Multi-agent trajectory planning: A decentralized iterative algorithm based on single-agent dynamic RRT*

Paolo Verbari, Luca Bascetta, Maria Prandini

***Abstract*— This paper addresses trajectory planning in a multi-agent cooperative setting, where n agents are moving in the same region and need to coordinate so as to maintain a certain pairwise safety distance, while avoiding obstacles. We introduce a decentralized strategy that is based on an iterative (re)plan-compare-assign process. The key features of the proposed strategy are that coordination is obtained via the compare-assign phase in at most n iterations (including the initialization), and (re)planning is performed by the agents using a single-agent planner, considering the tentative trajectories of the others fixed, and without sharing with them their tracking capabilities and adopted cost criterion. In the proposed implementation, each agent uses a dynamic Rapidly exploring Random Tree star (RRT*) planner that integrates a new prune and graft feature to avoid rebuilding a new tree from its root each time replanning is needed. The resulting Multi-RRT* algorithm is tested in 2D scenarios and shows promising results.**

I. INTRODUCTION

Trajectory planning in a multi-agent system consists in designing the trajectories of multiple agents that are moving in the same region of the space, from their origin to their destination positions, while minimizing some cost like, e.g., their travel time. Multi-agent trajectory planning problems are encountered in robotics when coordinating a group of robots or aerial vehicles sharing the same environment (see e.g., [1], [2], [3]), and also in transportation infrastructures like air transportation [4], [5], at different time scales (short, mid and long-term). Besides avoiding obstacles, the agents have typically to maintain a minimum safety distance from each other, so as to avoid conflicts. This makes the trajectory planning problem coupled, and calls for a cooperative solution involving all the agents.

In this work, we introduce a decentralized iterative solution that distributes the computational load between all agents, preserves privacy of their local information related, e.g., to their tracking capabilities, requires a limited amount of information broadcasting, and implements a fairness criterion. Initially, every agent neglects the others and plans its optimal trajectory. All planned trajectories are broadcast, and each agent that is involved in a conflict replans its own trajectory separately, treating other agents as moving obstacles. To this purpose, agents adopt a dynamic version of the Rapidly exploring Random Tree star (RRT*) planner [6], [7], which is enhanced here with a prune and graft feature to avoid the time consuming operation of rebuilding the tree from its root at each iteration, similarly to the approach in [8] where dynamic RRT is proposed. Performance degradation of all the replanning agents is broadcast, and the agent with minimal performance degradation has to adopt its replanned trajectory and broadcast it to the others. In the following iteration, agents that are involved in a conflict replan their own trajectory, broadcast their performance degradation, and so on. This replan-compare-assign scheme is repeated till all trajectories are conflict-free.

The authors are with Politecnico di Milano, Piazza Leonardo da Vinci 32 – 20133 Milano, Italy paolo.verbari@mail.polimi.it, {luca.bascetta, maria.prandini}@polimi.it

Multi-agent (MA) versions of RRT* have been proposed in the literature, e.g., in [9], [10]. However, multi-agent planning is performed centrally, which requires all information on the agents actuation capabilities, origin and destination, to be available to a single central unit that solves the planning problem for all agents. Privacy and computational issues may arise. In particular, the MA-RRT* algorithm introduced in [9] searches for a conflict-free multi-agent trajectory directly in the joint state-space of all the agents, whose dimension grows linearly with the number of agents (the sample space size grows exponentially!). On the contrary, in our algorithm, RRT* is run by each agent, based on the knowledge of its own actuation capabilities, on a state space whose dimension does not depend on the number n of agents involved. This comes at the price of repeatedly solving a planning problem till convergence. The number of iterations for conflict resolution is upper bounded by $n-1$ so that an agent might have to run RRT* n times in the worst case, including the initialization. However, each single instance of RRT* is computationally much less intensive.

In [10], agents are sorted according to an a-priori agreed priority criterion, and trajectories are planned in sequence by the central unit via a variant of the RRT* algorithm, starting from the trajectory of the highest priority agent and moving to the lowest-priority one, each time treating the trajectories that have already been designed as obstacles to avoid. Decentralized versions of multi-agent prioritized planning have been introduced where each robot computes its own trajectory through either synchronous [11] or asynchronous [12] iterations. The ordering must still be agreed a-priori.

