;

B. the initial and goal positions can not be located on the bridges;

C. the communication radius ρ is greater than the length of the longest pass in the network;

D. the communication graph during the assignment phase is connected.

We remark that assumptions B and C are introduced in order to avoid deadlocks according to Proposition 2.6. Moreover, assumption D guarantees the convergence of the assignment algorithm.

The assignment strategy is performed in two phases:

1. cost evaluation phase, each AGV $n \in A$ determines vector c_n, where each element c_{nj} is the cost required to reach destination $j \in U$ in the GPN;

2. assignment phase, the vehicles assign themselves a task by a discrete consensus algorithm.

3.2. Cost evaluation phase

Each AGV $n \in A$ builds the route r_{nj} that is the shortest sequence of zones connecting the current zone occupied by n with the destination zone of j. Now, the current notations are necessary to characterize a route r_{nj}:

- the k-th zone of the zone sequence r_{nj} is denoted by $r_{nj}(k)$ and the number of zones in r_{nj} is indicated by $|r_{nj}|$;
- the operation $r_{nj} = r_{nj} - r_{nj}(k)$ cuts from the route r_{nj} the zone $r_{nj}(k)$;
- the notation $z_{ij} \in r_{nj}$ means that the zone z_{ij} is part of the route r_{nj}.

Hence, the cost required by n to reach the destination zone of $j \in U$ is $c_{nj} = |r_{nj}|$ and is computed by each $n \in A$ for each $j \in U$ by using a generic optimal path search algorithm (for instance A* algorithm [19]).

3.3. Assignment phase

In this section we present the discrete consensus algorithm used by the AGVs for the destination assignment.

Let us denote with $y_n(t) \in \{0,1\}^K$ the assignment vector of vehicle n at time t, composed by the elements $y_{n,j}(t)$ where $y_{n,j}(t) = 1$ if task j is assigned to AGV n at time t, $y_{n,j}(t) = 0$ otherwise.

The state of the network at time t is denoted by the $K \times N$ matrix $Y(t) = [y_1(t)y_2(t)\dots y_N(t)]$.

Moreover, the following sets are defined:

- $I(t) = \{n\} \cup \mathcal{N}_n(t)$, set of vehicles involved in the optimization at time t;
- $J(t) = \{j \in U | y_{i,j}(t-1) = 1 \text{ and } i \in I(t)\}$, set of tasks assigned at time $t-1$ to the vehicles that are involved in the optimization at time t (i.e., the vehicles in set $I(t)$).

We consider the following objective function that represents the maximum value of the task cost (the distance to the task zone) assigned to each vehicle: it represents the makespan, i.e., the time necessary to complete all the tasks:

$$f = max_{n \in A}(c_n^T y_n) \tag{1}$$

The global objective is minimizing the GPN makespan. Now, let us define the following L-ILP that is solved by the set of vehicles $I(t)$ at time t:

L-ILP

$$min\{max_{i \in I(t)}(c_i^T y_i(t)\} \tag{2.a}$$

$$s.t.$$

$$\begin{cases} \sum_{i \in I(t)} y_{i,j}(t) = 1 & \forall j \in J(t) \quad (2.b) \\ \mathbf{1}^T y_i(t) \leq 1, & \forall i \in I(t) \quad (2.c) \\ y_{i,j}(t) \in \{0,1\} & \forall i \in I(t), \forall j \in J(t). \quad (2.d) \end{cases}$$

The local objective function (2.a) to be minimized is the cost of the tasks involved in the local optimization. The first constraint (2.b) states that each task must be assigned to only one AGV belonging to $I(t)$ and involved in

the current optimization; the second constraint (2.c) imposes that only one task is assigned to each vehicle (*capacity constraint*).

At the beginning of the assignment phase starting at time $t = 0$, the initial task assignment is chosen as follows: $y_{i,j}(0) = 1$ if $i = j$ and $y_{i,j}(0) = 0$ if $i \neq j$, $\forall\ i \in A$ and $j \in V$. In such a way, each vehicle $i \in A$ autonomously assigns itself the task $j \in U$ that has its same identification number (i.e., $j = i$). Hence, each L-ILP problem to be solved at time $t > 0$ is feasible. At the subsequent instant time, the updating of the task assignment is performed as follows. At time $t = 1$ a node n wakes up at random and solves the L-ILP with its neighbours. At time $t = t + 1$ for each $t > 1$, a random neighbour node of the agent selected at time t performs the subsequent optimization.

Now, Algorithm 1 describes in details how the vehicles assign to themselves the optimal or suboptimal task assignment in order to minimize the completion time.

Algorithm 1.
Step 1. ***Set*** $t = 0$ *and the initial task assignment* $y_{ij}(0) = 1$ *if* $i = j$ *and* $y_{ij}(0) = 0$ *if* $i \neq j$, $\forall i \in A$ *and* $j \in V$.
Step 2. ***Select*** $n \in A$ *at random with uniform probability.*
Step 3. ***Set*** $t = t + 1$
Step 4. ***Set*** $\mathcal{N}_n(t)$, $I(t) = \{n\} \cup \mathcal{N}_n(t)$, $J(t) = \{j \in U | y_{i,j}(t-1) = 1$ *and* $i \in I(t)\}$.
Step 5. ***Solve*** *the L-ILP problem (2).*
Step 6. ***Let*** *the task assignment* y_i^* *for* $i \in I(t)$ *be the solution provided by the L-ILP problem.* ***If*** $max_{i \in I(t)} c_i^T y_i^* = max_{i \in I(t)} c_i^T y_i(t-1)$ ***then set*** $y_i^* = y_i(t-1)$.
Step 7. ***Set*** $y_i(t) = y_i^*\ \forall\ i \in I(t)$, ***select*** $n \in I(t)$ *at random,* ***go to Step 3****.*

The AGVs start Algorithm 1 by assigning to themselves the tasks with the same identification number (Step 1). At Step 2 a random AGV $n \in A$ wakes up and at Step 4 node n defines the sets $\mathcal{N}_n(t)$ of the vehicles involved in the L-ILP problem solution (itself and its neighbours) and the set $J(t)$ of the tasks to be assigned by solving the L-ILP. Note that such tasks are the ones that at the previous time $t - 1$ were assigned to the neighbours. At Step 5, the L-ILP problem is solved by the AGVs belonging to $I(t)$. Since the objective function is non-increasing, if the solution does not improve the local objective function, i.e., $max_{i \in I(t)} c_i^T y_i^* = max_{i \in I(t)} c_i^T y_i(t-1)$, then at

Step 6 the AGVs set $y_i^* = y_i(t-1)$ (i.e., the assignment does not change at time t). Finally, the algorithm proceeds by updating the task assignment and selecting at random a new node belonging to $I(t)$ (Step 7).

