A time expanded network based algorithm for safe and efficient distributed multi-agent coordination

Mirko Ferrati and Lucia Pallottino

***Abstract*— In this paper we propose a novel approach for distributed traffic management of a group of mobile collaborative vehicles, moving within a shared environment. Our algorithm is based on a modified graph representation of the space-time where the robotic vehicles operate. The proposed graph representation augments standard path planning strategies by allowing multiple speeds. Robots can negotiate their route and set their speed to prevent and solve possible collisions while taking into account the energy consumption. Indeed, the algorithm will be formally proved to provide collision free paths.**

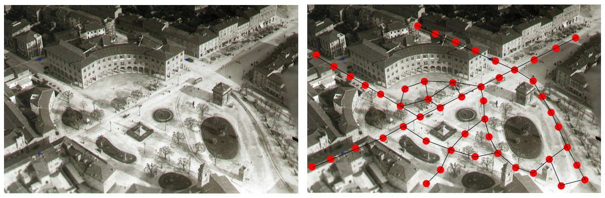

Fig. 1. A square in Pisa modelled as a graph

I. Introduction

The use of multi-vehicle robotic systems is increasing rapidly in, e.g., industrial transportation and logistics systems. However, their adoption raises management and coordination problems such as collision avoidance and conflict resolution both requiring fast and reliable negotiation of shared resources.

In this paper we propose a distributed coordination protocol to address such issues in case of a traffic network for autonomous vehicles. The goal is to guarantee the overall system safety while taking into account the energy consumption aspects.

In case of environment tessellation, cells are seen as resources for which vehicles compete. Several centralized and distributed control strategies have been proposed in the literature, see e.g. [1], [2], [3].

While centralized approaches are able to easily guarantee the absence of deadlock, they often require an unfeasible computational capabilities. The required computational capabilities could be lowered by using distributed algorithms, which in turn pose more challenging problems in proving deadlock freedom. Examples of research in this field are represented in [4], where the agent coordination is obtained by using a spatial temporal pattern, and in [5], where the collision avoidance policy is inspired by real world pedestrian collisions policy.

In [6], authors are able to prove that a class of algorithms is livelock free based on the proposed discretization of the environment while in [7], the problem is solved by allowing more than one agent in the same cell. A similar approach is used in this paper. In [8], collisions are prevented in a reactive fashion by the definition an ad-hoc protocol that allow agents to vary their speeds. However, as a result agents may often need to change speed resulting in a non smooth (and non efficient in terms of energy consumption) system evolution.

Although inspiring, all these approaches do not take into account the energy cost that an agent incurs in when it stops to re-plan its path by applying its collaboration algorithm. When a vehicle (a forklift, a car, etc) brakes, its kinetic energy is converted to heat by the friction of the brake pads. The total amount of energy loss depends on how often and how hard the vehicle brakes, see e.g. [9], [10], [11]. Indeed, based on the analysis results of the papers, fuel consumption is mainly affected by extreme acceleration or deceleration rates (as long as vehicles are driving below 80 Km/h). From those studies, we can state that the sudden deceleration/stop/acceleration behaviour will result in negative influences on fuel consumption.

Based on those results, the proposed coordination protocol allows the agents to keep moving while coordinating/communicating to avoid collision or planning their routes. Indeed, in the proposed framework, an agent stops only in case of exceptional events (such as unsolvable conflicts) which are handled with exceptional behaviours (i.e. stopping agents in the neighbourhood and using a fallback behaviour). The approach is based on the distributed use of a normalized and augmented time-expanded network for the computation of the vehicle's speed and is proven to be deadlock free under given conditions on the number of agents. To the authors best knowledge this is the first work that exploits time-expanded network to produce a distributed collision avoidance and coordinated planning which is able to control both agents speed and paths, in order to reduce energy consumption due to "stops&go" strategies.

II. Time Expanded Network

The concept of time expanded network (TEN) is briefly described here as in [12] for reader convenience. For more details and some use examples see e.g. [13], [14], [15], [16]. In [17] authors use time-expanded network to provide a centralized coordinator for multiple vehicle systems.

Consider a weighted digraph $G_0 = (V_0, E_0)$, where the weight d_{ij} of an arc $e_{i,h} = (v_i, v_j)$ is the arc travel time

The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreements n.257462 HYCON2 Network of excellence and n.2577649 PLANET.

Authors are with the Research Center "Enrico Piaggio", and the Dipartimento di Ingegneria dell'Informazione, University of Pisa, Italy, mirko.ferrati@gmail.com, l.pallottino@centropiaggio.unipi.it.