In [13] a variant of the standard decoupled approach to multi-robot motion planning [14] is proposed, where a central unit first computes a trajectory for each agent neglecting the others and then coordinates the motion of the agents to avoid collisions. In order to reduce the dimension of the configuration space on which planning is performed, [15] decouples a multi-agent path planning problem into subproblems whose solutions can be determined sequentially. In a similar vein, [16] adopts dynamic subdimensional expansion in probabilistic planners by initially planning in the configuration

space of each agent, separately, and coupling the spaces of multiple agents only when they get close with one another. A parallelization scheme for sampling-based motion planning algorithms including RRT* is proposed in [17] to speed them up. A hybrid control architecture integrating parallel problem solving is adopted in the decentralized approach in [18].

Admittedly, our approach presents many features in common with previous works. However, these features have been demonstrated separately in the literature and our goal here is to fully exploit and integrate them so as to obtain a computationally efficient, scalable solution that preserves privacy by combining a decentralized negotiation scheme with single-agent dynamic RRT* in the Multi-RRT* algorithm.

The rest of the paper is organized as follows. In Section II, we present the multi-agent trajectory planning problem, which is reduced to multiple single-agent trajectory planning problems in the decentralized scheme proposed in Section III. The Multi-RRT* algorithm is formulated in Section IVand its performance is shown in Section V with reference to 2D scenarios. Finally, some concluding remarks are drawn in Section VI.

II. Problem formulation

We consider an n-agent system where each agent i is moving in some region P from an origin $\mathbf{p}_{s,i} \in P$ to a destination position $\mathbf{p}_{e,i} \neq \mathbf{p}_{s,i} \in P$. Fixed obstacles are possibly present in P and are represented as an open subset $P_{obs} \subset P$. Consequently, the free space is defined as $P_{free} := P \setminus P_{obs}$. All origins and destinations are assumed to belong to P_{free}.

Each agent dynamics is described via a differential equation of the form

$$\dot{\mathbf{q}}_i(t) = \mathbf{f}_i\left(\mathbf{q}_i(t), \mathbf{u}_i(t)\right), \qquad \mathbf{q}_i(0) = \mathbf{q}_{0,i}, \tag{1}$$

where $\mathbf{f}_i$ is continuously differentiable as a function of both arguments, $\mathbf{q}_i \in Q_i \subseteq \mathbb{R}^d$ and $\mathbf{u}_i \in U_i \subseteq \mathbb{R}^m$ are the state and control input of agent i, respectively, and $\mathbf{q}_{0,i}$ is the initial state. State $\mathbf{q}_i$ includes the position $\mathbf{p}_i$, and $\mathbf{p}_i(0) = \mathbf{p}_{s,i}$.

An obstacle-free trajectory $\mathbf{z}_i$ of agent i is defined as the tuple $\mathbf{z}_i = (\mathbf{q}_i(\cdot), \mathbf{u}_i(\cdot), \tau_i)$, where τ_i is the duration of the trajectory, and $\mathbf{q}_i(\cdot) : [0, \tau_i] \to Q_i$ and $\mathbf{u}_i(\cdot) : [0, \tau_i] \to U_i$ represent the state and control input evolution, satisfying the differential constraint (1) for $t \in [0, \tau_i]$, $\mathbf{q}_i(0) = \mathbf{q}_{0,i}$, $\mathbf{p}_i(\tau_i) = \mathbf{p}_{e,i}$, and $\mathbf{p}_i(t) \in P_{free}$, $t \in [0, \tau_i]$.

Trajectory $\mathbf{z}_i = (\mathbf{q}_i(\cdot), \mathbf{u}_i(\cdot), \tau_i)$ of agent i is rated according to some cost criterion $J_i(\mathbf{z}_i)$, which is non-zero for all trajectories joining two different positions. For instance,

$$J_i(\mathbf{z}_i) = \int_0^{\tau_i} g_i\left(\mathbf{q}_i(t), \mathbf{u}_i(t)\right) \mathrm{d}t, \tag{2}$$

where $g_i : Q_i \times U_i \to \mathbb{R}_{\geq 0}$ is an instantaneous cost function weighting time and/or control effort. If $g_i(\cdot, \cdot) = 1$, then, agent i is minimizing the time to reach its destination.