Remark that the Algorithm 1 convergence properties are proved in [17] and the expected convergence time is evaluated by the theory of random walks. It turns out to be equal to $O(N^3)$ for connected graphs and $O(N^2)$ for regular graphs.

After the completion of Algorithm 1 a task $j \in U$ is assigned to each AGV $n \in A$ that has to follow the determined route r_{nj}.

4. Coordination Problem

In this section we describe the coordination phase that the vehicles have to perform in order to reach their destination zone, by avoiding collisions and deadlocks. Assume that the zones in the GPN are of equal length and that the AGVs either stand still or proceed at a fixed velocity. Hence, the time required to travel between two adjacent zones is constant and it is the time unit for the algorithm application. Such assumptions are important for the synchronization of the vehicle moves and the coordination of the AGVs. However, the coordination algorithm could be extended to deal with a more general assumption.

Moreover, each time unit t is further divided into two slots $t = t_c(t) + t_m(t)$: during $t_c(t)$ the neighboring AGVs communicate among them, compete in order to occupy the subsequent zone of their path and determine their state; during the slot $t_m(t)$ each vehicle moves by crossing the node that links its current zone and the following one or stops in its current zone. In the following, in order to simplify the notation, the dependence of sequence r from variable j is removed since to each AGV is assigned at most one destination task.

Now, the following definitions are introduced to describe the dynamic conditions of AGV $n \in A$:

- $\mu_n(t) \in \{PROCEED, STOP, RELOCATE, DEST\}$ denotes the action that n has to perform in the time slot $t_m(t)$ of t ;
- $d_n(t)$ is the sequence of the zones assigned to n at time t.

Moreover, the relationships between $\mu_n(t)$ and $d_n(t)$ at time t are specified as follows:

$\mu_n(t) = PROCEED,\ d_n(t) = r_n\ with\ |r_n| > 1$
$\mu_n(t) = STOP,\ d_n(t) = r_n\ with\ |r_n| > 1$
$\mu_n(t) = RELOCATE,\ d_n(t) = r_n = \langle z_{a,b}, z_{b,d} \rangle$
$\mu_n(t) = DEST,\ d_n(t) = r_n\ with\ |r_n| = 1.$

The state of $n \in A$ at time t is denoted by the pair $x_n(t) = [\mu_n(t), d_n(t)]$ that fully describes the condition of each AGV in the system. In particular:

- if $\mu_n(t) = PROCEED$ then n has to perform the remaining route $d_n(t) = r_n$ and, during $t_m(t)$, n has to move from $r_n(1)$ to $r_n(2)$;
- if $\mu_n(t) = STOP$, then n has to perform the remaining route $d_n(t) = r_n$ and during $t_m(t)$, n has to wait in its current position $r_n(1)$ without proceeding to the subsequent zone;
- if $\mu_n(t) = RELOCATE$, then n has already reached its destination but an AGV $i \neq n$ needs to occupy the position of n, say $z_{a,b}$: consequently, during $t_m(t)$, n has to move in an adjacent free zone, say $z_{b,d}$ (i.e., $d_n(t) = \langle z_{a,b}, z_{b,d} \rangle$);
- if $\mu_n(t) = DEST$ then n arrived to its destination zone and it can hold it till a new destination is assigned or it holds $d_n(t) = \langle z_{a,b} \rangle$, where $z_{a,b}$ is the zone occupied by n.

The following coordination algorithm is executed simultaneously by each AGV $n \in A$ during the time slot $t_c(t)$ in order to determine the state $d_n(t)$ and guarantee deadlock and collision avoidance.

Moreover, some priority rules are applied by the algorithm:

- if an AGV is in a pass it has to complete the route of the pass without stopping;
- if at time t several AGVs compete for the same zone, then the AGV with highest priority can occupy the zone. Here, we consider the priority equal to the number of the zones that the AGV has to visit starting from time t, i.e., we favour among the competing vehicles the one that has to perform the longest path.

Algorithm 2.

Step 1. ***Set*** $t = 1$ *and* $\forall n \in A$ ***Set*** $d_n(t) = r_n$.

Step 2. ***If*** $|r_n| = 1$
then set $\mu_n(t) = DEST$, ***send*** $x_n(t)$ *and* ***go to Step 8***
End If

Step 3. ***If*** $r_n(1) \in p_h$ *and* $p_h \in P$
then set $\mu_n(t) = PROCEED$, ***send*** $x_n(t)$ *and* ***go to Step 9***
End If

Step 4. ***Send*** $d_n(t)$ *and* ***Receive*** $d_i(t) \quad \forall i \in \mathcal{N}_n(t)$

Step 5. ***If*** $\exists\ i \in \mathcal{N}_n(t)$ *such that* $r_i(1) = r_n(2)$ *and* $\mu_i(t) = DEST$
then send $RELOCATE$ *request*
Set $\mu_n(t) = STOP$ *and* ***go to Step 10***
End If

Step 6. ***If*** $\exists\ i \in \mathcal{N}_n(t)$ *such that* $|r_i| > 1$
If $D(r_i(1)) = D(r_n(1))$ *or* $r_i(2) = r_n(2)$
If $|d_n(t)| < |d_i(t)|$
then set $\mu_n(t) = STOP$ *and* ***go to Step 10***
End If
End If
End If

Step 7. ***If*** $r_n(2) \in p_h$, $p_h \in P$
If $\exists\ i \in \mathcal{N}_n(t)$ *such that* $D(r_i(1)) \in p_h \vee r_i(2) \in p_h$
If $\mu_i(t) = PROCEED$ *or* $|d_n(t)| < |d_i(t)|$
then set $\mu_n(t) = STOP$ *and* ***go to Step 10***
End If
Else set $\mu_n(t) = PROCEED$ *and* ***go to Step 9***
End If

Step 8. ***If*** n *receives a* $RELOCATE$ *request*
If $\exists\ z_{b,d}$ *available and adjacent to* $r_n = \langle z_{a,b} \rangle$
then set $r_n = \langle z_{a,b}, z_{b,d} \rangle$, $d_n(t) = r_n$, $\mu_n(t) = RELOCATE$
End If
End If

Step 9. ***Execute*** *the operation according to* $\mu_n(t)$ *in* $t_m(t)$
If $\mu_n(t) = PROCEED$ *or* $\mu_n(t) = RELOCATE$
Then set $d_n(t) = r_n - r_n(1)$,
End If

Step 10. ***Set*** $t = t + 1$ *and* ***go to Step 2***.