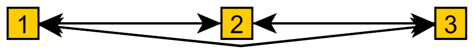

Fig. 2. Basic graph, the weight of arc (1, 3) doubles the weight of arcs (1, 2) and (2, 3).

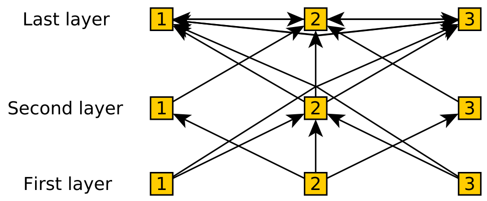

Fig. 3. TEN associated to the graph in Fig. 2 with $M-1=3$

or delay where, if $t>0$ is the leaving time from node v_i, $t+d_{ij}$ is the arrival time at node v_j.

In case of discrete models, both the time variable t and travel time d_{ij} are natural numbers so that $t+d_{ij}\in\mathbb{N}$.

The finite TEN digraph $G_t(V_t,E_t)$ is obtained by expanding the original dynamic network G_0 in the time dimension, and making a separate copy of all nodes for the integer values of time $t\in\mathcal{T}=\{0,1,\dots,M-1\}$, where M is a finite maximum value for the time dimension.

Every node in V_t is a node-time pair (v_i,t) where $v_i\in V_0$ and $t\in\mathcal{T}$, where t represents the *time layer* associated to v_i. The nodes with time layer value $M-1$ represent all time layers with $m\geq M-1$.

Every edge in E_t is an edge from (v_i,t_1) to (v_j,t_2) with $t_2>t_1$. Furthermore, if $((v_i,t_1),(v_j,t_1))\in E_t$ with $i\neq j$ then, necessarily, $t_1=M-1$.

Given the planar graph G_0 represented in Fig. 2, the TEN associated to G_0 with $M-1=3$ is reported in Fig. 3. Notice that, in the TEN, to take into account the weights $d_{i,j}$ of $e_{i,j}\in E_0$, arcs connecting time layer k with time layer $k+h$ with $h>1$ may be considered. Such arcs are *h-multi-layer arcs*. Notice that, given a time discretization of step T, we have $h=\frac{d_{i,j}}{T}$.

The first layer represents the current time: agent is always at the first layer l_1 and whenever it reaches the second layer l_2 all layers are translated by one to the bottom (l_{i-1} is replaced by l_i).

Time-expanded networks have several properties. For example, in case of positive travel times, TEN are acyclic when neglecting the arcs at the time layer $M-1$. Moreover, a shortest path problem in a dynamic network can be solved by applying a static shortest path algorithm to its TEN representation.

III. Preliminaries, model description and problem statement

To model the two dimensional world (street maps or indoor environment) in which agents move, we identify regions of interest and we associate a key-point to each region, usually corresponding to their center. Key-points will be connected by roads and agents will only be able to go from a region to another by moving on roads, see Figure 1.

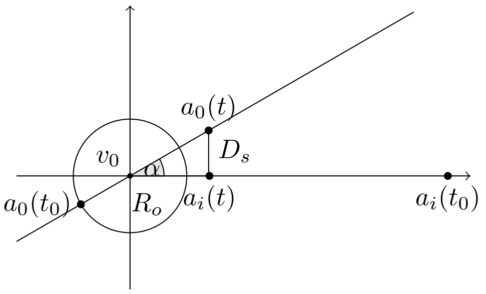

Fig. 4. Time separation for avoiding collisions on a node.

Let $G_0=(V_0,E_0)$ be the graph representation of the world, where $V_0=\{v_1,v_2,\dots,v_N\}$ and $E_0\subseteq V_0\times V_0$ represent the set of nodes (key-points) and the set of edges (road from one key-point to another), respectively. To each node v_i we associate the corresponding coordinates $(X,Y)_{v_i}$ in the environment. On the other hand, each arc e_{ij} has a weight that is the distance $L_{e_{ij}}$ from $(X,Y)_{v_i}$ to $(X,Y)_{v_j}$.

Consider a group of m agents, the position of an agent a_j in the environment is represented by its coordinates $(X,Y)_{a_j}$. The maximum speed of agents is assumed to be the same for all agents and indicated with v_{max}, if this is not the case assume in a conservative fashion $v_{max}=\min_i v_{max_i}$, so that all agents are able to move at a common maximum speed. Similarly, the minimum non null speed is $v_{min}=\max_i v_{min_i}$. Let D_s the minimum required distance (*safety distance*) to be kept between agents to avoid collisions that depends on vehicles physical dimensions and motion constraints.