Trajectory $\mathbf{z}_i$ is said to be conflict-free if it is obstacle-free and also keeps agent i at a safety distance $d_S > 0$ from all the other agents, i.e., $\|\mathbf{p}_i(t) - \mathbf{p}_j(t)\| > d_S$, $t \in [0, \tau]$, $j = 1, \ldots, n$, $i \neq j$ and $\tau = \min(\tau_i, \tau_j)$[1]. The safety distance condition makes the problem coupled and calls for determining a joint trajectory $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ that is conflict-free and satisfactory in terms of cost for all the n agents.

Our goal is to provide a decentralized cooperative solution that distributes the computational workload between all agents, requires a reduced amount of information exchange and a reduced number of iterations, preserves privacy on the agents capabilities, and is fair and acceptable by all agents.

We look for a solution where throughout iterations each agent just needs to use a planner for the design of a trajectory that avoids obstacles which are either fixed or possibly time-varying (this is the case if obstacles represent the other agents) but according to a known trajectory. In the sequel, we shall refer to this planner as a *single-agent planner*

III. Decentralized cooperative resolution

We suppose that all agents are willing to cooperate so as to plan a joint conflict-free trajectory $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, that brings all of them from their origin to their destination while avoiding an excessive deterioration of their cost $J_i(\mathbf{z}_i)$ in (2), $i = 1, \ldots, n$, with respect to the cost that they would achieve by disregarding the safety separation constraint.

The proposed decentralized cooperative resolution scheme is iterative and starts with an initialization step, where every agent neglects the presence of all the other agents and computes an optimal obstacle-free trajectory $\mathbf{z}_i^\star$ with its onboard single-agent planner. In the following, we shall refer to the cost associated with the computed $\mathbf{z}_i^\star$ as $J_i^\star = J_i(\mathbf{z}_i^\star)$. This represents the best performance that agent i can achieve. The trajectory $\mathbf{z}_i^\star$ is broadcast to all agents so that they can check for possible conflicts. If a conflict is detected, then, the iterative conflict resolution process starts.

In the first iteration, agents that are involved in a conflict replan their own trajectory separately, designing a conflict-free optimal trajectory through their single-agent planner with all the other agents modelled as time varying obstacles that are moving according to the trajectory that they broadcast. Each one of these agents then broadcasts the deterioration of performance associated with its new trajectory. The agent with minimal performance degradation is assigned the task of contributing to conflict resolution by updating its trajectory and replacing it with the newly planned one. Its new conflict-free trajectory is broadcast to the other agents. In all the following steps of the decentralized resolution scheme, the agent is treated as a moving obstacle and its trajectory is not further modified.

This replan-compare-assign scheme is repeated through iterations till all trajectories are conflict-free. After at most $n-1$ iterations we obtain a conflict-free joint trajectory. In the worst case, an agent has to compute its trajectory n times, including the initialization phase.

Performance deterioration of agent i at iteration k is evaluated in percentage with respect to its minimum cost

[1]Agents contribute to the safety requirement up to the time they reach their destination. This allows to account for agents with the same destination, like aircraft landing at the same airport.

as follows:

$$D_i^{(k)} = \frac{J_i(\mathbf{z}_i^{(k)}) - J_i^\star}{J_i^\star} \cdot 100, \tag{3}$$

where $\mathbf{z}_i^{(k)}$ denotes the trajectory computed by agent i at iteration k. This allows to account for the fact that costs are different possibly due to a larger distance to travel, limited actuation capabilities, etc. Also it allows for different cost criteria to be adopted by different agents.