First, at Step 1, n initializes the state variable t and the route to the assigned destination $d_n(t)$.

At Step 2, if the vehicle is already arrived at destination, then it sets its state variable $\mu_n(t) = DEST$ and sends to its neighbours its state vector $x_n(t)$ in order to communicate that it could be available to relocate its position.

At Step 3, if the AGV is currently in a pass, it necessarily has to set $\mu_n(t) = PROCEED$ in order to carry on its route, sends the state vector to its neighbours and goes to Step 9.

At Step 4 if the AGV is not arrived at destination and is not currently in a pass, then it sends $d_n(t)$ to its neighbors, also receiving the same data from them.

At Step 5 if n needs to enter a zone, which is already occupied by one of its neighbours $i \in \mathcal{N}_n(t)$ at destination, then n sends a $RELOCATE$ request, sets its state $\mu_n(t) = STOP$ and goes to Step 10.

Step 6 deals with the collision avoidance procedure: AGV n at time t competes with every neighbor $i \neq n$ which has to cross $D(r_n(1))$ or occupy $r_n(2)$. In this case, the AGV with the highest priority can cross the contended node or occupy the contended zone. We give higher priority to the AGV farther from its destination: if $|d_n(t)| < |d_i(t)|$ then n sets $\mu_n(t) = STOP$, otherwise it continues the execution of the algorithm.

At Step 7 AGVs execute the procedure dealing with the access to a pass: if the next zone in the route of vehicle n belongs to a pass, i.e., $r_n(2) \in p_h$, the following cases are possible:

- if pass p_h is already occupied by a vehicle i ($\mu_i(t) = PROCEED$) then n sets $\mu_n(t) = STOP$;
- if pass p_h is free and there is an AGV i trying to access the same pass, then it checks the priority: if $|d_n(t) < |d_i(t)$ then it set $\mu_n(t) = STOP$ else $\mu_n(t) = PROCEED$;
- if n is the only one trying to access the pass or if $r_n(2)$ does not belong to a pass, then n can set $\mu_n(t) = PROCEED$.

Step 8 is reached only if the AGV is already at destination. If it receives a $RELOCATE$ request from a vehicle $i \in \mathcal{N}_n(t)$ it should move to a free

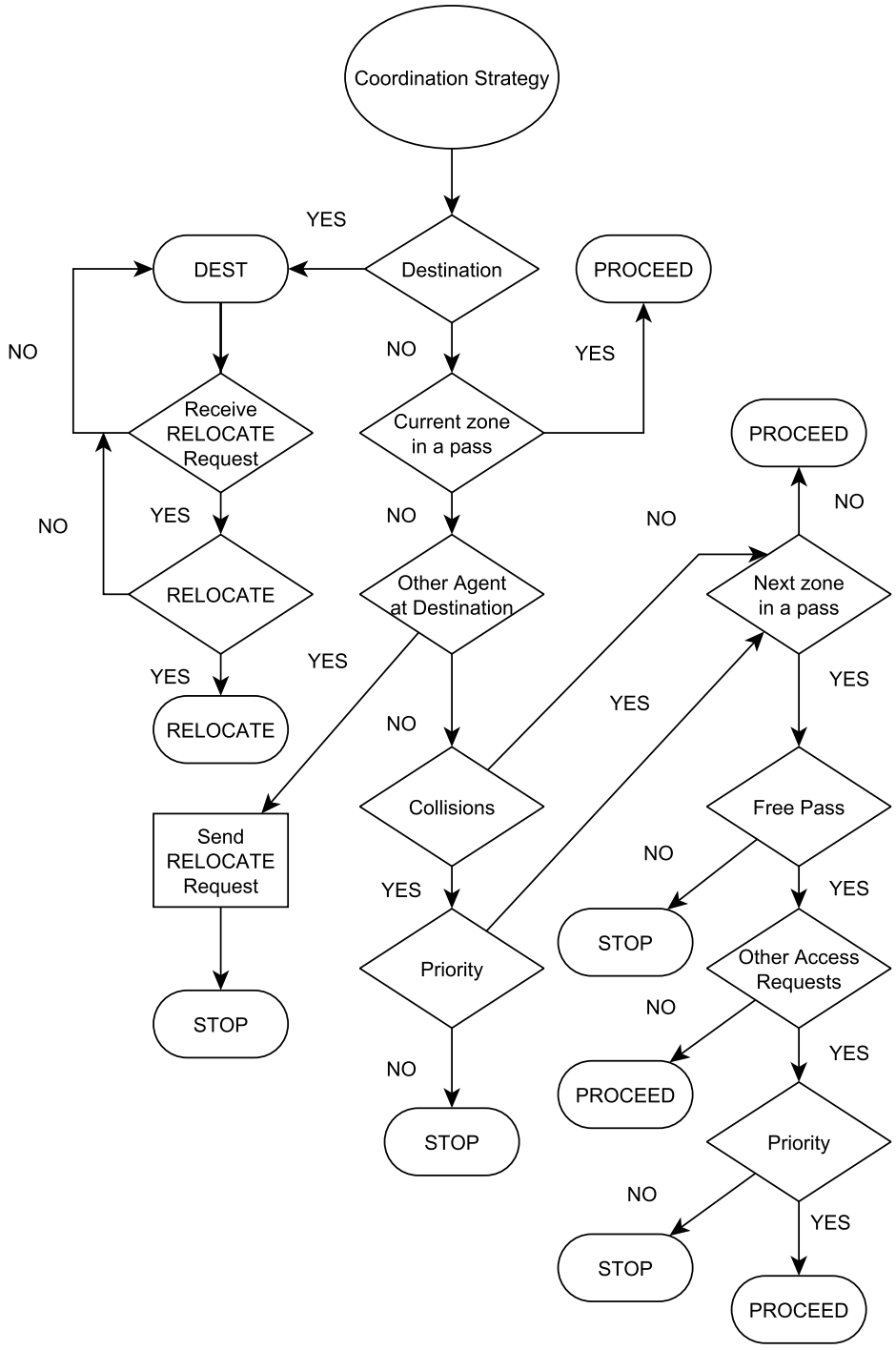

Figure 3: Flow diagram of the coordination strategy.

adjacent zone. If this is possible, it sets $\mu_n(t) = RELOCATE$, otherwise it does not change the value of its variable ($\mu_n(t) = DEST$).