Assumption 1: Two (or more) different roads in the real world may overlap only close to regions associated to key–points, i.e. in a circle of radius R_k centered in them.
As a consequence, road intersections are always considered as key–points, so that agents moving on different arcs will not collide except when close to the same key–point.

From now on, we will consider regions centered in the nodes with the most conservative $\tilde{R}$ value, i.e. $\tilde{R}=\max_{k\in\{1,\dots,N\}}R_k$.

Definition 1: A node v_i is *occupied* by agent j at time t, denoted with $v_i(t)=a_j$, if $\left\|(X,Y)_{a_j}-(X,Y)_{v_i}\right\|<\tilde{R}$ holds at time t.

Assumption 2: We assume that the distance between any two key–points is larger that $2\tilde{R}+\frac{D_s}{2}$.
Under this assumption there can be no collision between two agents occupying different nodes.

Definition 2: There is no *collision* between agents a_i and a_j if $\left\|(X,Y)_{a_j}-(X,Y)_{a_i}\right\|\geq D_s$.
We first examine the case of collisions between agents near the same node, while we will examine the case of collisions between agents on the same arc in Theorem 1.

Proposition 1: Given an agent a_0, at time t_0, occupying node v_0 and moving at v_{min} from node v_0 to node v_b and an agent a_i moving at v_{max} from node v_c to v_0, where $v_b\neq v_c$, there is no collision on node v_0 if agent a_i does *not* occupy it *before* time $t_0+\Delta t$, where $\Delta t=D_s\frac{1+\cos\alpha}{\sin\alpha}\frac{l_{max}}{v_{max}}$, $l_{max}=v_{max}/v_{min}$ and α is the angle of roads intersection.

Proof: Without loss of generality, we will assume that the road (v_0,v_c) is parallel to the x axis of a reference system centered in the node v_0 (i.e., $(X,Y)_{a_0} = (0,0)$ and a_i is moving on $y = 0$) and that $t_0 = 0$, see Fig. 4.

In the worst case, if a collision between agents a_i and a_0 does not occur, there $\exists \bar{t} : \|(X,Y)_{a_i} - (X,Y)_{a_0}\| = D_s$, i.e. the distance is exactly the minimum allowed one. At that time $\bar{t}$ the position of agent a_0 is $X_{a_0} = v_{min}(\bar{t} - \frac{R_o}{v_{min}})\cos\alpha$, $Y_{a_0} = v_{min}(\bar{t} - \frac{R_o}{v_{min}})\sin\alpha$. To reach a configuration in which agents are at minimum distance, at time $\bar{t}$, agent a_i must be in $(x, 0)$ such that $X_{a_0}^2 - 2X_{a_0}x + x^2 + Y_{a_0}^2 = D_s^2$ where this second order equation has two coincident solutions. In this case $\bar{t} = (D_s + R_o \sin\alpha)/(v_{min}\sin\alpha)$ and then $x = D_s/\tan\alpha$. Since the position of agent a_i at time $t_0 = 0$ was $x + v_{max}\bar{t}$, we can determine the time Δt at which a_i occupies v_0:
$\Delta t = (D_s/\tan\alpha - R_o)/v_{max} + \bar{t} = (\frac{D_s}{\sin\alpha}(l_{max} + \cos\alpha) + R_o(l_{max} - 1))/v_{max}$, where we used $l_{max} = v_{max}/v_{min}$.

Considering $R_o = \frac{D_s}{\tan\alpha}$, we obtain that $\Delta t = D_s l_{max}(1 + \cos\alpha)/(v_{max}\sin\alpha)$ and hence the thesis. ■

Remark 1: Proposition 1 provides a time value Δt that will be used to create a discretized TEN. In the proof of the Proposition we have provided a value of the parameter R_o to obtain a nice expression of the time separation required to ensure collision avoidance on a node in the worst case scenario of the following cases: an agent occupying a node, another one crossing an incoming arc of that node, an agent occupying a node, another one crossing an outgoing arc of that node, two agents crossing two different incoming arcs of the same node.
Such value depends on the angle of intersection of roads that can be however minimized by a value α_{min}. Since larger angles produce larger values of R_o, the proof of the Proposition is still valid considering $R_o = \tilde{R} = \frac{D_s}{\tan\alpha_{min}}$).
Notice that the ratio $l_{max} = v_{max}/v_{min}$ is another parameter that will be set later in the paper.

Finally, by extending the Proposition 1 to all agents and nodes, the absence of collision on all the nodes of the graph can be stated as:

$$\forall t, \forall \delta t \in [0, \Delta t], \forall i \neq j \qquad (1)$$
$$v_h(t) = a_i, v_m(t + \delta t) = a_j \Rightarrow h \neq m.$$

Definition 3: A *path* $P(v_s, v_t) = v_s, \ldots, v_t$ is a sequence of adjacent nodes from v_s to v_t. The *path length* is the sum of all weights of the arcs connecting nodes of the path. The first i nodes of a path P are indicated with P_i.

