Auton Robot (2012) 32:385–403
DOI 10.1007/s10514-012-9275-2

Decentralized path planning for multi-agent teams with complex constraints

Vishnu R. Desaraju · Jonathan P. How

Received: 20 September 2011 / Accepted: 16 January 2012 / Published online: 2 February 2012

Abstract This paper presents a novel approach to address the challenge of planning paths for multi-agent systems subject to complex constraints. The technique, called the Decentralized Multi-Agent Rapidly-exploring Random Tree (DMA-RRT) algorithm, extends the Closed-loop RRT (CL-RRT) algorithm to handle multiple agents while retaining its ability to plan quickly. A core component of the DMA-RRT algorithm is a merit-based token passing coordination strategy that makes use of the tree of feasible trajectories grown in the CL-RRT algorithm to dynamically update the order in which agents replan. The reordering is based on a measure of each agent's incentive to change the plan and allows agents with a greater potential improvement to replan sooner, which is demonstrated to improve the team's overall performance compared to a traditional, scripted replan order. The main contribution of the work is a version of the algorithm, called Cooperative DMA-RRT, which introduces a cooperation strategy that allows an agent to modify its teammates' plans in order to select paths that reduce their combined cost. This modification further improves team performance and avoids certain common deadlock scenarios. The paths generated by both algorithms are proven to satisfy inter-agent constraints, such as collision avoidance, and numerous simulation and experimental results are presented to demonstrate their performance.

V.R. Desaraju (✉) · J.P. How
Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA
e-mail: rajeswar@alum.mit.edu

J.P. How
e-mail: jhow@mit.edu

Keywords RRT · Collision avoidance · Cooperative planning

1 Introduction

As autonomous systems are deployed in increasingly complex scenarios, it is essential for a team of agents to be able to work in parallel and perform far more complicated tasks than an agent operating alone. Consider, for example, an automated warehouse as shown in Fig. 1. The overall objective for the team of autonomous agents operating in this warehouse is to manage the flow of inventory efficiently, and agents are assigned tasks such as taking inventory and moving items. However, as the figure illustrates, agents must also avoid collisions with objects in the environment and other agents while executing these tasks. This is particularly difficult in unstructured and minimally-prepared environments, such as a supply depot on a forward operating base, where agents cannot rely on guidance infrastructure or roadmaps and must plan paths in real time (Teller et al. 2010).

This highlights a well-known but essential requirement for virtually any multi-agent system: agents must be able to navigate safely through the environment in which the tasks are to be performed (Parker 2002; Leonard et al. 2007; Murray 2007). In practice, agents are subject to actuation limits and constrained dynamics, such as a minimum turn radius, which can restrict their mobility. These limitations as especially important for teams deployed in crowded environments, including indoor settings. Consequently, path planners that are developed for such environments must explicitly account for these constraints in order to guarantee safe navigation (Frazzoli et al. 2002; Purwin et al. 2008; Aoude et al. 2010).

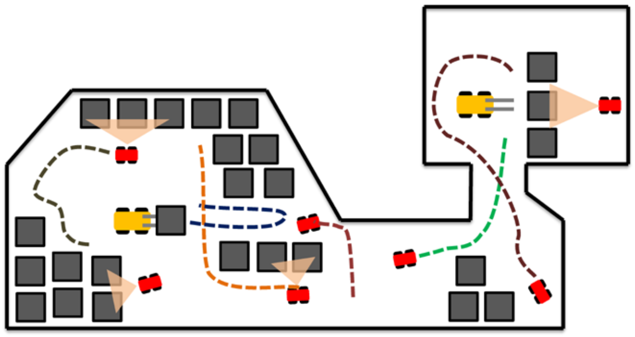


Fig. 1 A team of autonomous forklifts and ground-guidance robots operating in a constrained, minimally-prepared warehouse environment

For bigger teams operating in larger environments, the computational resources required to solve the associated centralized planning problem can become prohibitive. The communication required for a centralized planner to manage all agents can also become excessive for large-scale problems (Trodden 2009). This necessitates the development of decentralized planners, where each agent makes local decisions. However, this approach has its own challenges, and it is widely recognized that any decentralized planner must be careful to avoid conflicting local decisions (Hoffmann and Tomlin 2008; Azarm and Schmidt 1997; Pallottino et al. 2004; Jager and Nebel 2001; Purwin et al. 2008; Keviczky et al. 2004). Furthermore, minimizing a global cost function in a decentralized setting is itself a challenging problem (Basar 1988) and has motivated the development of strategies in which agents cooperatively modify their plans to reduce the global cost (Venkat et al. 2005; Purwin et al. 2008; Kuwata and How 2011).

This work examines the problem of planning dynamically feasible paths through complex environments for a team of autonomous agents attempting to reach specified goal locations. The agents are assumed to have constrained, nonlinear dynamics, as seen in any practical system, and as a result, may be unable to follow paths generated by simple path planners (e.g., piecewise-linear paths). Therefore dynamic feasibility is a crucial requirement. Agents must also be able to navigate safely in cluttered or otherwise complex real-world environments while also satisfying all inter-agent constraints (e.g., collision avoidance between all agents). Finally, the typical path planning objective of minimizing a given cost metric, (e.g., travel time or fuel consumption) must also be considered across the entire team. The approach presented here is not only able to quickly identify paths that satisfy these constraints but also reuses this path information to avoid conflicting decisions between agents. This allows multiple agents to plan paths without a severe increase in computational requirements over single-agent path planning.

2 Related work

2.1 Path planning

Due to the fundamental role of path planning in any form of mobile robotics, there have been numerous algorithms developed for autonomous systems (LaValle 2006). Standard search techniques such as A* and variants thereof (Koenig and Likhachev 2002; Likhachev and Stentz 2008) have been used to find paths through discretized environments. However, these techniques typically do not scale well to large problems due to the rapid increase in the dimension of the search space, especially when requiring dynamic feasibility (Si 2004). Several optimization-based planners have been developed, including the Mixed Integer Linear Program (MILP) formulation in Richards et al. (2002) and techniques based on Model Predictive Control (MPC) as in Dunbar and Murray (2006). While these methods can typically find minimum cost paths, they also scale poorly with the number of constraints imposed and the complexity of the dynamics and environment (Richards 2005). Potential field methods are a popular alternative due to their ease of implementation and low computational requirements for many simple scenarios (Khatib 1985; Barraquand et al. 1992). However, these methods inherently have difficulty with local minima and cluttered environments (Koren and Borenstein 1991) and suffer from higher computational complexity when applied to constrained, nonlinear dynamics (Tanner et al. 2001).

Sampling-based motion planners, such as Probabilistic Roadmaps (PRM) (Kavraki et al. 1996) and Rapidly-exploring Random Trees (RRT) (LaValle 1998), have gained popularity in recent years. These algorithms quickly build a set of feasible paths by connecting points sampled from the environment. The sampling process allows these techniques to handle increasingly complex problems, but as a result, they only provide weaker guarantees, such as probabilistic completeness. In particular, the RRT algorithm is a fast way to plan paths in complex, high-dimensional spaces due to the iterative process of growing the tree of paths and updating the current path. Frazzoli et al. (2002) propose a real-time RRT algorithm that was later extended by Kuwata et al. (2008) as the Closed-loop RRT (CL-RRT) algorithm. CL-RRT uses a closed-loop simulation of the system dynamics to grow the tree of feasible paths. As a result, it is able to handle the types of complex and constrained dynamics found in physical systems by embedding the complexity in the simulation.

The work presented in this paper builds upon the CL-RRT algorithm, and as such, an overview of the algorithm is presented in Sect. 3. Recent work by Karaman and Frazzoli (2010) has developed the RRT* algorithm, which reconnects nodes in the RRT to converge to the optimal path. It has also been applied to dynamically constrained systems (Karaman et al. 2011). However, this path refinement comes at the cost of additional computational complexity, especially when accounting for complex constraints.