The proposed algorithm has the following key features:

- agents do not share private information regarding their actuation capabilities coded through the input set U_i, $i = 1, \ldots, n$, and their cost function J_i;
- each agent has to be able to perform only trajectory planning with obstacle avoidance, since multi-agent coordination is achieved via the compare-assign procedure based on the performance degradation indices. Any obstacle-free trajectory planner which accounts for optimality can be adopted, possibly also different ones among agents with different computation capabilities;
- except for the initialization step, at the following iterations agents broadcast only a scalar quantity (their performance degradation), and just the agent with the lowest performance degradation has to transmit its replanned trajectory;
- the algorithm ends in a finite number of iterations, which is upper bounded by $n-1$, since at every iteration the number of agents that are possibly involved in a conflict decreases by at least 1, and it is initially at most n.

Note that the proposed approach is effective if indeed each agent can compute a conflict-free trajectory without requiring the other agents to change their own trajectory. This is the case for ground robots that can stop and then restart. As for aerial robots, stop-and-restart solutions are not feasible, but solutions where they take some detour and then continue towards their destination could be implemented.

The implemented priority ordering method is aiming at imposing some degree of fairness in the solution, while reducing the impact of conflict resolution on the overall multi-agent system performance. As a matter of fact, the choice of which agent should be in charge of conflict resolution is made sorting them based on performance, instead of using an a-priori defined order. The resulting multi-agent joint trajectory is not guaranteed to be Pareto optimal. However, this is the price to pay for the simplicity of the scheme.

Agents have to agree on the coding of the trajectory information to be broadcast. Since trajectory information is used for checking the safety distance and for characterizing moving obstacles to be avoided, each trajectory can be coded as a list of timed waypoints in the p_i component of the state q_i. This list is transmitted and is then interpolated by each agent with linear segments travelled at constant speed to derive the moving obstacle description that is fed into its own single-agent planner. If the time distance between consecutive waypoints is sufficiently small, then, the resulting piecewise linear trajectory is an accurate estimate of the actual trajectory. The safety distance could be eventually slightly increased when modelling the agent as a moving obstacle to account for the introduced estimation error.

IV. MULTI-RRT*

In this section, we present the Multi-RRT* implementation of the proposed decentralized cooperative resolution scheme. Like in the multi-agent trajectory planner [10], the Rapidly exploring Random Tree star (RRT*) algorithm is used as a single-agent planner for the agents to recompute at each iteration their trajectory while accounting for other agents as moving obstacles. However, here the designed conflict-free joint trajectory is the result of a single-agent replanning that is performed according to an order that is not a-priori agreed, but is defined through iterations based on performance deterioration so as to enforce fairness in the obtained solution.

RRT* belongs to the family of sampling-based planners that were introduced to handle systems with high dimensional state spaces [6], [7], and originates from RRT. In RRT, a roadmap in the form of a tree of feasible trajectories is built by starting from the initial state as root of the tree and then sequentially sampling a state (node) at random in the obstacle-free state space, and connecting it to a neighbouring node in the tree with a trajectory (edge) that is optimal according to the chosen cost criterion and compatible with the agent's constraints, till a neighbour of the destination position is reached. A collision check module allows to determine the feasibility of a tentative trajectory with respect to the obstacle-free requirement.

RRT satisfies the probabilistic completeness property, i.e., it returns a feasible solution with a probability converging to one as the number of samples grows to infinity, if such a solution exists, but does not provide any performance guarantee because nodes are not necessarily connected to the tree optimally. This requires to possibly rewire the tree every time a new node is sampled by testing connections with pre-existing nodes that are in a suitably defined neighborhood. This is the additional key feature introduced in RRT*, which is in fact asymptotically optimal, i.e., the probability of finding an optimal solution, if there exists one, converges to 1 as the tree cardinality grows to infinity, [6]. Evidently, the computational complexity of RRT* increases as the number of nodes in the tree grows. If the number of nodes is lower, a lower performing solution is found. One can set the number of nodes as a compromise between performance and computing time.

In the proposed Multi-RRT* algorithm, at every iteration all the agents that are involved in a collision (either with a fix or a moving obstacle representing an agent) have to run RRT*, which can be computationally intensive especially if a large number of nodes is used. In order to alleviate the computational load caused by multiple runs of RRT*, we propose a solution where each agent builds the complete tree only once, at the initialization step, and, then, at each of the following iterations, it uses the tree computed at the previous iteration and applies a prune and graft procedure.