Definition 4: A *Filter map* is a boolean function $b : E \rightarrow \{0, 1\}$ that returns false if arc e_{ij} cannot be used in a path-planning algorithm, true otherwise.
Filter maps will be used during the planning phase because arcs claimed by higher priority agents will need to be excluded by lower priority agents.

Definition 5: A *filtered graph* G^* is a subgraph of G where the filtered arcs have been removed by the filter map.

Time step discretization: Proposition 1 states that Δt is the time required by any agent to cross a node (the total time of node occupancy) before another agent can occupy it without colliding. This naturally leads to create a discrete TEN with a time steps discretization $T = \Delta t = D_s \frac{1+\cos\alpha}{\sin\alpha}\frac{1}{v_{min}}$. In this case the set $\mathcal{T}$ representing the time steps is $\mathcal{T} = \{0, T, 2T, \ldots, T(M-1)\}$.

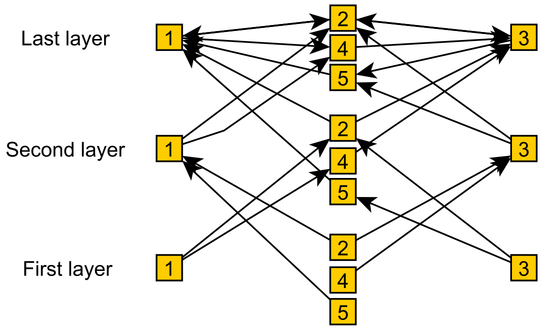

Fig. 5. Normalized TEN graph

A. The Normalized Augmented TEN

In the proposed approach we require agents to start occupying a node of the TEN only at time steps $\mathcal{T}$ and we also want agents to occupy a node at each time step $\mathcal{T}$. To model this requirement we have to cope with two different aspects: arcs possibly have different length and agents may move at different speeds. The proposed approach solves this problem with a normalization and a subsequent augmentation of the TEN.

Referring to the graph G_0 and its associated TEN represented in Fig. 2 and 3 respectively, we notice that the arc $(1, 3)$ in G_0 is represented by a double layer arc in the TEN since an agent requires double time to cross those arcs at the same speed. Hence, after a time step, an agent may still lay in the middle of an arc (as in the case of arc $(1, 3)$). To ensure that at each time step each agent occupies a node, we *normalize* the TEN by splitting l-layer arcs introducing $l - 1$ virtual nodes. We denote with L_{basic} the normalized arcs length.

Referring to graph G_0 of Fig. 2 nodes v_4 and v_5 are introduced to split direct arcs $(1, 3)$ and $(3, 1)$ respectively. The associated normalized TEN is reported in Fig. 5.

With the normalization procedure, the TEN has arcs with equal travel times. In order to allow agents to travel at different speeds we *augment* the network by inserting again l-layer arcs with a different meaning: an l-layer arc of the normalized and augmented TEN corresponds to an arc of length L_{basic} of the normalized TEN traveled at speed $\frac{v_{max}}{l}$.

Notice that, in the newly obtained normalized and augmented TEN (NA–TEN), for any l-layer arc from (v_i, t) to $(v_j, t + l)$ there exist l different q-layer arcs from (v_i, t) to $(v_j, t + q)$ with $q = 1, \ldots, l$. On the graph G_0 this implies that arc (v_i, v_j) can be traveled by l agents at different speeds, i.e. the maximum value scaled by $q = 1, \ldots, l$. Hence the number l_{max} is environment-dependent and represents the maximum number of agents that can cross an arc. Since we have that all normalized arcs are L_{basic} long, we set the parameter l_{max} so that $l_{max} \leq L_{basic}/D_s$.

In Fig. 6 the final NA–TEN, of G_0 in Fig. 2, is reported.

a) Arc Weights: In the NA–TEN we choose arcs weights in order to penalize multi-layer arcs that are slower than single layer ones by increasing their weights. The weight of a l-layer arc is $l\beta \; \forall l \in \{1, l_{max}\}$, where $\beta \geq 1$ is a penalization factor so that an agent will try to move at the

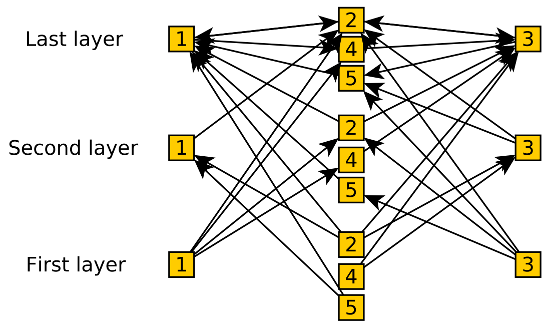

Fig. 6. Normalized and augmented TEN of the graph G_0 of Fig. 2

fastest speed in low traffic arcs instead of slowing down in a crowded arc. Finally, arcs weights of layer $M-1$ are set sufficiently high to ensure that any vehicle reaches its target at lower layers if possible.