At Step 9, all the vehicles execute the operation according to its state $\mu_n(t)$ and all the vehicles with $\mu_n(t) = PROCEED$ or $\mu_n(t) = RELOCATE$ update their $d_n(t)$ by deleting $r_n(1)$.

The presented algorithm ensures that there are no collisions between vehicles (thanks to the procedure of Step 6) and also satisfies Proposition 2.6, allowing each pass to be occupied only by an AGV at once, thanks to the procedure presented at Step 7.

A flow diagram of the coordination strategy is reported in Fig. 3 and it is easy to infer that the complexity of Algorithm 2 is of $O(N)$.

With respect to the coordination algorithm proposed in [18], this paper describes more in detail the state of each AGV by introducing the new state condition $DEST$. As a consequence, the updating of the vehicle states and the actions that the AGVs have to perform at each step are simplified and clarified. Moreover, the more rational formalism to describe the vehicle states allows reducing the data that the vehicles have to exchange. More precisely, in [18] each AGV has to send, at the beginning of the algorithm, its state, the zone that it occupies and the requested one. In this version of the algorithm, the AGVs send just the necessary data.

5. Application of the Zone-Control Strategy by a Simulation Software

In this section the proposed algorithms are applied to a GPN inspired by a company located in the Southern Italy. The company designs AGV systems and the relative control software, then it asks for decentralized management strategies in order to deal with large fleets of AGVs. With the aim of showing the real applicability of the proposed algorithm, a simulation software describes the GPN vehicles behaviour under the presented zone-control strategy. For the sake of brevity, a simplified topology of the real case study is considered.

5.1. Simulation Software

The simulation software is developed using C++ with the Ubuntu QML Toolkit and performs the following four procedures.

1. `calculate_path_cost()`: the software allows the definition of the GPN and the sense of motions for each edge. Then, it is possible to select the

positions of the AGVs and the destinations. Moreover, the possible paths and their relative costs for each AGV in the network to reach the available destinations are determined;

2. `evdaa()`: Algorithm 1 is applied iteratively and the L-ILP problems are solved by an AGV with its neighbours. This procedure makes use of the C++ library `lp_solve`;

3. `coordination()`: the software performs Algorithm 2 and updates the paths for each AGV in the network;

4. `gui()`: the stored paths are displayed to the user step by step, highlighting the destination zones and showing that each vehicle reaches its destination avoiding collisions and deadlock.

5.2. Example Simulation

In this subsection the GPN of Fig. 4 is described: the menu in Fig. 5 allows the user to determine the GPN width and to select the nodes which belong to the graph by clicking on them. Then, the user selects the edges to be removed from the GPN: in this example, edges $z_{15,21}$, $z_{16,22}$, $z_{17,23}$ and $z_{18,24}$ are removed since they are not present in the graph of Fig. 4. In addition, the user can select the bridge edges belonging to the graph and performs the operations reported in Assumption 2.5.

The software shows the GPN with the corresponding sense of motions for each edge (Fig. 6) and allows selecting the positions of the AGVs and the destination zones: $N = 9$ vehicles (in red) and $K = 9$ destination tasks (in green) are shown in Fig. 7. Note that the communication radius is $\rho = 3$ since the longest pass in the network is $p_1 = \{v_3, z_{3,4}, v_4, z_{4,5}, v_5\}$ of length 2: therefore this choice abides by the assumptions of Section 3.2.

Then, the software performs the function `calculate_path_costs()` which computes the costs of the shortest paths connecting each vehicle and each destination. Table 1 shows on the first column the initial positions of each vehicle i and on the second row the positions of each task j with the corresponding cost $c_{i,j}$.

Moreover, given the initial positions of the vehicles and the value of ρ, it is possible to obtain the communication graph G_c for $t = 0$, represented in Fig. 8.

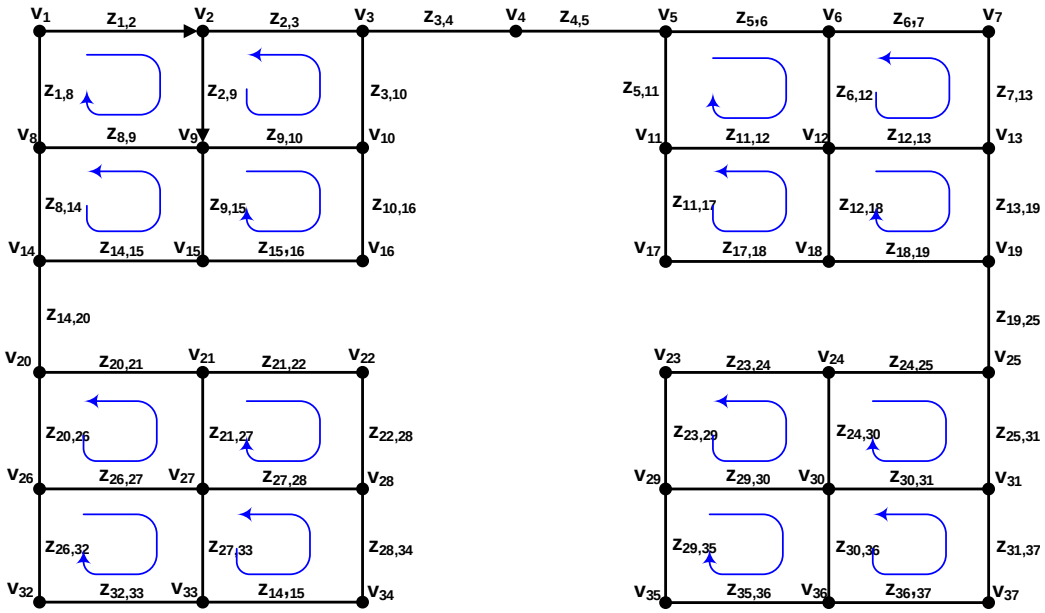

Figure 4: The guided path network of the example.