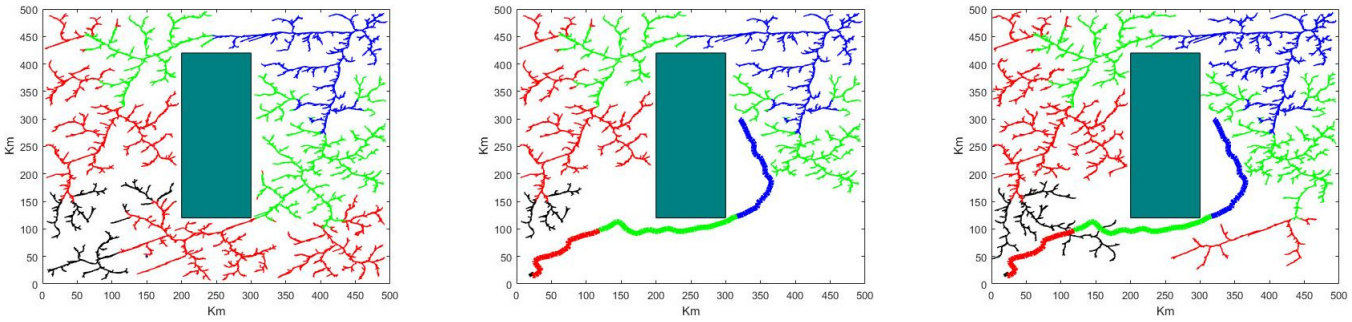

Fig. 1. Example of a tree avoiding a fixed rectangular obstacle (left plot): branches that are travelled within the time intervals [0, 15], (15, 30], (30, 45], are coloured in blue, green, red, respectively, and those travelled at a time larger than 45 are coloured in black. Time is measured in minutes, space in kilometres. The middle plot represents the tree pruned from branches where a conflict occurs with a moving obstacle (thick line). The pruned tree is then grafted with collision-free edges as shown in the right plot.

This has the positive effect of saving time, memory, and avoiding repeating the same computations. More precisely, when an agent detects a collision with an obstacle, it prunes its own tree removing the branches that are involved in the collision and then starts regrowing the tree by grafting it with new edges. This is shown pictorially in Figure 1. In the left plot we can see a tree that is built to avoid the fixed rectangular obstacle with edges travelled in the same time slot coded with the same color. When a moving obstacle representing an agent is added, those branches where the two agents are closer than the safety distance are pruned, as shown in the center plot. The pruned tree is then regrown with collision-free edges as shown in the right plot.

Note that agents run RRT* also at the initialization stage, so as to perform avoidance of the fix obstacle and build the tree that will then be pruned and grafted if needed to avoid the other agents.

V. NUMERICAL EXAMPLES

In this section, we report some simulation results to the purpose of showing the performance of Multi-RRT*. Two configurations referring to a 2-dimensional region will be presented, one with 3 agents and the other one with 5 agents. In both examples, we consider as cost function for all agents the time to reach their destination.

For simplicity, in our simulations we assume that each agent is travelling at constant speed and simplify its dynamic by considering as steering function for building the RRT* tree a simple function that connects two consecutive points through a line segment and avoids too sharp turning between consecutive segments. This approximation is useful to get a fast execution of the RRT* planner, that is fundamental within Multi-RRT*, since RRT* is run multiple times. To make the piecewise linear planned trajectories joining consecutive points smoother, one can adopt a cubic spline interpolation of those points, and account for the introduced error by incrementing the fixed obstacle size and the safety distance of some small amount. More complex steering functions accounting for more complex dynamics and constraints (see, e.g., [19] and [20]) could be implemented without modifying the decentralized cooperative scheme, due to the modular structure of Multi-RRT*.

In our examples, we further assume that all agents can broadcast and receive information without transmission errors. Each agent will communicate to the others the ideal trajectory that it would like to follow, its performance degradation index (3) in case its ideal trajectory is not conflict-free, and finally, if its performance degradation is found out to be the lowest, its replanned trajectory.