2.2 Decentralized planning

Decentralization is one approach to handle large-scale multi-agent planning problems. A fundamental challenge of decentralization is guaranteeing global feasibility of a set of plans generated by multiple local decision makers, so a variety of coordination and cooperation strategies have been developed for decentralized path planning, see for example Hoffmann and Tomlin (2008), Azarm and Schmidt (1997), Pallottino et al. (2004), Jager and Nebel (2001), Purwin et al. (2008), Keviczky et al. (2004).

Consensus based approaches allow agents to coordinate actions by exchanging information on the state of each member of the team (Olfati-Saber et al. 2007). Scerri et al. (2007) propose a deconfliction strategy in which the team must reach consensus on the feasibility of each agent's updated plan. While this reduces the amount of local knowledge required to check for path feasibility, it also introduces considerable delay into the planning process and is better suited to sparse environments where only infrequent planning is required. Purwin et al. (2008) propose a related cooperation strategy where agents operate in reserved regions of the map and must reach consensus on any changes to these regions. This allows for some asynchronous planning but is conservative since any potential intersection between regions triggers a consensus check, even if it does not correspond to a conflict in time (i.e., a collision).

Reachability based planning approaches have been studied for collision avoidance along intersecting paths (Hoffmann and Tomlin 2008; Desaraju et al. 2009; Aoude et al. 2010), where the reachability analysis is used to predict potential trajectories of moving objects. In general, these methods are concerned with agents that are not necessarily cooperative, such as two vehicles approaching an intersection, and thus typically consider worst-case scenarios.

The decentralized planning problem has also been formulated as a decentralized control problem, and techniques such as such as dynamic programming (Swigart and Lall 2010) and linear programming (Ayanian and Kumar 2010) have been used to find controllers for collision-free navigation. These approaches typically provide rigorous convergence guarantees, but the computational complexity limits their application to problems with large teams in complex environments (Ayanian and Kumar 2010). Decentralized Model Predictive Control (DMPC) reformulates the MPC problem as a set of subproblems corresponding to each agent (Inalhan et al. 2002; Venkat et al. 2005; Kuwata et al. 2007; Kuwata and How 2011) and embeds additional constraints, such as collision avoidance. Typically all agents wait to plan in order in each iteration of the algorithm. However, Trodden and Richards (2006) propose a single subsystem update method where only one agent replans in each iteration, while all others continue executing their previously computed plans. This eliminates the synchronization required to cycle through all agents in each iteration.

Prioritized planning techniques also avoid conflicts through sequential planning, where the planning order is determined by the priority assigned to each agent (Erdmann and Lozano-Pérez 1987). A search over different priority orders can be used to find an ordering that reduces the global cost (Azarm and Schmidt 1997). Other work has focused on determining priorities for sequential planning in large, decentralized teams using local information (Chun et al. 1999; van den Berg et al. 2009; Velagapudi et al. 2010).

However, it is also known that in distributed computer systems that require ordered task execution (such as processor coordination and load balancing), having a fixed execution order can limit system performance. For example, a busy node or a deactivated node may block other nodes from executing additional tasks (Kim and Kim 1997; Wang and Zhuang 2008). Similarly, a fixed, sequential planning order can also lead to an inefficient use of planning time: each agent is forced to wait for its teammates to finish replanning in sequence, even if some teammates have already found their best paths to goal and have no reason to replan. As the number of agents increases, the amount of time wasted on these unnecessary replanning iterations will naturally also increase, and this time could instead be used to further improve team performance.

2.3 Overview

This paper develops a novel multi-agent path planning strategy that extends the CL-RRT algorithm (Kuwata et al. 2009) to the multi-agent case by introducing a merit-based coordination strategy that takes advantage of the RRT trees to ensure global feasibility. This coordination strategy draws inspiration from the single subsystem update technique in Trodden and Richards (2006) but takes advantage of the underlying CL-RRT planner to quickly find improved paths and produce a dynamic planning order based on each agent's incentive to replan. The resulting Decentralized Multi-Agent Rapidly-exploring Random Tree (DMA-RRT) algorithm is able to plan quickly even with complex constraints. To further improve team performance and avoid certain common deadlock scenarios, the DMA-RRT algorithm is extended by introducing a cooperation strategy. The Cooperative DMA-RRT algorithm retains the advantages of the DMA-RRT algorithm, while also allowing agents to modify their teammates' plans in order to select paths that reduce their combined cost. Furthermore, the paths generated by both algorithms are guaranteed to satisfy all inter-agent constraints, such as collision avoidance, and this is demonstrated through an extensive set of simulations and experiments.

3 Path planning with complex constraints: CL-RRT

This section provides a brief overview of the CL-RRT algorithm, which is a core component of the DMA-RRT algorithm described in Sect. 5. More detailed discussion and analysis of CL-RRT are available in Kuwata et al. (2009), Luders et al. (2010). The CL-RRT algorithm is considered here because of its capability to quickly plan paths for vehicles with constrained, nonlinear dynamics operating in complex environments. Note that CL-RRT does not aim to generate optimal paths. Instead, the primary objective is to quickly identify a set of feasible paths that are guaranteed to satisfy all constraints. The secondary objective is then to select the path from this set with the lowest cost. Other incremental planners could be used in place of CL-RRT as the core of the DMA-RRT algorithm, as long as they guarantee dynamic feasibility and constraint satisfaction. However, only CL-RRT is considered in this work as it is faster than alternative methods, such as the RRT* algorithm (Karaman and Frazzoli 2010). Furthermore, CL-RRT was successfully demonstrated at the 2007 DARPA Urban Challenge as the path planning system for Team MIT's vehicle (Leonard et al. 2008), underscoring its utility as a practical, online path planner. Aoude et al. (2010) also demonstrated that the algorithm can be extended to handle other agents moving in the environment. The algorithm consists of three components: the tree expansion process, the main execution loop, and the lazy check for feasibility.

3.1 Tree expansion

The tree expansion process (Algorithm 1) for the CL-RRT algorithm begins by randomly sampling a point x_{sample} from the set $\mathcal{X}_{free}$ of all feasible configurations of the agent in the environment. It then identifies the closest node N_{near} in the tree. The CL-RRT algorithm generates a reference signal r to guide the agent's estimated state $\hat{x}$ at N_{near} toward x_{sample}. It then uses a feedback control law of the general form

$$u = \kappa(r, \hat{x}) \tag{1}$$

to generate control inputs u that will track r and satisfy the constraint

$$u \in \mathcal{U}, \tag{2}$$

Algorithm 1 CL-RRT: Tree Expansion

```
1:  Sample point $x_{sample}$ from the environment
2:  Identify nearest node $N_{near}$ in tree
3:  $k \leftarrow 0$
4:  $\hat{x}(t+k) \leftarrow$ last state of $N_{near}$
5:  while $\hat{x}(t+k) \in \mathcal{X}_{free}(t+k)$ and $\hat{x}(t+k)$ has not reached $x_{sample}$ do
6:      Compute reference input $\hat{r}(t+k)$ from $x_{sample}$
7:      Compute control input $\hat{u}(t+k)$ from control law (1)
8:      Compute next state $\hat{x}(t+k+1)$ from propagation model (3)
9:      $k \leftarrow k+1$
10: end while
11: $N \leftarrow \hat{r}_{final}$
12: for each feasible node $N$ produced do
13:     Update cost estimates for $N$
14:     Add $N$ to tree
15: end for
```

where $\mathcal{U}$ is the set of feasible control inputs. These inputs are fed into a model of the agent's dynamics

$$\dot{\hat{x}} = f(\hat{x}, u) \tag{3}$$

to propagate the estimated state $\hat{x}(t+k)$ one timestep. If the propagation model is an accurate representation of the actual system dynamics, selecting $\kappa(r, \hat{x})$ to match the stabilizing control law used on the system will result in trajectories that are dynamically feasible (i.e., satisfy (2) and (3)) and match the behavior of the system given the same reference.