IV. The DDISTRA Algorithm

In this section we will describe the distributed coordination algorithm on the NA–TEN named DDISTRA algorithm. The protocol will guarantee that agents negotiate a mutual exclusion access of nodes (or *resources*) while avoiding collisions. Once the access to a resource is obtained by an agent the resource is said to be *locked* by it.

We now describe the algorithm from the point of view of agent $\bar{a}$ with its unique identification number $UID_{\bar{a}}$, that will be used also in the following as a unique priority value.

We assume that at the beginning of the algorithm, all agents are occupying a node. Since the algorithm is based on agents coordination, we first describe the information exchanged by agents to obtain a mutual exclusion access of a resource, i.e. to lock it. The information is structured in a t-tuple as follows:

- the UID of the agent UID_{a_j} (that will be denoted by a_j itself);
- a set of K nodes $V(a_j) = \{(v_{j_1}, t+1), \ldots, (v_{i_K}, t+f)\}$ where f is the number of future time steps transmitted.

We will denote the information packet (*Locking Resource Packet*) with $LRP(a_j, V(a_j))$.

Notice that, since there are multi-layer arcs, $f \geq K$ where $f = K$ only in case of transmitted single-layer arcs. The number f is environment-dependent and represents the temporal length of the desired path. A large f will result in a long locked path with possibly few path changes during the algorithm execution. Moreover, a large f may require more negotiation steps to obtain the mutual exclusion access. On the other hand, a small f allows for more responsiveness (more possible path changes) and requires less negotiation steps. On the other hand, the number K of nodes in $V(\bar{a})$ is a parameter to be tuned considering that KL_{basic} is the maximum distance at which the agents communicate.

A. Negotiation phase

The algorithm consists of a negotiation phase during which agents exchange information and re-plan their motion to obtain a mutual exclusion access to desired nodes as described next. The negotiation happens while agents are moving and once it successfully ends, agents possibly have a new desired node and head towards it.

The negotiation phase takes place at every time-step $t \in \mathcal{T}$ and consists of $\tilde{N}+1$ negotiating steps of time $\delta t << T$, where $\tilde{N}$ is the number of neighbours (known upon communication) and δt is such that $(\tilde{N}+1)\delta t < T$. If the communication channel or the computational power of the agents cannot satisfy this condition, agents cannot react fast enough to environmental changes, thus T should be incremented accordingly (i.e. v_{max} should be decremented).

For each negotiating step, a classical Dijkstra algorithm is computed to produce a shortest path P on a possibly filtered NA–TEN from the current node $(v_i(\bar{a}) \in G_0)$ to the desired one. The filter is based on the LPR packets received by $\bar{a}$ from other agents with higher priorities, i.e. $a_j > \bar{a}$. Indeed, the nodes in $V(a_j)$ would like to be locked by higher priorities agents and then discarded by $\bar{a}$ through the Filter Map, see Definition 4.

Given P, $\bar{a}$ tries to lock all the resources $V(\bar{a}) = \{(v,t) \in P | t < f\}$ by sending an $LRP(\bar{a}, V(\bar{a}))$ to neighbours. Then the agent waits for δt seconds to receive the LRPs from its $\tilde{N}$ neighbours.

The lock of the resources is *successful* if for any received $LRP(a_j, V_S(a_j))$ with $V(a_j) \cap V(\bar{a}) \neq \emptyset$ it holds $a_j < \bar{a}$, i.e. all the resources of interest for $\bar{a}$ are of interest of only lower priority agents. Otherwise, $\bar{a}$ starts another negotiating step by re-executing Dijkstra on the newly filtered graph in which the nodes requested by agents with higher priorities have been deleted together with their incoming arcs.

When Dijkstra algorithm reports a failure, agent $\bar{a}$ is not able to plan a path. This emergency situation occurs in case of traffic jams and is treated with an emergency behaviour (see IV-B). If after $\tilde{N}$ negotiating steps no emergency situation has occurred the negotiation phase is said to be ended successfully.