Three-agent system example

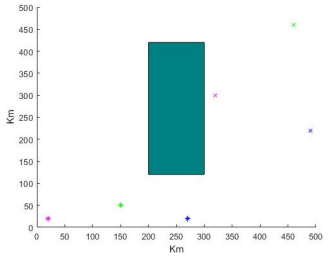

Fig. 2. 3-agent system: configuration – fixed rectangular obstacle in the middle, origin (x) and destination ($\star$) of agent 1 (green), agent 2 (blue), and agent 3 (magenta).

The considered configuration involves 3 agents and is reported in Figure 2, where each agent is assigned a color: agent 1 is green, agent 2 is blue, and agent 3 is magenta. The starting positions are marked with an "x" sign, and the destination positions are marked with a "$\star$" sign. The rectangular shape in the center of the figure is a fixed obstacle that must be avoided.

The cardinality of the tree in the RRT* planner adopted by each agent is 11000. New samples are extracted according to the uniform distribution in a square region of size 500 kilometers. The speed of each agent, assumed to be constant, is 15.216 km/min. The maximum steering angle of all the agents is $\pi/4$. The initial heading angle of both agent 1 and agent 2 is set equal to $\pi/2$, while that of agent 3 is set equal to $-\pi/2$. The safety distance d_S is set equal to 10 kilometres.

We next describe the evolution of the Multi-RRT* algorithm throughout its iterations. We report the outcome of a representative run.

At the initialization step, each single agent computes its own ideal obstacle-free trajectory neglecting other agents

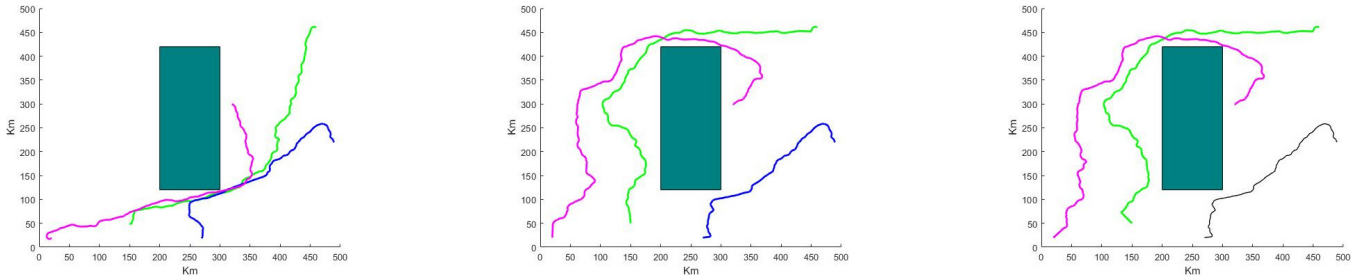

Fig. 3. 3-agent system: initialization (left plot): initially planned obstacle-free trajectories of the 3 agents computed neglecting the others (green for agent 1, blue for agent 2, and magenta for agent 3; first iteration (middle plot) – newly planned trajectories of the 3 agents (green for agent 1, blue for agent 2, and magenta for agent 3); second iteration (right plot) – newly planned trajectory of agent 1 (green) and agent 3 (magenta), and final trajectory of agent 2 (black).

(see the left plot of Figure 3), broadcasts it, and checks for possible conflicts with the trajectories received from the other 2 agents.

In this example, each agent predicts some conflict and then, at the first iteration of Multi-RRT*, replans its trajectory so as to avoid those received from the other 2 agents. The replanned tentative trajectories are reported in the middle plot of Figure 3. Given that agent 2 turns out to be the one with lowest performance degradation, its replanned tentative trajectory becomes its actual trajectory, and at the second iteration agents 1 and 3 provide new tentative trajectories as reported in the right plot of Figure 3, where the trajectory of agent 2 is plotted in black since it is its final one. Now it is agent 1 with the lowest performance degradation with respect to agent 3. Its tentative trajectory becomes its actual final one. Agent 3 can maintain its obstacle-free trajectory planned at the initialization stage and its performance degradation is then zero.