This path expansion continues until $\hat{x}$ reaches x_{sample} or becomes infeasible by violating the constraint

$$\hat{x} \in \mathcal{X}_{free}. \tag{4}$$

A new node N is created from the final reference point $\hat{r}_{final}$ (the last $r(t+k)$ computed) and is added to the tree along with an associated path cost. Nodes serve as waypoints when executing the path, and each path is uniquely characterized by its waypoints in conjunction with the dynamics and reference law used to grow it between those points.

3.2 Execution loop

The execution loop of the CL-RRT, outlined in Algorithm 2, uses Algorithm 1 to find paths that take the agent to some predefined goal region $\mathcal{X}_{goal}$. It initializes the tree of potential paths with the agent's current state, $x_{initial}$. Then, during each iteration of the execution loop, the state estimate $\hat{x}(t+\Delta t)$ is used to compute additional potential paths using the tree expansion process described above. $\hat{x}(t+\Delta t)$ is used instead of the current state $x(t)$ to account for the

Algorithm 2 CL-RRT: Execution Loop

```
1:  $t \leftarrow 0, x(t) \leftarrow x_{initial}$
2:  Initialize tree with node at $x_{initial}$
3:  while $x(t) \notin \mathcal{X}_{goal}$ do
4:      Update current state $x(t)$
5:      $\hat{x}(t+\Delta t) \leftarrow x(t)$ propagated by $\Delta t$
6:      while $elapsed\_time < \Delta t$ do
7:          Grow tree using Algorithm 1
8:      end while
9:      if no paths exist in tree then
10:         Apply safety action and goto line 19
11:     end if
12:     Select best path $p^*$ from tree using cost estimates
13:     Recheck feasibility of $p^*$ using Algorithm 3
14:     if rechecked $p^*$ is feasible then
15:         Apply $p^*$
16:     else
17:         Goto line 9
18:     end if
19:     $t \leftarrow t + \Delta t$
20: end while
```

fixed duration Δt of the tree expansion process. Note that if no path to the goal region is found in this time (as is often the case in complex environments), the algorithm proceeds using the partial paths in the tree.

Once the new potential paths are added to the tree, the minimum-cost path p^* in the tree is selected. Although the tree expansion process guarantees that all paths satisfy the current constraints, these constraints may change over time. To avoid selecting paths that have become infeasible, the lazy check described in Algorithm 3 is performed on p^* before confirming it as the new path. If p^* is pruned (due to infeasibility), the process repeats until a feasible path is found or the entire tree is exhausted. If the tree is empty, a predefined safety action, such as applying maximum braking, is executed to keep the agent in $\mathcal{X}_{free}$ (Kuwata et al. 2009).

3.2.1 *Lazy check*

The lazy check mimics the tree expansion process, but instead of generating new nodes, it re-simulates the agent's trajectory through the nodes defining p^*. If the re-simulated state $\hat{x}(t+k)$ violates a constraint, the path is pruned beyond the last feasible node. Thus p^* is guaranteed to be feasible once the lazy check is complete, confirming the property that the CL-RRT algorithm only returns paths that satisfy all constraints.

4 Decentralized coordinated planning

The most straightforward approach to decentralized, multi-agent path planning is to allow all agents to continuously

Algorithm 3 CL-RRT: Lazy Check

```
1:  for each node $N_i \in p^*$ do
2:      while $\hat{x}(t+k) \in \mathcal{X}_{free}(t+k)$ and $\hat{x}(t+k)$ has not
        reached $N_i$ do
3:          Compute reference input $\hat{r}(t+k)$ from $N_i$
4:          Compute control input $\hat{u}(t+k)$ from (1)
5:          Compute next state $\hat{x}(t+k+1)$ from (3)
6:          $k \leftarrow k+1$
7:          if $\hat{x}(t+k) \notin \mathcal{X}_{free}(t+k)$ then
8:              Remove $N_i$ and all children from $p^*$,
                goto line 11
9:          end if
10:     end while
11: end for
```

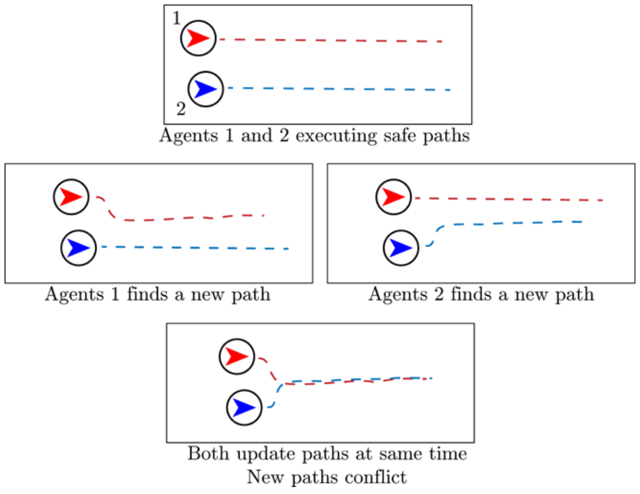

1
2
Agents 1 and 2 executing safe paths
Agents 1 finds a new path
Agents 2 finds a new path
Both update paths at same time
New paths conflict

Fig. 2 Simple asynchronous planning failure

plan their own paths subject to constraints imposed by the other agents' paths. While this has the benefit of being a fully asynchronous approach, it can easily lead to conflicting plans if one agent modifies the constraints caused by its path while another agent is replanning.

For example, consider the simple scenario in Fig. 2. Agents 1 and 2 begin with feasible plans but find alternate plans that appear to be safe given their current knowledge of the other agent. Since this is fully asynchronous, the agents can update their plans at any time. However, if both update at the same time, the new paths are not guaranteed to be safe since the constraints imposed while selecting the paths do not match the constraints that exist when the agents begin execution.

4.1 Merit-based token passing

The following presents a coordination strategy to ensure that the decisions taken by individual agents do not conflict, thus avoiding the scenario in Fig. 2. The approach taken builds on the single subsystem update idea for DMPC (Trodden and Richards 2006), which fixes all inter-agent constraints while an agent is replanning and does not require extensive synchronization between agents. Note, however, that approach still requires the agents to cycle through a fixed planning order, regardless of whether an agent will see any substantial benefit from replanning.

The approach taken here instead relies on a measure of Potential Path Improvement (PPI) that reflects an agent's incentive to replan. The PPI is the potential improvement in its own path cost that an agent expects to see if allowed to update its plan next. The single subsystem update strategy is augmented with this PPI information to form a *merit-based token passing* strategy. Rather than iterating through a list of agents, a token is used to identify which agent is allowed to update its plan at each planning iteration.

The idea of token-passing for mutual exclusion and access control is widespread in distributed computer systems (Amir et al. 1993; Banerjee and Chrysanthis 1996; Mueller 2001; Johnson et al. 2003). The merit-based token passing strategy described here uses the core feature of this technique: using a token to control which agent can access shared resources (i.e., which agent is allowed to modify the inter-agent constraints). Dynamic token-passing schemes have also been developed for distributed systems where the token is passed to whichever member satisfies some criteria, such as being the most stable or least active member of the team (Kim and Kim 1997; Wang and Zhuang 2008). There has also been considerable work on token-passing in networks with changing topology and lossy communications (Mueller 2001; Johnson et al. 2003) that, although not considered here, may be useful for future extensions of the algorithms presented in the following sections.

In the merit-based token passing strategy, agents without the token compute their PPI and broadcast these values as bids to be the next token holder. When the current token holder is finished replanning, it passes the token to the agent with the best bid, i.e., the greatest PPI. If there is a tie, one of the agents is selected at random. This effectively produces a dynamic planning order where agents that may benefit the most from replanning are able to do so sooner, without having to cycle through the list and wait for other agents that may have very little incentive to replan. As a result, agents that quickly find paths to reach their goals will typically generate higher bids more often, allowing them to act on these plans sooner and reach more of their goals. However, computing each agent's PPI quickly at any given time is a fundamental challenge that must be addressed in order to implement this type of intelligent coordination strategy.