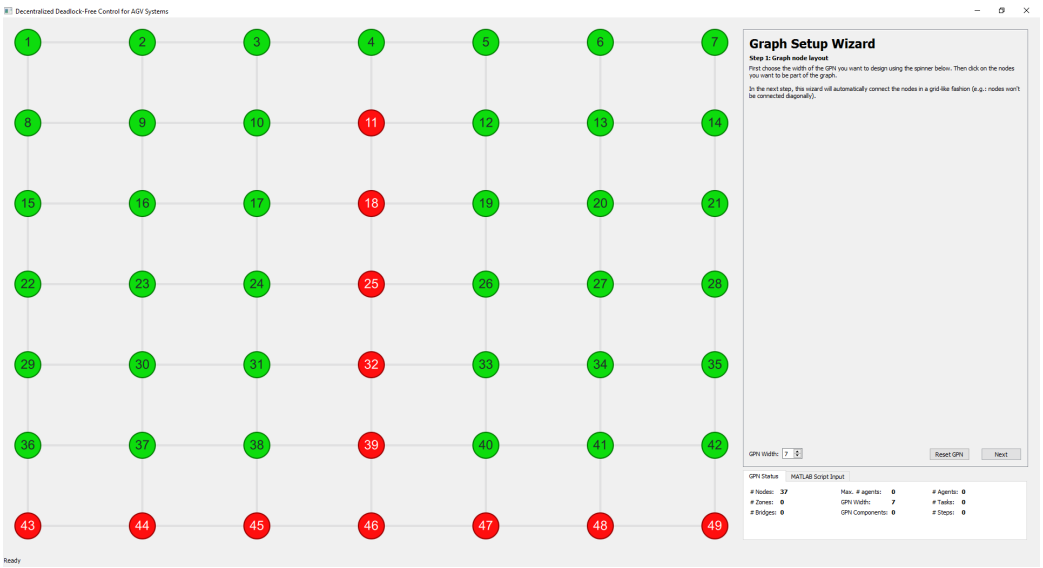

Figure 5: Simulation Software: selecting the nodes.

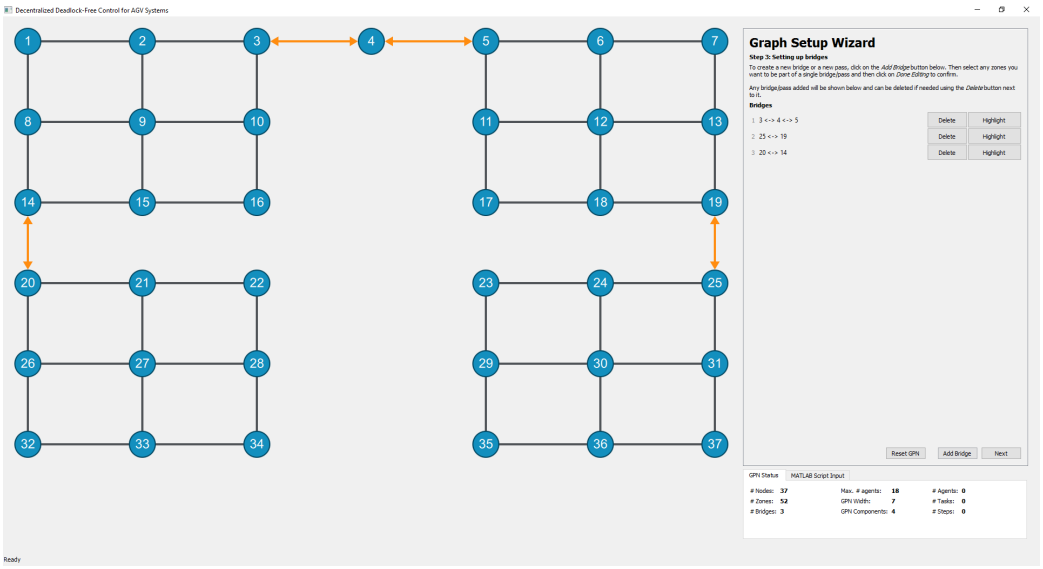

Figure 6: Simulation Software: GPN setup.

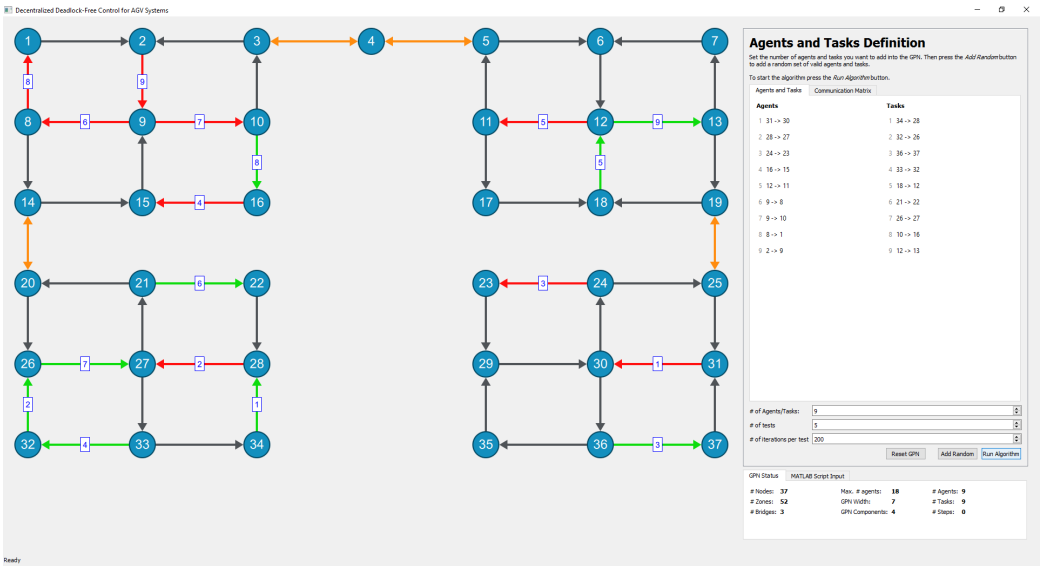

Figure 7: Simulation Software: initial AGV positions.

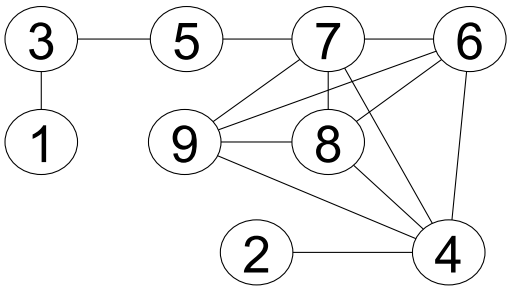

Figure 8: Communication graph at time $t = 0$.