Table I reports the value in percentage of the performance degradation (see (3)) with respect to the optimal costs $J_i^\star$, $i = 1,2,3$, computed separately by each agent and shared with the others since a conflict is detected. At the first iteration all agents are involved in the conflict resolution process, and agent 2 has the lowest performance degradation with a cost increase of 0.32% with respect to $J_2^\star$. At the following iteration, performance degradation is not computed for agent 2 since its trajectory is set to be the one computed at iteration 1. At iteration 2, the lowest performance degradation is that of agent 1, which amounts to about 15% of $J_1^\star$, and agent 1 has then to follow its tentative trajectory computed at iteration 2.

TABLE I

3-AGENTS: PERFORMANCE DEGRADATION IN A RUN OF MULTI-RRT*.

iteration	1	2
agent 1	14.79	15.52
agent 2	0.32	–
agent 3	39.84	39.63

Figure 4 represents the distances between agent pairs as a function of time. In black the distance between agents 3 and 1, in green the distance between agents 2 and 3, and in blue the distance between agents 1 and 2. The horizontal red line represents the safety limit.

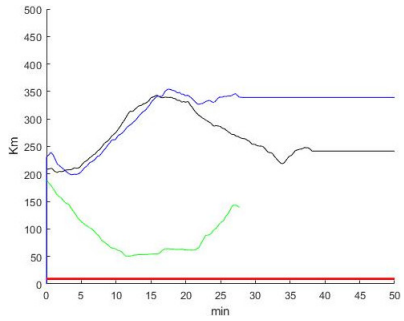

Fig. 4. 3-agent system: distance between the first and the third agent (black), the first and the second agent (blue), and the second and the third agent (green). The red line defines the safety distance.

Five-agent system example

The considered 5-agent system configuration is reported in Figure 5, together with the optimal trajectories computed by each agent neglecting the others in a representative Multi-RRT* run. Agent 1 is coded with green color, agent 2 with yellow, agent 3 with magenta, agent 4 with blue, and agent 5 with black. The starting positions are marked with an "x" sign, and the destination positions are marked with a "$\star$" sign. There is no fixed obstacle in this example.

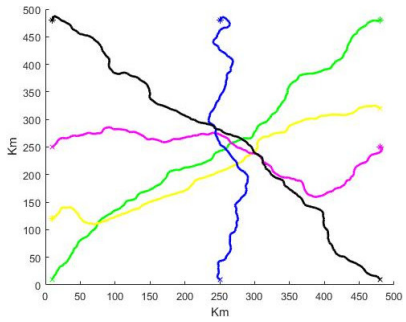

Fig. 5. 5-agent system: origins (x), destinations ($\star$), and optimal trajectories computed neglecting the other agents by agent 1 (green), agent 2 (yellow), agent 3 (magenta), agent 4 (blue), and agent 5 (black).

The cardinality of the tree in the RRT* planner adopted by each agent is 20000. New samples are extracted according

to the uniform distribution in a square region of size 500 kilometers. The (constant) speed of each agent is 15.216 km/min, the maximum steering angle of all the agents is $\pi/4$, and the safety distance is 10 kilometres, as in the previous example. The initial heading angle of agents 1, 3, and 4 is set equal to 0, and that of agents 2 and 5 is set equal to π.

Table II reports the performance degradation index (3) computed in percentage with respect to the values for the optimal costs $J_i^\star$, $i = 1,\ldots,5$, determined separately at the initialization step by agents $1,\ldots,5$ using RRT*. Finally, Figure 6 plots the joint conflict-free trajectories computed by Multi-RRT*.

TABLE II

5-AGENTS: PERFORMANCE DEGRADATION IN A RUN OF MULTI-RRT*.

iteration	1	2	3	4
agent 1	29.98	25.52	37.31	16.80
agent 2	22.69	19.38	–	–
agent 3	28.88	25.70	23.44	–
agent 4	48.07	23.90	41.31	32.37
agent 5	10.83	–	–	–

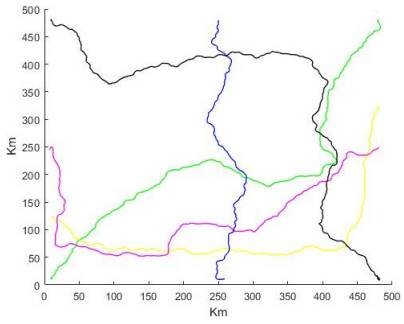

Fig. 6. 5-agent system: final set of conflict-free trajectories of agent 1 (green), agent 2 (yellow), agent 3 (magenta), agent 4 (blue), agent 5 (black).