4.1.1 PPI from RRT

Computing the PPI requires the agent to compare costs between the current plan and a new plan, i.e., $PPI = cost(p_{current}) - cost(p_{new})$. For many path planning algorithms, this would require solving for a new path p_{new} just to compute a PPI bid, which, given the complexity of solving the path planning problem in the first place, would be prohibitively expensive.

However, the tree of feasible paths maintained in the CL-RRT algorithm can be used to compute the PPI very quickly. In each planning iteration, the agent that is allowed to replan (i.e., the current token-holder) does not need to compute its PPI. So it simply selects the best path in its tree (as in CL-RRT) and begins executing it. The agents without the token continue executing their previously selected plans, but also continue to grow their own trees in order to compute their PPI. By continuing to grow their trees, it is easy for agents to identify new, lower-cost paths by simply searching the leaf nodes of the tree. The difference in path cost between the agent's current path p_k and the best path in the tree p_k^* is its PPI, i.e.,

$$PPI = cost(p_k) - cost(p_k^*). \tag{5}$$

For example, if the best path in the tree has a much lower cost than the path the agent is currently taking, the PPI would be large and it would be beneficial to replan soon to select the better path.

There are distinct advantages to using the RRT for the PPI calculation. The incremental tree growth that is central to all RRT algorithms is a computationally efficient way of generating and updating paths, as opposed to searching the entire environment for every path change (Frazzoli et al. 2002). Being able to find potential path updates quickly also allows for the cost of the current path to be compared with the costs of many new and up-to-date path options. This provides a better measure of each agent's incentive to replan than comparing to a small and possibly old set of plans. Other incremental path planning algorithms have been developed, including incremental versions of Dijkstra's algorithm (Narváez et al. 2000) and A* (Koenig et al. 2004). However, these search-based techniques again scale poorly with complex constraints and may not be able to quickly find paths traversing the environment. In contrast, sampling-based planning enables the RRT-based approach to quickly explore the environment while maintaining a dense tree of paths (LaValle 1998). Having this dense set of paths spanning a large part of the environment makes it easier to identify significantly lower cost paths, and in particular, find paths that reach the goal. This allows agents to compute non-trivial PPI values quickly and gives merit-based token passing a distinct advantage over randomly selecting (or prespecifying) the next agent to replan.

5 Decentralized multi-agent RRT (DMA-RRT)

The DMA-RRT algorithm combines the CL-RRT algorithm with merit-based token passing to coordinate the agents' actions. The algorithm consists of two components that are run in parallel on each agent. The *individual component* handles the path planning, while the *interaction component* handles all information received from other agents. Several key assumptions made in developing these components are outlined below.

5.1 Assumptions

(1) *Network*: The agents are assumed to form a fully connected network, allowing for fast information transfer across the team. This assumption may be difficult to enforce in large-scale scenarios, but it is often reasonable to assume that inter-agent constraints only need to be enforced between agents that are close enough to form a fully connected sub-network. The extension to sub-networks has been considered for other multi-agent planners (van den Berg et al. 2009), but is not considered in this work. The network is also assumed to be lossless and have negligible delay. These properties maintain data consistency between agents and can be enforced with the appropriate selection of communication protocols (Bertsekas and Gallager 1992).

(2) *Agent models*: Each agent is assumed to have a model of its own dynamics, as in the CL-RRT algorithm. Furthermore, the model is assumed to be known to all other agents, allowing for heterogeneous teams. The robustness to modeling error provided by the CL-RRT (Kuwata et al. 2009; Luders et al. 2010) often allows even relatively simple models to work well, which again motivates the extension of CL-RRT to multiple agents.

(3) *Environment*: The environment that the agents operate in is assumed to be known and only contain static obstacles. Since the behavior of the agents can be predicted accurately given a model of dynamics and appropriate communication between agents, this assumption enables each agent to anticipate the state of the world for some time into the future and plan accordingly.

(4) *Inter-agent constraints*: Finally, the set of constraints imposed between agents, such as collision avoidance or rendezvous requirements, are assumed to be symmetric between agent pairs. That is, the constraint $c(i, j)$ on agent i due to agent j is satisfied if and only if the constraint $c(j, i)$ on agent j due to agent i is also satisfied:

$$c(i, j) > 0 \quad \Leftrightarrow \quad c(j, i) > 0. \tag{6}$$

For collision avoidance, $c(i, j) = \|x_i - x_j\| - d_{ij}$, and (6) implies that all agents must maintain the same minimum separation distance, i.e., $d_{ij} = d_{ji}\ \forall i, j$.

Algorithm 4 DMA-RRT: Individual component

```
1:  Initialize with $p_0$
2:  HaveToken ← *false* except for one randomly selected agent
3:  **while** Agent is active **do**
4:      Grow CL-RRT (Algorithm 2), identify best path $p_k^*$ satisfying all constraints (Algorithm 1)
5:      **if** HaveToken **then**
6:          $p_k \leftarrow p_k^*$
7:          winner ← agent with best bid
8:          Broadcast waypoints of $p_k$ and winner to team
9:          HaveToken ← *false*
10:     **else**
11:         $p_k \leftarrow p_{k-1}$
12:         bid ← $(cost(p_k) - cost(p_k^*))$
13:         Broadcast bid
14:     **end if**
15:     $k \leftarrow k + 1$
16: **end while**
```

5.2 Individual component

The first component of the DMA-RRT algorithm, described in Algorithm 4, embeds the CL-RRT algorithm. The agent is initialized on line 1 with some dynamically feasible path, p_0, that satisfies all constraints (but need not reach the goal). For a ground vehicle, this could just be a stopped state. One agent is initialized as the token holder and the rest are assigned `HaveToken` ← *false* (line 2). Each agent grows a tree of feasible trajectories, and at the end of the planning time it identifies the best path, p_k^*, in the tree according to the cost function (line 4). The merit-based token passing strategy then determines if the agent will update its path to p_k^* and pass the token to `winner` (lines 6–9), or if it will bid to be the next token holder (lines 11–13), as described in Sect. 4.1. If the agent updates its plan (line 6), it broadcasts its new `waypoints` to the other agents (line 8), allowing them to update their constraints. This update occurs in the interaction component.

5.3 Interaction component

The second component of the DMA-RRT algorithm, described in Algorithm 5, controls interaction between agents. Communication between agents involves two different message types. The first has a list of `waypoints` corresponding to an updated plan and the name of the token `winner`, and it is sent after the current token holder updates its plan. The second message is just the PPI `bid` and is sent when an agent without the token finds a better path.

When the `waypoints` and `winner` message is received, the agent must update its constraints. Using a model

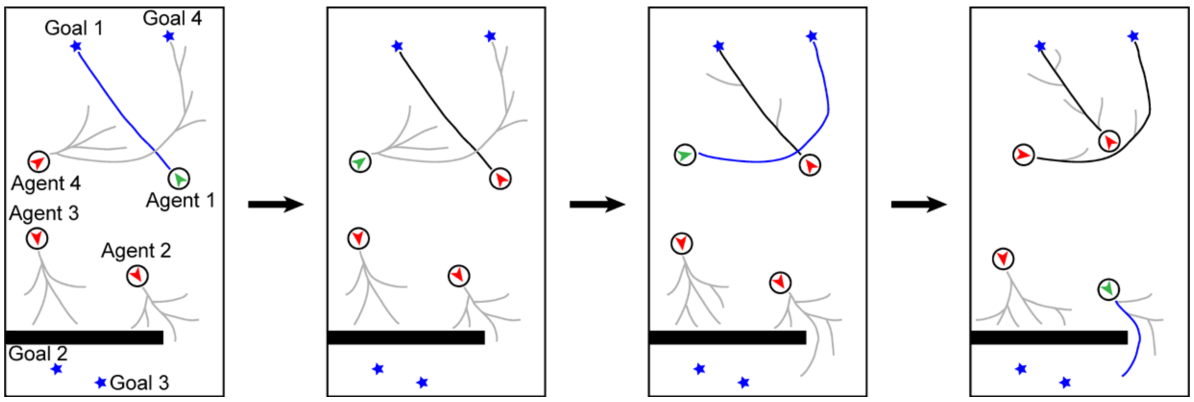


Fig. 3 Example scenario illustrating merit-based token passing

Algorithm 5 DMA-RRT: Interaction component

```
1:  while Agent is active do
2:      Listen for messages
3:      if received waypoints and winner message then
4:          Simulate other agent's trajectory along waypoints, update constraints
5:          if agent is winner then
6:              HaveToken ← true
7:          end if
8:      end if
9:      if Received bid message then
10:         Update sender's bid
11:     end if
12: end while
```