Table 1: AGV and Task Positions and Costs

	Task Position								
AGV Position	$z_{30,31}$	$z_{27,28}$	$z_{23,24}$	$z_{15,16}$	$z_{11,12}$	$z_{8,9}$	$z_{9,10}$	$z_{1,8}$	$z_{2,9}$
$z_{28,34}$	19	19	2	18	5	18	16	13	6
$z_{26,32}$	3	3	18	2	15	2	4	7	12
$z_{36,37}$	21	21	4	20	7	20	18	15	8
$z_{32,33}$	9	9	14	8	11	8	6	3	8
$z_{12,18}$	13	13	10	12	3	12	10	7	4
$z_{21,22}$	7	7	16	6	13	6	4	5	10
$z_{26,27}$	11	11	12	10	9	10	8	1	6
$z_{10,16}$	10	10	15	9	12	9	7	4	9
$z_{12,13}$	8	8	13	7	10	7	5	2	7

The software then performs the `evdaa()` function simulating the AGVs that apply Algorithm 1. The AGVs start at time $t = 0$ with the specified initial task assignment: $y_{i,j}(0) = 1$ if $i = j$ and $y_{i,j}(0) = 0$ if $i \neq j$, for $i = 1, ..., 9$ and $j = 1, ..., 9$.

Moreover, the vehicle assignments evolve as shown in Table 2: the first column denotes the current iteration of the algorithm, the second column indicates the vehicle that at time t begins the optimization with the neighbors that are reported in the third column. In addition, the subsequent columns denote the ids of the tasks assigned to each AGV in the current iteration and the last one reports the value of the global objective function in the considered iteration.

Table 2: Execution of Algorithm 1

t	i	$\mathcal{N}_i$	AGVs and Assigned Tasks									$f(y^*(t))$
			1	2	3	4	5	6	7	8	9	
0			1	2	3	4	5	6	7	8	9	19
1	3	1,3,5	5	2	3	4	1	6	7	8	9	13
2	5	3,5,7	5	2	3	4	7	6	1	8	9	11
3	7	4,5,6,7,8,9	5	2	3	4	9	1	8	7	6	8
4	4	2,4,6,7,8,9	5	1	3	4	9	2	8	7	6	8
5	6	4,6,7,8,9	5	1	3	4	9	2	8	7	6	8
6	7	4,5,6,7,8,9	5	1	3	4	9	2	8	7	6	8

We observe that for $t \geq 4$ the task distribution between the AGVs does not change, leading to the final solution with the objective function $f(y^*(t)) = 8$.

Then function `coordination()` runs and simulates the Algorithm 2 application. Table 3 reports the sequences of the zones occupied by each vehicle at the time instants shown in the first column. The algorithm proceeds until $t = 14$ when all the vehicles reach their respective destinations.

The execution of Algorithm 2 is displayed step by step by function `gui()`, showing each AGV reaching the assigned destination without collisions and deadlocks.

Furthermore, Figures 9, 10 and 11 report the travels of the vehicles in three steps. In particular, Fig. 9 shows the initial positions of each AGV, Fig. 10 and 11 show the AGV positions at steps $t = 3$ and $t = 12$, displaying how the system manages pass accesses: at step $t = 3$ the AGVs 4, 6, 8 and 9 need to cross pass $\{v_{14}, z_{14,20}, v_{20}\}$; vehicle 6 is the first one to occupy the pass by crossing v_{14} while the other AGVs stop. At step $t = 12$, AGVs 6, 9 and 4 have crossed the pass and AGV 8 can cross it in order to reach its destination zone.

Note that at the bottom left of each figure the time required to run the control strategy is shown and it is equal to 3.493 seconds.

Lastly, Fig. 12 shows the configuration of the network at the final step of Algorithm 2 where all the vehicles reach their destinations.

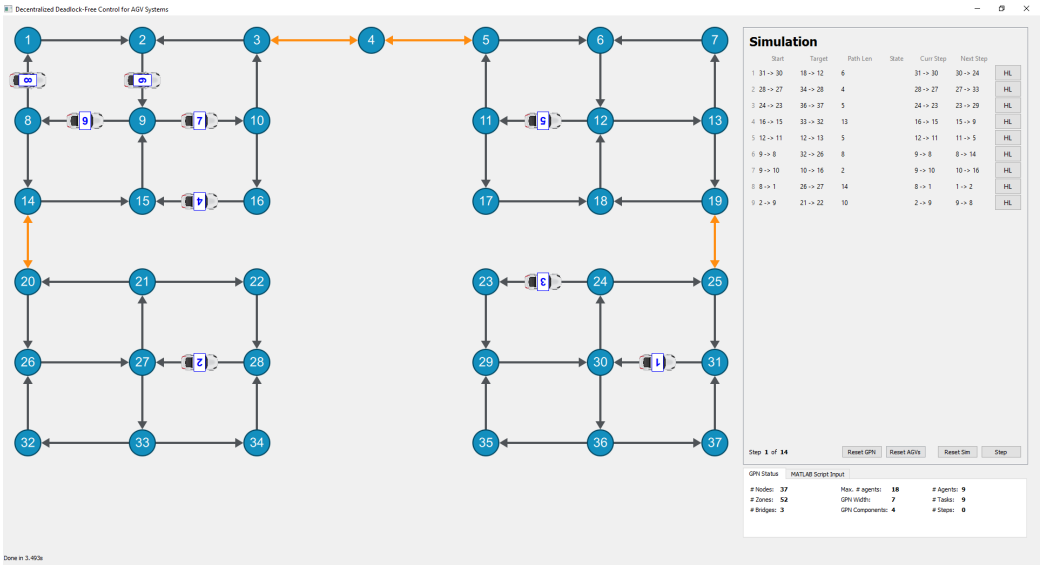

Figure 9: Coordination: initial AGV positions.

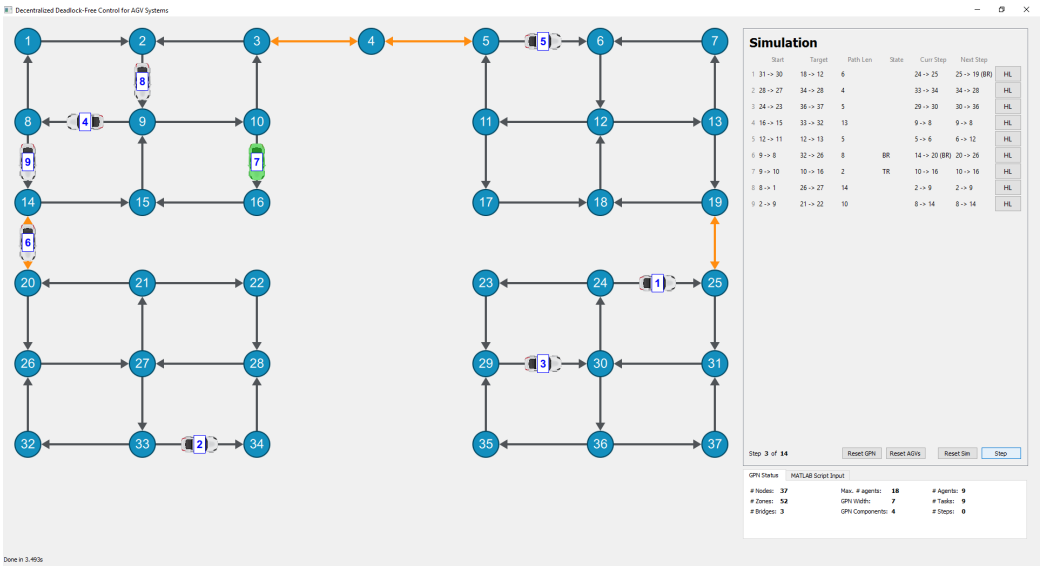

Figure 10: Coordination: access to the pass at time $t = 3$.