VI. CONCLUSIONS

This paper addressed the problem of trajectory planning for multiple agents moving in the same region with obstacles. A cooperative decentralized scheme is proposed, where agents exchange information on their planned optimal obstacle-free trajectories, which are then possibly modified if the safety distance constraint is not satisfied through a suitable designed iterative resolution scheme. Convergence to a joint obstacle-free and safe trajectory is achieved in a finite number of iterations.

Trajectory planning for each single agent can be performed via a single-agent planner like the RRT* algorithm, which is able to account for optimality and constraints, and that can be implemented with an ad-hoc prune and graft procedure to avoid rebuilding the tree of feasible trajectories from its root every time a replanning is needed. The resulting Multi-RRT* algorithm has been tested in 2D scenarios, adopting a simplified implementation where kinematic and dynamic constraints are neglected and edges joining nodes are assumed to be linear trajectories travelled at constant speed.

Future work includes the implementation of Multi-RRT* in a 3D environment, using a more accurate steering function. Also, the approach could be extended to large scale multi-agent systems by appropriately clustering agents in non-interfering groups, each group running separately the proposed decentralized trajectory planning scheme so as to limit the communication broadcasting and number of iterations. This requires further investigations.

REFERENCES

[1] J. Yu and S. LaValle, "Optimal multi-robot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[2] J. Snape, S. Guy, J. Van Den Berg, and D. Manocha, "Smooth coordination and navigation for multiple differential-drive robots," in *Experimental Robotics, ser. Springer Tracts in Advanced Robotics*, O. Khatib, V. Kumar, and G. Sukhatme, Eds. Springer Berlin, 2014, vol. 79, pp. 601–613.

[3] J. Van Den Berg and M. Overmars, "Prioritized motion planning for multiple robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 430–435.

[4] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.

[5] J. Tang, "Review: Analysis and improvement of traffic alert and collision avoidance system," *IEEE Access*, vol. 5, pp. 21 419–21 429, 2017.

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[7] ——, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *IEEE Conference on Decision and Control*, 2010, pp. 7681–7687.

[8] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *IEEE International Conference on Robotics and Automation*, 2006, pp. 1243–1248.

[9] M. Čáp, P. Novák, J. Vokřínek, and M. Pěchouček, "Multi-agent RRT*: Sampling-based cooperative pathfinding," in *12th International Conference on Autonomous Agents and Multiagent Systems*, 2013, pp. 1263–1264.

[10] M. Ragaglia, M. Prandini, and L. Bascetta, "Multi-agent Poli-RRT*," in *Modelling and Simulation for Autonomous Systems*, ser. Lecture Notes in Computer Science, 2016.

[11] P. Velagapudi, K. Sycara, and P. Scerri, "Decentralized prioritized planning in large multirobot teams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4603–4609.

[12] M. Čáp, P. Novák, M. Selecky, J. Faigl, and J. Vokřínek, "Asynchronous decentralized prioritized planning for coordination in multi-robot system," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3822–3829.

[13] M. Saha and P. Isto, "Multi-robot motion planning by incremental coordination," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5960–5963.

[14] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[15] J. Van Den Berg, J. Snoeyink, M. Lin, and D. Manocha, "Centralized path planning for multiple robots: Optimal decoupling into sequential plans," *Robotics*, vol. 5, pp. 137–144, 2010.

[16] G. Wagner, M. Kang, and H. Choset, "Probabilistic path planning for multiple robots with subdimensional expansion," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 2886–2892.

[17] M. Otte and N. Correll, "C-FOREST: Parallel shortest path planning with superlinear speedup," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 798–806, 2013.

[18] K. Azarm and G. Schmidt, "Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation," in *International Conference on Robotics and Automation*, vol. 4, 1997, pp. 3526–3533.

[19] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[20] M. Ragaglia, M. Prandini, and L. Bascetta, "Poli-RRT*: Optimal RRT-based planning for constrained and feedback linearisable vehicle dynamics," in *European Control Conference*, 2015, pp. 2521–2526.