of the others agent's dynamics, it can simulate the other agent's path along the received `waypoints` (line 4). This choice to exchange only waypoints greatly reduces the amount of information communicated between agents, but increases the computational effort, though it is relatively easy to reproduce the detailed, time-parameterized path from a sparse set of waypoints and the known closed-loop dynamics (Sect. 3.1). The lower bandwidth required to exchange waypoints, as opposed to full, nonlinear trajectories, is especially important as the number of agents and therefore the amount of inter-agent communication increases.

This trajectory information can then be used in the CL-RRT's feasibility checks to ensure the path satisfies all inter-agent constraints. When the token winner receives the `waypoints` and `winner` message, it becomes the token holder (line 6) and can select a new plan as soon as it updates its constraints. When a `bid` message is received, it is simply added to the list of bids to check after the next plan update in line 10. These bids are stored even when the agent does not have the token so that once it does receive the token, it is not required to wait for bids from the other agents. These `bid` messages are sent much more frequently than the `waypoints` and `winner` messages, thereby ensuring that token winner is selected based on the current PPI values for each agent. Of course, if only a single agent is active, it is always the token holder and DMA-RRT reduces to CL-RRT.

5.4 Example

Figure 3 illustrates the DMA-RRT algorithm and the advantage of merit-based token passing. The agents are assumed to have steering-constrained bicycle dynamics (Kuwata et al. 2008) and must simply reach their respective goals. In the first panel, agent 1 has the token and selects a path to its goal. Then agent 4 has the best PPI due to a potential new path that reaches its goal, so it receives the token from agent 1. Agent 4 selects this path while the other agents continue executing their old plans and growing their trees. In the last panel, agent 2 has the best potential new path. So it receives the token from agent 4 and updates its path. As this process continues, agents 2 and 3 will continue replanning to reach their goals, while agent 4 will likely find an improved path to its goal. Since agent 1 has a direct and unconflicted path to its goal, it has very little reason to modify its plan. As a result, it generates low PPI bids that will not disrupt the rest of the team.

In contrast, if a fixed planning order was used (assuming the order is 1, 2, 3, 4; though similar situations will arise for any fixed ordering of the agents) instead of merit-based token-passing, agent 4 would have to wait for agents 2 and 3 to replan before modifying its path, despite having a very strong incentive to select the new path. Agent 1 would also be given many chances to replan, despite having found a direct and non-conflicting path to its goal. These delays would obviously increase with the number of agents. The ability to replan quickly is essential for efficient navigation in complex, cluttered environments, and allowing these delays to persist and accumulate can severely limit team performance.

5.5 Path feasibility

As described in Sect. 3, any path returned by the CL-RRT algorithm is guaranteed to be dynamically feasible and satisfy all constraints. More formally, a trajectory returned by CL-RRT at iteration k with length $N+1$ timesteps, denoted as $p_k = \{x(k), x(k+1), \ldots, x(k+N)\}$, will satisfy the standard CL-RRT feasibility constraints (Luders et al. 2010) given in (2), (3), and (4), as well as the inter-agent constraints added by DMA-RRT in (6). However, since trajectories returned by the CL-RRT have finite length, an additional constraint is introduced in order to guarantee path feasibility and constraint satisfaction for all time:

$$\{x(k+M) \in \mathcal{X}_f \,|\, \forall M > N\} \tag{7}$$

$$\mathcal{X}_f \subseteq \mathcal{X}_{free}, \tag{8}$$

where the control invariant set $\mathcal{X}_f$ can be entirely characterized by a set of time-parameterized waypoints. With this constraint enforced, an agent's behavior is both well-defined and feasible even if it reaches the end of its nominal path, allowing other agents to plan accordingly. Examples of simple invariant terminal sets include a stopped state for a ground vehicle or a loiter pattern for a fixed-wing aircraft, provided the set is feasible when entered (Schouwenaars et al. 2004; Kuwata et al. 2007). Using these properties, DMA-RRT provides the following guarantee: any path $p_{i,k}$ generated by agent i at iteration k using the DMA-RRT algorithm is guaranteed to satisfy all constraints at all times. This guarantee holds even if the constraints change, provided they still satisfy (6).

Theorem 1 *Given a set of n cooperative agents and a set of inter-agent constraints satisfying* (6), *if the initial set of paths $\{p_{i,0} | i = 1, \ldots, n\}$ satisfies all constraints, then using the DMA-RRT algorithm, the set of all future paths $\{p_{i,k} | i = 1, \ldots, n$ and $k \geq 0\}$ will satisfy all constraints.*

Proof Assume the set of all agents' paths $\{p_{i,k} | i = 1, \ldots, n\}$ at planning iteration k satisfies all constraints. Then at iteration $k+1$, let agent j be the token holder, in which case agent j updates its path, while all other agents keep their existing paths:

$$p_{j,k+1} = p^*_{j,k+1}, \tag{9}$$

$$\{p_{i,k+1} = p_{i,k} | i \neq j\}. \tag{10}$$

Since $p_{j,k+1}$ is generated by the CL-RRT algorithm, it is guaranteed to satisfy all constraints (Sect. 3). As a result, the set of constraints imposed on other agents is only changed in such a way that their existing plans continue to satisfy them (Sect. 5.1). Then $p_{i,k}$ satisfying all constraints implies that $p_{i,k+1}$ satisfies all constraints for $i \neq j$. Thus, the set of paths $\{p_{i,k+1} | i = 1, \ldots, n\}$ at iteration $k+1$ satisfies all constraints. By assumption, $\{p_{i,k} | i = 1, \ldots, n\}$ satisfies all constraints for $k = 0$. Therefore, by induction, $\{p_{i,k} | i = 1, \ldots, n\}$ satisfies all constraints for all $k \geq 0$, i.e., all agents' paths satisfy all constraints for all time. □

6 Cooperative DMA-RRT

Although the DMA-RRT algorithm implements a coordination strategy, each agent only aims to minimize its own path cost when selecting a path from the tree. However, this locally greedy approach does not necessarily minimize the global cost (Basar 1988), i.e., the sum of path costs across all agents in the team. As a result, one agent may begin executing a path that prevents several of its teammates from selecting paths that would reduce the total cost across the team. And in some scenarios, this behavior can lead to deadlock. For example, consider a cluttered environment with narrow passages between obstacles. It is then possible for agents to select paths that are conflict-free but obstruct these passages, thus preventing multiple agents from making additional progress toward their goals (see Sect. 6.2).

The ability to modify other agents' paths would prevent many such scenarios, and cooperation strategies have been proposed for other algorithms based on this idea. For example, Kuwata and How (2011) present a decentralized MILP-based planner in which agents modify a set of candidate plans to minimize the global cost. However, it requires all agents to wait until the entire team has made necessary modifications to the candidate plans before applying control actions. In the cooperation strategy described by Purwin et al. (2008), agents communicate to modify planning regions, but only when an agent's desired planning region conflicts with another agent's current region. Section 6.4 presents a further comparison of that approach with the one in this paper.