Remark 2: After a successful negotiation phase, all agents are able to move and currently locked nodes of layer 0 can be released (unlocked). Consider the situation in which there more than one agent trying to go toward the same planar node $v_i \in G_0$ at the same time t. Since the negotiation phase is successful each vehicle managed to lock one resource $(v_i, t + kT)$ with a different value $k \leq m$ for each agent. Hence, each agent has a mutual exclusion access to a new resource and can release the level 0 resource locked.

After a successful negotiation, the agent $\bar{a}$ changes its direction (if required) and moves toward its first locked node. To ensure that the time requirement of reaching the nodes at time steps $\mathcal{T}$ is respected, the agent sets its speed accordingly.

It may occur that, after negotiation, two agents would cross the same planar arc at different speeds, with the faster agent that overtakes the slower one. Since our arcs represent one road large enough for only one agent, an overtaking implies a collision and hence it must be avoided. More formally

Definition 6: An *overtaking* on a planar arc $(v_i, v_j) \in E_0$ between agents $\bar{a}$ and $\hat{a}$ occurs when $\exists i, j, t_0 < t_1 < t_2 < t_3 : (v_i, t_0) \in P_{\bar{a}}, (v_j, t_3) \in P_{\bar{a}}$ while $(v_i, t_1) \in P_{\hat{a}}, (v_j, t_2) \in P_{\hat{a}}$

To avoid overtakings and hence collision, at the last step of the negotiation phase, nodes (v_j, t_2) and (v_j, t_3) are swapped between $P_{\hat{a}}$ and $P_{\bar{a}}$.

B. Emergency Situation

Whenever an agent $\bar{a}$ fails to find a path, an emergency situations occurs (usually in case of traffic jams). The agent hence sets its speed to zero and sends to all its neighbours an emergency packet containing the identification number of the planar node $v(\bar{a})$ that is:

- the currently occupied node, or
- the tail node of the arc that $\bar{a}$ is crossing.

Suppose now that $\bar{a}$ sends an emergency packet. Each agent that receives the packet and that is moving, in the NA–TEN, toward a node of type $(v(\bar{a}), k)$, with $k \in \mathcal{T}$ (i.e. is going to collide with $\bar{a}$) sets its speed to zero. Then it sends an emergency packet itself so that the emergency situation is propagated backward and no agent will collide.

Since agents that were not involved in the emergency situation will continue to move, at any time step $k \in \mathcal{T}$ it is possible for agents that stopped their motion to be able to start moving again, once the jam has been solved as it typically occurs on real traffic networks.

V. DDISTRA properties: Collisions and deadlocks

Theorem 1: Based on assumptions 1 and 2, the proposed DDISTRA algorithm ensures a collision free agents evolution.

Proof: Since, during the execution of the algorithm, agents can either be occupying a planar node or crossing a planar arc (an agent is said to be on an arc only if it does not occupy neither its head nor its tail nodes), based on Definition 2, we need to prove that no collision occurs between two agents in all possible configurations on the planar graph as described next.

The case in which two agents occupy node v may occur only if during the negotiation phase $(v,\, t_1)$ and $(v,\, t_2)$ have been assigned to the agents. Also in the worst case, i.e. $t_2 = t_1 \pm T$, Proposition 1 ensures the absence of collisions since $T = \Delta t$.

In case of agents on different nodes or arcs, collision avoidance is ensured by assumptions 1 and 2.

Consider now the case of one agent on a node and the other on an incoming/outgoing arcs of that node. The minimum distance between an agent on a node and the agent on an incoming/outgoing arcs of that node is $T\, v_{min}$. Indeed, DDiSTRA guarantees a time step separation T (see Remark 2) and the minimum allowed speed is v_{min}. By applying Proposition 1 in case of $v_{max} = v_{min}$, the absence of collisions is ensured (indeed T depends only on v_{min}).

If two agents are on the same arc, the last step of negotiation ensures that no overtaking takes place on arcs. Agents may only travel on the same arc with the same speed or with speeds that increase their mutual distance.

Finally, in case of an emergency, the speeds of all the agents that may incur in a collision with the agent in the emergency situation are set to zero and no collision occurs. ■

Definition 7: A *deadlock* is defined in [18] as a state in which all the following conditions are satisfied:

1) Agents get exclusive lock of the resources they require ("mutual exclusion" condition).
2) Agents hold resources already locked to them while waiting for additional resources ("wait for" condition).
3) Resources cannot be forcibly unlocked from the agents holding them until the resources are used to completion ("no preemption" condition).
4) A circular chain of agents exists, such that each agent locks one or more resources that are being requested by the next agent in the chain ("circular wait" condition).