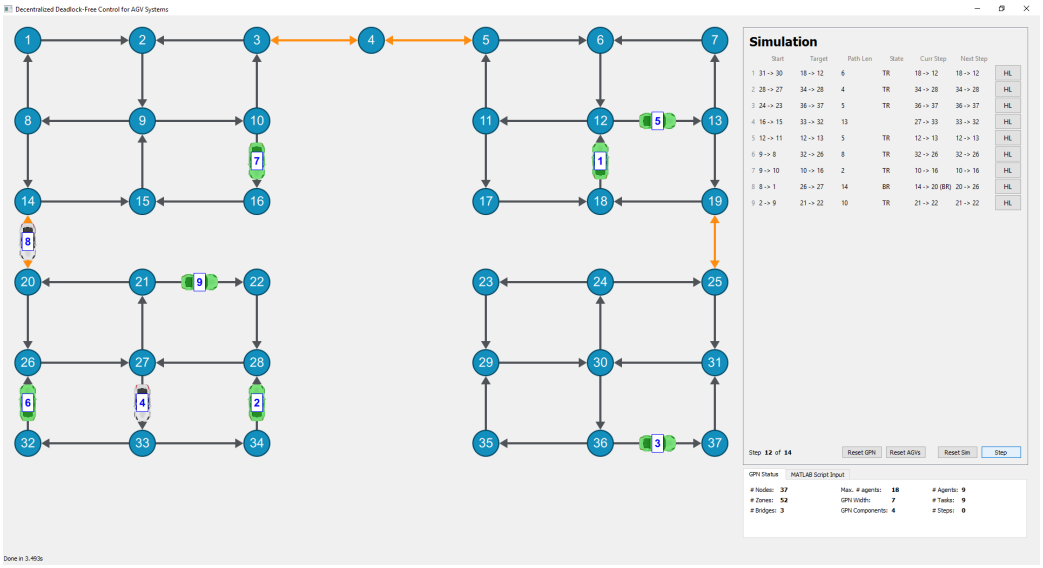

Figure 11: Coordination: access to the pass at time $t = 12$.

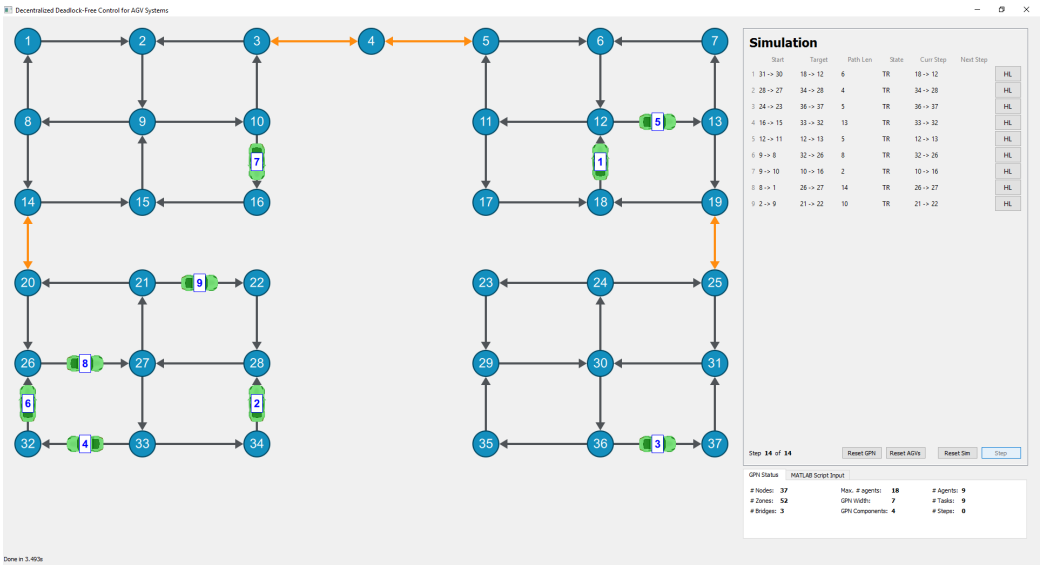

Figure 12: Final positions of vehicles.

Table 3: Execution of Algorithm 2

	Occupied Zones								
t	1	2	3	4	5	6	7	8	9
1	$z_{30,31}$	$z_{27,28}$	$z_{23,24}$	$z_{15,16}$	$z_{11,12}$	$z_{8,9}$	$z_{9,10}$	$z_{1,8}$	$z_{2,9}$
2	$z_{24,30}$	$z_{27,33}$	$z_{23,29}$	$z_{9,15}$	$z_{5,11}$	$z_{8,14}$	$z_{10,16}$	$z_{1,2}$	$z_{8,9}$
3	$z_{24,25}$	$z_{33,34}$	$z_{29,30}$	$z_{8,9}$	$z_{5,6}$	$z_{14,20}$	$z_{10,16}$	$z_{2,9}$	$z_{8,14}$
4	$z_{19,25}$	$z_{28,34}$	$z_{30,36}$	$z_{8,9}$	$z_{6,12}$	$z_{20,26}$	$z_{10,16}$	$z_{2,9}$	$z_{8,14}$
5	$z_{18,19}$	$z_{28,34}$	$z_{36,37}$	$z_{8,9}$	$z_{12,13}$	$z_{26,27}$	$z_{10,16}$	$z_{2,9}$	$z_{8,14}$
6	$z_{12,18}$	$z_{28,34}$	$z_{36,37}$	$z_{8,14}$	$z_{12,13}$	$z_{27,33}$	$z_{10,16}$	$z_{8,9}$	$z_{14,20}$
7	$z_{12,18}$	$z_{28,34}$	$z_{36,37}$	$z_{8,14}$	$z_{12,13}$	$z_{32,33}$	$z_{10,16}$	$z_{8,9}$	$z_{20,26}$
8	$z_{12,18}$	$z_{28,34}$	$z_{36,37}$	$z_{8,14}$	$z_{12,13}$	$z_{26,32}$	$z_{10,16}$	$z_{8,9}$	$z_{26,27}$
9	$z_{12,18}$	$z_{28,34}$	$z_{36,37}$	$z_{14,20}$	$z_{12,13}$	$z_{26,32}$	$z_{10,16}$	$z_{8,14}$	$z_{21,27}$
10	$z_{12,18}$	$z_{28,34}$	$z_{36,37}$	$z_{20,26}$	$z_{12,13}$	$z_{26,32}$	$z_{10,16}$	$z_{8,14}$	$z_{21,22}$
11	$z_{12,18}$	$z_{28,34}$	$z_{36,37}$	$z_{26,27}$	$z_{12,13}$	$z_{26,32}$	$z_{10,16}$	$z_{8,14}$	$z_{21,22}$
12	$z_{12,18}$	$z_{28,34}$	$z_{36,37}$	$z_{27,33}$	$z_{12,13}$	$z_{26,32}$	$z_{10,16}$	$z_{14,20}$	$z_{21,22}$
13	$z_{12,18}$	$z_{28,34}$	$z_{36,37}$	$z_{32,33}$	$z_{12,13}$	$z_{26,32}$	$z_{10,16}$	$z_{20,26}$	$z_{21,22}$
14	$z_{12,18}$	$z_{28,34}$	$z_{36,37}$	$z_{32,33}$	$z_{12,13}$	$z_{26,32}$	$z_{10,16}$	$z_{26,27}$	$z_{21,22}$