This section introduces a cooperation strategy that allows an agent to modify its own path as well as the path of another agent in order to minimize the combined path cost and thus reduce the global cost. Minimizing the global cost would, in general, require modifying plans for all agents, as in the case of a centralized planner. However, this pairwise coordination begins to mimic the advantage of the centralized planner without sacrificing the decentralized implementation or incurring the communication penalty. The resulting Cooperative DMA-RRT algorithm introduces *emergency stop* nodes along each agent's path where the agent could safely terminate the path if asked to by another agent. The decision to terminate another agent's path is based on a cost comparison between teammates and enables the team to avoid costly deadlock scenarios.

This bears some resemblance to the idea of inserting idle times in paths to avoid conflicts (Lee et al. 1995;

Algorithm 6 Cooperative: Individual component

```
1: Initialize with $p_0$
2: HaveToken $\leftarrow$ *false* except for one predetermined agent
3: while Agent is active do
4:     Grow CL-RRT ignoring others' paths, identify best path $p_k^*$
5:     if HaveToken then
6:         if $p_k^*$ conflicts with some agent $j$ then
7:             Check emergency stops (Algorithm 7)
8:         end if
9:         if $p_k^*$ conflicts with some other agent $j'$ then
10:            $p_k^*$ pruned to avoid conflict with agent $j'$
11:        end if
12:        Identify emergency stop nodes on $p_k^*$
13:        $p_k \leftarrow p_k^*$
14:        if agent $j$'s plan was modified then
15:            winner $\leftarrow$ agent $j$
16:        else
17:            winner $\leftarrow$ agent with best bid
18:        end if
19:        Broadcast waypoints of $p_k$ (with emergency stops) and winner to team
20:        HaveToken $\leftarrow$ *false*
21:    else
22:        $p_k \leftarrow p_{k-1}$
23:        bid $\leftarrow (p_k.cost - p_k^*.cost)$
24:        Broadcast bid
25:    end if
26:    $k \leftarrow k+1$
27: end while
```

Jager and Nebel 2001). However, the emergency stop cooperation strategy has the advantages of allowing agents to anticipate conflicts, resolve these conflicts in a way that reduces the global cost, and update their plans based on the state of the world after the cooperation check. These key aspects of the algorithm are detailed below.

6.1 Modified DMA-RRT algorithm

Most of the algorithmic changes required to implement the cooperation strategy occur in the individual component (Algorithm 4). The modified component is presented in Algorithm 6 with additions highlighted in gray.

When agent i selects a plan p_k^*, it also identifies several nodes where it could stop if needed (line 12). These emergency stop nodes can be selected by taking every nth safe node or selecting the next safe node every n seconds along the path, where safe indicates the agent can stop at the node for all future time without violating any constraints. This idea of stopping safely can be generalized to a safe, invariant set, such as those described in Sect. 5.5. When the agent

Algorithm 7 Emergency Stop Check

```
1: if CheckEstops then
2:     for all viable emergency stop nodes $N_l$ in agent $j$'s path do
3:         Find last safe stop node in $p_k^*$ if agent $j$ stops at $N_l$
4:         TotalCost$_l$ = cost of path ending at this node + agent $j$'s cost to stop at $N_l$
5:     end for
6:     Select terminal node and $N_l$ that minimize TotalCost$_l$
7:     if $N_l$ is not agent $j$'s original terminal node then
8:         Send estop message to agent $j$ to stop at $N_l$
9:     end if
10:    if selected terminal node is not original terminal node then
11:        $p_k^*$ pruned past new terminal node
12:    end if
13: else
14:    $p_k^* \leftarrow p_k^*$ pruned to satisfy all constraints
15: end if
```

broadcasts its plan on line 19, it also indicates which nodes can be emergency stops. Like the terminal node, the invariant sets corresponding to these nodes block off areas where the agent can remain safe for all time. Thus no other agents can select paths going through those areas, unless they do so before agent i is expected to arrive. These nodes already have associated costs from the tree-growing process of the CL-RRT, so other agents know the cost of forcing this agent to stop at one of these nodes.

Instead of checking for path conflicts when growing the tree (line 4), the algorithm uses the Emergency Stop Check described in Algorithm 7 to maintain feasibility with the other agents' paths. If the path selected by agent i does not come into conflict with another agent's path, the algorithm proceeds as in Sect. 5. However, if the path conflicts with some agent j, the Emergency Stop Check is performed (line 7). If the same path conflicts with another agent j', the Emergency Stop Check is not performed and agent i's path is truncated at that point, as in the DMA-RRT (lines 9 and 10). This avoids the problem of having to check emergency stops for all conflicting agents, which is important, as the number of combinations to consider could grow exponentially with the number of agents. The algorithm may be able to avoid additional deadlock scenarios if modified to consider interactions between more than two agents, but only as long as the number of combinations to check remains tractable for online planning.

In line 2 of the Emergency Stop Check, agent i identifies the last node in its path p_k^* at which it can stop safely for each of agent j's emergency stop nodes. Since agent i knows the costs associated with each of its own nodes, as

Algorithm 8 Cooperative: Interaction component

```
1: while Agent is active do
2:     Listen for messages
3:     if received waypoints and winner message then
4:         Simulate other agent's trajectory along waypoints, update constraints
5:         if agent is winner then
6:             HaveToken ← true
7:         end if
8:     end if
9:     if Received bid message then
10:        Update sender's bid
11:    end if
12:    if Received estop message then
13:        Terminate $p_k$ at node stop specified in estop
14:        CheckEstops ← false
15:    end if
16: end while
```

well as agent j's costs for stopping early, it can select the combination of terminal node and emergency stop node that yields the best total cost (line 6). If this requires agent j to stop early, a message is sent indicating which emergency stop node it should select (line 8), and the modified path p_k^* is returned to Algorithm 6. Then agent i passes the token to agent j so that it can update its plan (line 15 of Algorithm 6). However, to prevent cycles between agent i and agent j, agent j cannot use the emergency stop logic on this planning iteration. At the very least, agent j will then broadcast the truncated version of its old plan, thus updating all other agents and preventing them from detecting conflicts with the old path.

Only minimal changes are required to the interaction component (Algorithm 5) to implement this cooperation strategy. These changes are highlighted in gray in Algorithm 8. When an agent receives an `estop` message, it prunes all nodes in its current path p_k after the emergency stop node in the message (line 13). It also sets the `CheckEstops` flag to $false$ in line 14 to avoid cycles, as described above. This emergency stop cooperation strategy is illustrated by the following example.

6.2 Example

Figure 4 shows a simple two-agent scenario that highlights the benefits of this cooperation strategy. The agent in green (Agent 1) is the current token holder. The first column (Fig. 4(a)) shows the DMA-RRT algorithm from Sect. 5. In the first panel, Agent 1 plans a path that ends at the entrance to the narrow hallway, since planning through narrow passages is typically difficult for RRT algorithms (Yershova et al. 2005), especially with limited time for tree expansion (Sect. 3.2). Agent 2 then finds a path to the goal (panel 2) but is forced to terminate the path in the hallway to avoid a conflict with Agent 1's path (panel 3). This leads to a deadlock scenario in the last panel where neither agent is able to progress toward its goal by selecting paths only based on local cost functions.