Theorem 2: The DDISTRA policy is deadlock free if no emergency situation occurs. Otherwise, in case of emergency situation occurrence, let C_n be the dimension of the smallest cycle in G_0^* (normalized planar graph) and m the number of agents in the environment, there always exists one agent that is able to move if $(l_{max} + 1)C_n > m$.

Proof: In absence of emergency situations, since the proposed NA–TEN is an acyclic graph *along the time dimension* (see Section II) and since agents claim resources from the NA–TEN, we have that a circular chain of agents can not exists. So the "circular wait" condition of Definition 7 is never satisfied and hence no deadlock arises.

If there is an emergency situation, we need to prove that a circular wait is not possible even in the planar graph (since an agent in emergency mode locks the node for every possible layer in the NA–TEN graph). Consider the smallest cycle C in G_0^* (normalized planar graph) consisting of C_n nodes and C_n arcs. Consider the worst case scenario in which an agent in the cycle C (on nodes or on arcs) incurs in an emergency situation. The maximum number of agents that can be conveyed in the cycle is l_{max} for any arc plus C_n (agents may occupy a node o lay on an arc). If $(l_{max} + 1)C_n > m$ there always exists at least a node on the NA–TEN that is not occupied and hence a node toward which at least one agent can move. Hence, under the given hypothesis no deadlock may occur. ■

Notice that there exist less restrictive conditions under which Theorem 2 is still valid. For space limitations a simpler condition is provided for reader convenience. However, another condition is:

- $m < (l_{max} + 1)\, N$, where N is still defined as the total number of nodes in the planar graph, and
- the planar graph is fully connected.

Intuitively, these conditions describe the worst case scenario in which every arc is crowded by vehicles (with minimum security distance separation), except for one available slot on an arc (or node). But also in this case, there will always be at least one vehicle that is able to move.

VI. Simulation results

The DDISTRA algorithm has been applied to different application scenarios. We will now describe the results obtained in several simulation. To view the original videos please go to the AscariSimulator channel on http://www.youtube.com/user/ASCARIsimulator/videos?flow=grid where also a simulation of another scenario, not described for space limitations, can be found.

A. Continuous X-cross circuit

This scenario shows the synchronization capabilities of agents and is reported in Fig. 7. Eight agents are divided

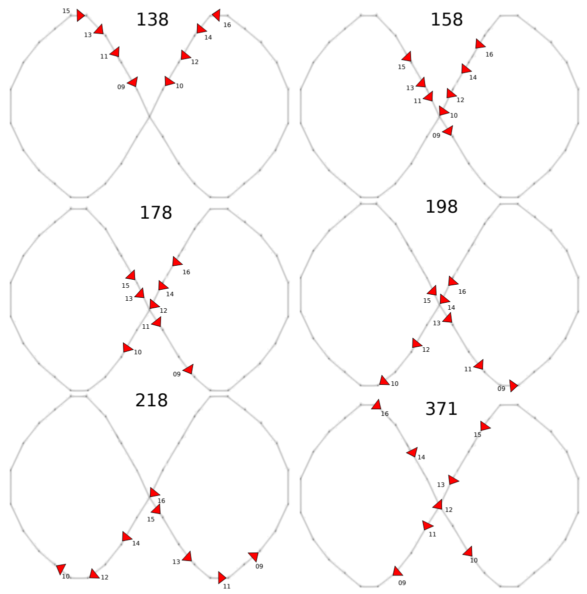

Fig. 7. Evolution of 6 vehicles in the X-cross scenario.

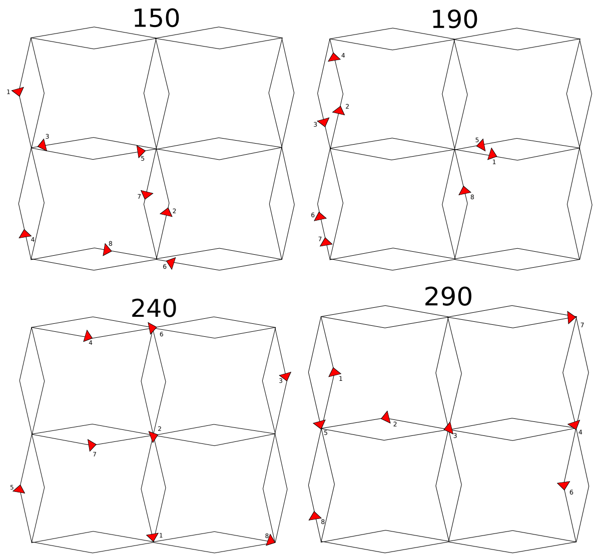

Fig. 8. Snapshot of the evolution of 6 vehicles in a simplified warehouse scenario.

into two groups, each running towards the same x-cross, i.e. we have one node and two arcs. Each agent in each group maintains the safety distance with the other. When an agent reaches the cross, it slows down to let one agent of the other group crossing first and so on. After the crossing the speed can be set again to the maximum one.