6. Conclusions

This paper proposes a decentralized control strategy for assigning paths and coordinating Autonomous Guided Vehicles (AGVs) in Guided Path Networks (GPNs).

First, the AGVs apply a decentralized discrete consensus protocol in order to autonomously assign themselves the destination task by iteratively solving a Local-Integer Linear Programming problem. Then the AGVs move across the network to reach the assigned destination zone by performing a decentralized coordination algorithm based on a zone-controlled approach. The proposed coordination procedure guarantees collision avoidance and it is proved that, in the proposed class of guided-path network layouts, also guarantees the deadlock avoidance. Then, a software is presented, which allows performing simulations of the proposed strategy, also showing to the user the result of the application of the two algorithms to a realistic scenario.

Future work will investigate on two main aspects: i) analysing the possibility of optimizing the task distribution also considering different objective functions and more generic costs; ii) studying deadlock avoidance strategies that could be applied on more generic network topologies.

References

[1] E. Roszkowska, S. Reveliotis, On the liveness of guidepath-based, zone-controlled dynamically routed, closed traffic systems, IEEE Transactions on Automatic Control 53 (7) (2008) 1689–1695.

[2] M. P. Fanti, Event-based controller to avoid deadlock and collisions in zone-control AGVs, International Journal of Production Research 40 (6) (2002) 1453–1478.

[3] M. Fanti, A deadlock avoidance strategy for AGV systems modelled by coloured Petri nets, in: Sixth International Workshop on Discrete Event Systems, 2002, 2002, pp. 61–66.

[4] J. Park, S. Reveliotis, Liveness-enforcing supervision for resource allocation systems with uncontrollable behavior and forbidden states, IEEE Transactions on Robotics and Automation 18 (2) (2002) 234–240.

[5] R. Lochana Moorthy, W. Hock-Guan, N. Wing-Cheong, T. Chung-Piaw, Cyclic deadlock prediction and avoidance for zone-controlled AGV system, International Journal of Production Economics 83 (3) (2003) 309–324.

[6] A. Krnjak, I. Draganjac, S. Bogdan, T. Petrovic, D. Miklic, Z. Kovacic, Decentralized control of free ranging agvs in warehouse environments, in: Robotics and Automation (ICRA), 2015 IEEE International Conference on, 2015, pp. 2034–2041.

[7] D. Marino, A. Fagiolini, L. Pallottino, Distributed collision-free protocol for AGVs in industrial environments, CoRR (2011) –1–1.

[8] N. Ayanian, D. Rus, V. Kumar, Decentralized multirobot control in partially known environments with dynamic task reassignment, in: IFAC Workshop on Distributed Estimation and Control in Networked Systems, Santa Barbara, CA, 2012, pp. 311–316.

[9] N. Wu, M. Zhou, Agv routing for conflict resolution in AGV systems, in: IEEE International Conference on Robotics and Automation, 2003 ICRA '03, Vol. 1, 2003, pp. 1428–1433 vol.1.

[10] M. Jager, B. Nebel, Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots, in: 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2001, Vol. 3, 2001, pp. 1213–1219 vol.3.

[11] V. Digani, L. Sabattini, C. Secchi, C. Fantuzzi, Ensemble coordination approach in multi-agv systems applied to industrial warehouses, Automation Science and Engineering, IEEE Transactions on 12 (3) (2015) 922–934.

[12] V. Digani, L. Sabattini, C. Secchi, A probabilistic eulerian traffic model for the coordination of multiple agvs in automatic warehouses, IEEE Robotics and Automation Letters 1 (1) (2016) 26–32.

[13] B. Park, J. Choi, W.-K. Chung, An efficient mobile robot path planning using hierarchical roadmap representation in indoor environment, in: 2012 IEEE International Conference on Robotics and Automation (ICRA), 2012, pp. 180–186.

[14] S. Reveliotis, E. Roszkowska, Conflict resolution in free-ranging multivehicle systems: A resource allocation paradigm, IEEE Transactions on Robotics 27 (2).

[15] L. Pallottino, V. Scordio, A. Bicchi, E. Frazzoli, Decentralized cooperative policy for conflict resolution in multivehicle systems, IEEE Transactions on Robotics 23 (6) (2007) 1170–1183.

[16] M. Fanti, M. Franceschelli, A. Mangini, G. Pedroncelli, W. Ukovich, Discrete consensus in networks with constrained capacity, in: 2013 IEEE 52nd Annual Conference on Decision and Control (CDC), 2013, pp. 2012–2017.

[17] M. M. Fanti, A. Mangini, G. Pedroncelli, W. Ukovich, Discrete consensus for asynchronous distributed task assignment, in: 2016 IEEE 52nd Annual Conference on Decision and Control (CDC), 2016, pp. 2012–2017.

[18] M. P. Fanti, A. M. Mangini, G. Pedroncelli, W. Ukovich, Decentralized deadlock-free control for agv systems, in: 2015 American Control Conference (ACC), 2015, pp. 2414–2419.

[19] P. Hart, N. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107.