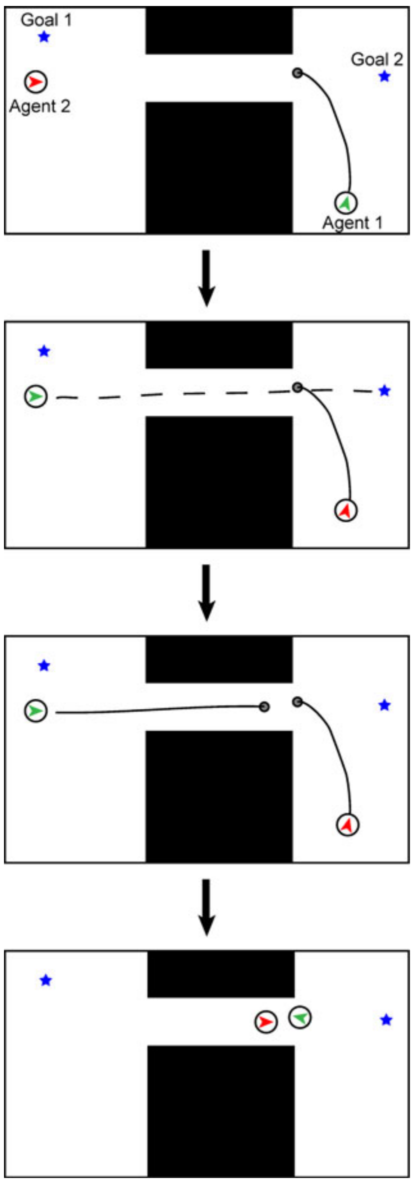
The second column (Fig. 4(b)) shows the same scenario, now with the cooperation strategy in Sect. 6.1. Agent 1 generates the same plan but also identifies two emergency stop nodes, indicated by the gray dots. These stop nodes are embedded in the waypoint information that is broadcast to the rest of the team. This could be as simple as appending the cost to specific nodes to identify them as emergency stops and thus does not significantly increase the amount of data that must be transmitted. Agent 2 then finds a path to its goal and compares the three different scenarios shown in the third panel. The first option leaves Agent 1's plan untouched, the second option forces Agent 1 to stop at the second emergency stop node, and the third option forces Agent 1 to take its first stop. Since the cost associated with each emergency stop node is embedded in the received waypoint information, the only additional computation required to make a decision is a search over all combinations of Agent 2's waypoints and Agent 1's emergency stop nodes. The second option allows Agent 2 to reach its goal while cutting off only a small part of Agent 1's path. This reduces the cost penalty incurred by Agent 1 while allowing Agent 2 to select its low cost path. Since this option yields the greatest reduction in the overall team cost, it is selected (panel 4). Following through on these new plans allows Agent 1 to plan a path to its goal in the last panel, as soon as Agent 2 passes.

6.3 Path feasibility

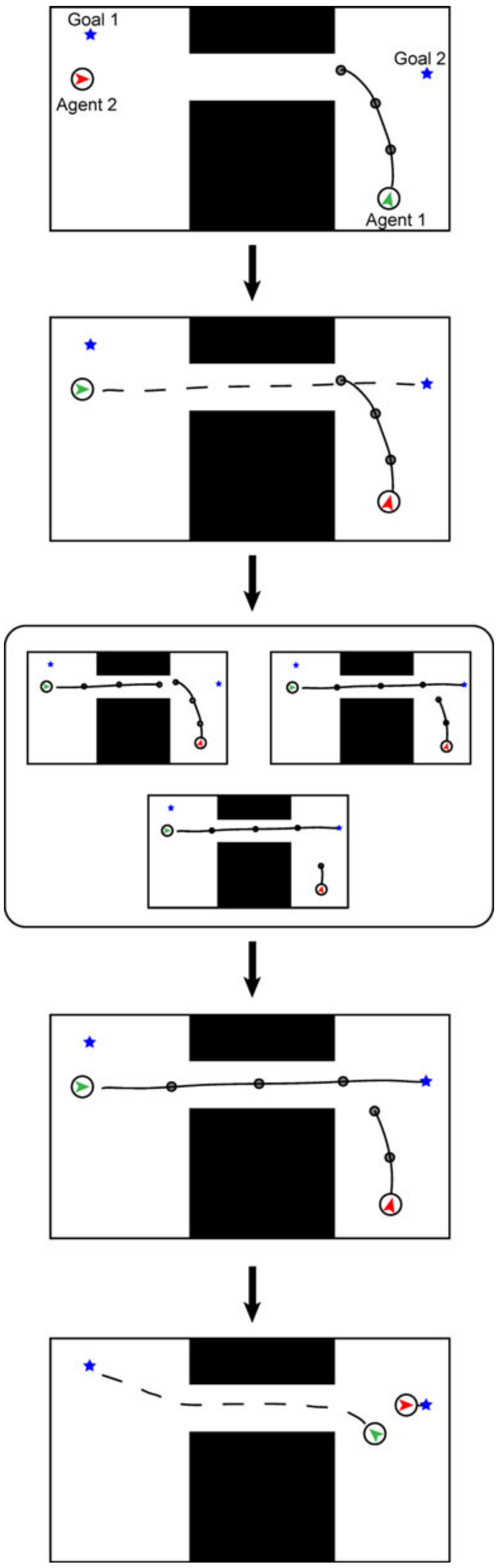
Adding this cooperation strategy to DMA-RRT preserves the path feasibility guarantee from Theorem 1. By construction, all emergency stop nodes satisfy the safe stopping requirement for terminal nodes (Sect. 5.5). Thus, the modified path ending at an emergency stop node is a path that satisfies the terminal node requirement. As with the terminal nodes, these emergency stop nodes are known to all agents, ensuring no other plans will be in conflict with them.

Furthermore, the only emergency stop nodes that are considered are ones that are sufficiently far ahead in time for the agents to react to. That is, agent i only performs the emergency stop check on nodes that agent j will not have reached by the end of agent i's replan time. This compensates for any delays in the other agent truncating its path, ensuring feasibility even during message passing. Therefore, the modified set of plans across all agents is feasible before, during, and after each planning iteration, and the proof of Theorem 1 holds.

Fig. 4 Example scenario illustrating key difference between (**a**) DMA-RRT and (**b**) Cooperative DMA-RRT



(a) Agent 1 plans a path that prevents Agent 2 from reaching its goal. The two agents reach the ends of their paths, only to be obstructed by the other agent.



(b) Agent 1 plans a path with emergency stops. Agent 2 finds a path to the goal, considers Agent 1's emergency stop nodes and selects the best one. Agent 1 truncates its path and waits for Agent 2 to pass

6.4 Limiting case

The separation between emergency stop nodes is a user-selectable parameter. Maximum separation effectively reverts the cooperative algorithm to the DMA-RRT algorithm of Sect. 5. Smaller intervals provide greater flexibility in path modifications at the cost of evaluating the additional combinations and blocking off more of the environment. In the limiting case with no separation, all nodes in the path can be treated as emergency stops. Although this will greatly increase the computational requirements to check all the emergency stops, it also allows agents to stop almost anywhere on their current paths without the risk of violating constraints.

In some respects, having no separation is a special case of the approach described by Purwin et al. (2008), in which agents reserve non-intersecting areas of the environment where they are free to plan without the risk of colliding and must cooperate to modify these safe areas. However, the emergency stop nodes retain the distinct advantage of being time-parameterized. If an agent is expected to arrive at a node at time t_1, the safe area corresponding to that node (i.e., the invariant set) is only infeasible for other agents starting at time t_1, rather than blocking the space for all time, as in Purwin et al. (2008). This allows agents to plan conflict-free paths through that space for any time $t < t_1$. Agents planning through that space for $t \geq t_1$ must perform the emergency stop check instead.

Allowing the agent to stop virtually anywhere on its path also introduces an option for relaxing the static environment assumption from Sect. 5.1. If the agents are operating in a world with dynamic objects whose behavior is uncertain but non-adversarial (such as humans working alongside the autonomous agents), the ability to stop at any node typically is enough to avoid collisions with these objects.

7 Simulation

7.1 Setup

The DMA-RRT algorithms are implemented in a real-time Java simulation. To maintain the decentralized nature of the algorithms, each agent is simulated on a different computer, with all the computers connected to the same local area network and communicating via UDP. All agents are assumed to be identical two-wheeled, skid-steer vehicles, whose dynamics are given by the modified bicycle model

$$\dot{x} = v\cos\theta, \qquad \dot{y} = v\sin\theta, \qquad \dot{\theta} = \frac{\Delta v}{L_w}, \tag{11}$$

where $v = \frac{1}{2}(v_L + v_R)$, the average of the left and right wheel velocities, $\Delta v = v_L - v_R$, and L_w is the distance between the left and right wheels (Sae-Hau 2003). Since v_L and v_R are also the two control inputs to the agent, no additional velocity controller is needed. However, the velocities are subject to the physical constraints $|v_L| \leq v_{\max}$ and $|v_R| \leq v_{\max}$. The pure-pursuit steering control law in Kuwata et al. (2008) is used to track the reference path with a smooth trajectory. This control law, adapted to skid-steer dynamics is given by

$$\Delta v = -v\left(\frac{L_w \sin\eta}{\frac{L_1}{2} + l_a\cos\eta}\right) \tag{12}$$

where L_1 is the look-ahead distance, l_a is the distance of the anchor point from the axle, and η is the angle from the vehicle to the look-ahead point on the reference path.