The result is that the safety distance among agents in the same group is doubled, so that for the next group crossing there is no need to slow down.

B. Warehouse simplified scenario

In this example a simplified warehouse is chosen as environment: the graph represents a warehouse which has 3 east-west and 3 north-south routes, with a total of 9 nodes in G_0, see Fig.8. Six agents move in the warehouse with random targets, during evolution collision are avoided and no vehicle enters in the emergency situation.

VII. Conclusions

A new approach to graph modeling oriented to the distributed coordination of multi-agent systems has been proposed. The approach is proved to be deadlock free and it enables agents to set both their speed and path to avoid other neighbours during their motion. The conditions that ensure the system to be livelock free are still under investigation.

Based on a shortest path algorithm applied to a custom TEN graph, DDiSTRA algorithm aim at reducing the power consumption of robots by decreasing braking and accelerations. Furthermore, it increases graph arcs flow by redirecting the agents to lower traffic paths.

References

[1] S. M. LaValle, “Planning Algorithms,” 2006.

[2] S. Morinaka, T. Nishi, M. Konishi, and J. Imai, “A distributed routing method for multiple agvs for motion delay disturbances,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS*, 2005, pp. 1986–1991.

[3] Y. Guo and L. Parker, “A distributed and optimal motion planning approach for multiple mobile robots,” in *Proceedings IEEE International Conference on Robotics and Automation*, vol. 3, 2002, pp. 2612–2619.

[4] Y. Ikemoto, Y. Hasegawa, T. Fukuda, and K. Matsuda, “Zipping, weaving: control of vehicle group behavior in non-signalized intersection,” in *Proceedings IEEE International Conference on Robotics and Automation*, vol. 5, 2004, pp. 4387–4391.

[5] W. Koh and S. Zhou, “An extensible collision avoidance model for realistic self-driven autonomous agents,” in *IEEE International SymposiumDistributed Simulation and Real-Time Applications*, 2007, pp. 7–14.

[6] S. A. Reveliotis and E. Roszkowska, “Conflict resolution in free-ranging multivehicle systems: A resource allocation paradigm,” *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 283–296, 2011.

[7] E. Roszkowska and S. A. Reveliotis, “A distributed protocol for motion coordination in free-range vehicular systems,” *Automatica*, vol. 49, no. 6, pp. 1639–1653, 2013.

[8] K. M. Krishna and H. Hexmoor, “Reactive collision avoidance of multiple moving agents by cooperation and conflict propagation,” in *Proceedings IEEE International Conference on Robotics and Automation*, vol. 3, 2004, pp. 2141–2146.

[9] E. Ericsson, “Independent driving pattern factors and their influence on fuel-use and exhaust emission factors,” *Transportation Research Part D: Transport and Environment*, vol. 6, no. 5, pp. 325 – 345, 2001.

[10] G. Santos, H. Behrendt, and A. Teytelboym, “Part ii: Policy instruments for sustainable road transport,” *Research in Transportation Economics*, vol. 28, no. 1, pp. 46–91, 2010.

[11] M. S. Young, S. A. Birrell, and N. A. Stanton, “Safe driving in a green world: A review of driver performance benchmarks and technologies to support “smart” driving,” *Applied ergonomics*, vol. 42, no. 4, pp. 533–539, 2011.

[12] I. Chabini and S. Lan, “Adaptations of the a* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 60–74, 2002.

[13] S. Pallottino and M. G. Scutella, “Shortest Path Algorithms in Transportation models : classical and innovative aspects,” 1997.

[14] B. C. Dean, “Algorithms for Minimum-Cost Paths in Time-Dependent Networks with Waiting Policies,” 2004.

[15] J.-f. Berube, J.-y. Potvin, and J. Vaucher, “Time-dependent shortest paths through a fixed sequence of nodes : application to a travel planning problem,” vol. 33, pp. 1838–1856, 2006.

[16] X. Cai, T. Kloks, and C. K. Wong, “Time-Varying Shortest Path Problems with Constraints,” *Networks*, vol. 29, no. 3, pp. 141–150, 1997.

[17] J. Yu and S. M. LaValle, “Time optimal multi-agent path planning on graphs,” in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[18] E. G. Coffman, M. Elphick, and A. Shoshani, “System deadlocks,” *ACM Computing Surveys (CSUR)*, vol. 3, no. 2, pp. 67–78, 1971.