The inter-agent constraint, collision avoidance, is implemented by modeling other agents as moving obstacles. Whenever another agent's path information is received, the corresponding trajectory is computed by simulating the agent's dynamics along the nodes in the path and storing this as a moving (time-parameterized) obstacle. Then, without any modifications, the underlying CL-RRT is able to check the collision avoidance constraint when selecting a path. The samples for the CL-RRT are drawn from a uniform distribution, as opposed to a distribution biased according to the environment, to reflect the minimally-prepared nature of the environment.

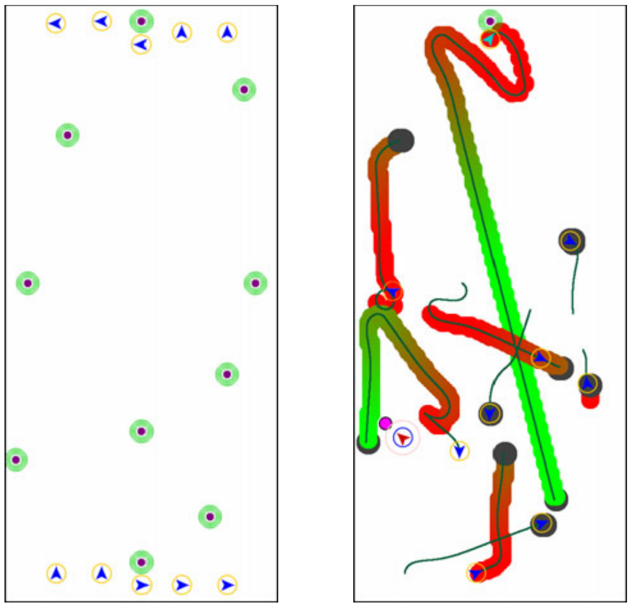
Two simulation scenarios are presented in the following sections using this implementation of the DMA-RRT algorithms. The results for each are based on 120 minutes of simulation data (12 ten-minute trials), and performance is measured in terms of the average number of goals each agent is able to reach in ten minutes.

7.2 Scenario: ten agents on open map

The first scenario involves a team of ten agents operating in an open environment. Though there are no static obstacles (other than the perimeter), the density of agents in this limited area increases the complexity of the environment. Each agent is assigned one of the initial positions and one of the goal locations in Fig. 5(a). One arbitrarily selected agent starts with the token. Upon reaching a goal, agents select the next goal in the list (looping back to the first goal after the tenth) and resume planning.

7.2.1 DMA-RRT without merit-based token passing

This scenario is first run *without* merit-based token passing. Instead, the agents use a round-robin approach, cycling though a fixed planning order. Figure 5(b) shows the state of the world as seen by agent 1 (blue circle with red arrow) at one instance in time. The other agents are represented by yellow circles with blue arrows, and the current


(a) Ten initial positions (circles with arrows) and ten goal locations (bull's-eye shapes)

(b) State of the world with time-parameterized obstacles as seen by agent 1

Fig. 5 (Color online) Snapshots from Ten Agents on Open Map simulation

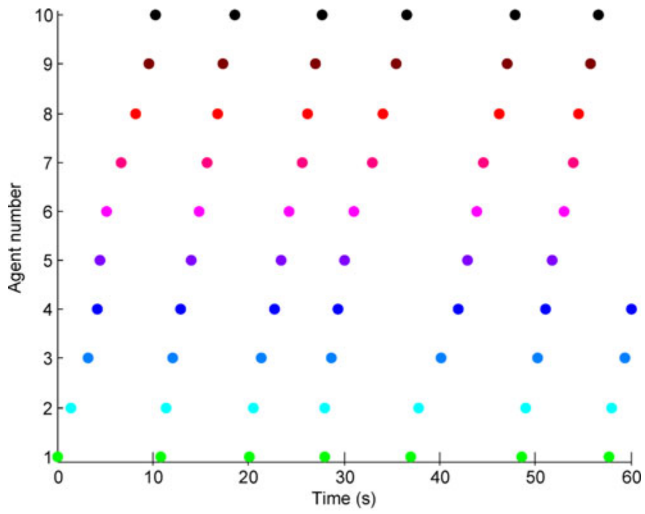
token-holder's arrow is highlighted.[1] The magenta dots are the waypoint nodes along agent 1's current path. The time-parameterized obstacles produced by the other agents' trajectories are shown by the red-to-green paths. Time is represented by the color gradient from bright red (current obstacles, i.e. the other agents' current positions) to bright green (obstacles at least 10 seconds ahead in time). The gray areas are the safe terminal nodes for each path. These are all based on agent 1's re-simulation of the other agents' trajectories.

The fixed planning order is evident in the example replan start times shown in Fig. 6(a). On average, each agent reached a total of 12.5 goals in a span of ten minutes with a standard deviation of 3.8. Note that this provides a more realistic performance baseline than having the agents just run CL-RRT with no coordination strategy, as the latter approach cannot guarantee collision avoidance (as illustrated in Fig. 2).

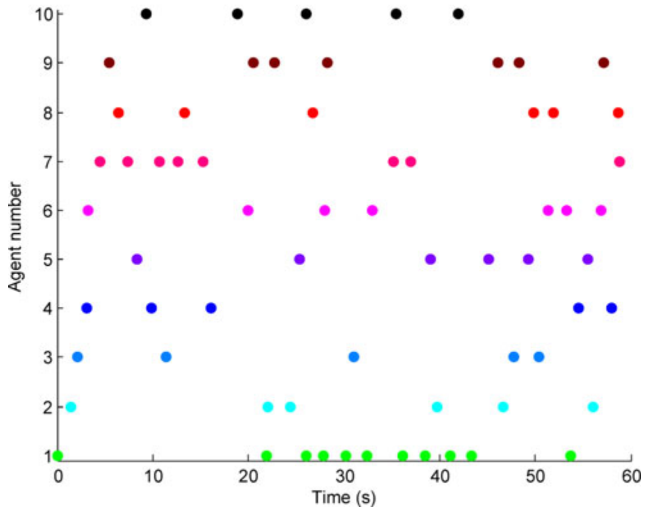
7.2.2 *DMA-RRT*

The same test is also performed with the full DMA-RRT algorithm, i.e., using merit-based token passing. Each agent computes its path cost, and thus its bid, based on the path's length in seconds (i.e., travel time). Adding the intelligent coordination policy yields significantly better results. Each agent reached an average of 15.1 goals in ten minutes with a standard deviation of 3.2. The dynamic planning order produced by this token passing strategy is also apparent from a comparison of the replan start times in Fig. 6(b) with those in Fig. 6(a).

[1]The arrow is cyan in this case since one of the other agents has the token. If agent 1 were the current token holder, its arrow would be green instead.



(a) Sequential planning order - consistent pattern of replan times



(b) Merit-based planning order - agents plan more often or less often at different times

Fig. 6 Replan start times (**a**) without and (**b**) with merit-based token passing (Ten Agents on Open Map)

7.2.3 *Analysis*

This simulation scenario demonstrates the DMA-RRT algorithm's ability to handle large teams of agents. When using the round-robin strategy, all agents are given equal opportunities to plan, irrespective of whether they need to replan at that time. However, if an agent finds a new path after passing the token, it is forced to wait for the token to pass through

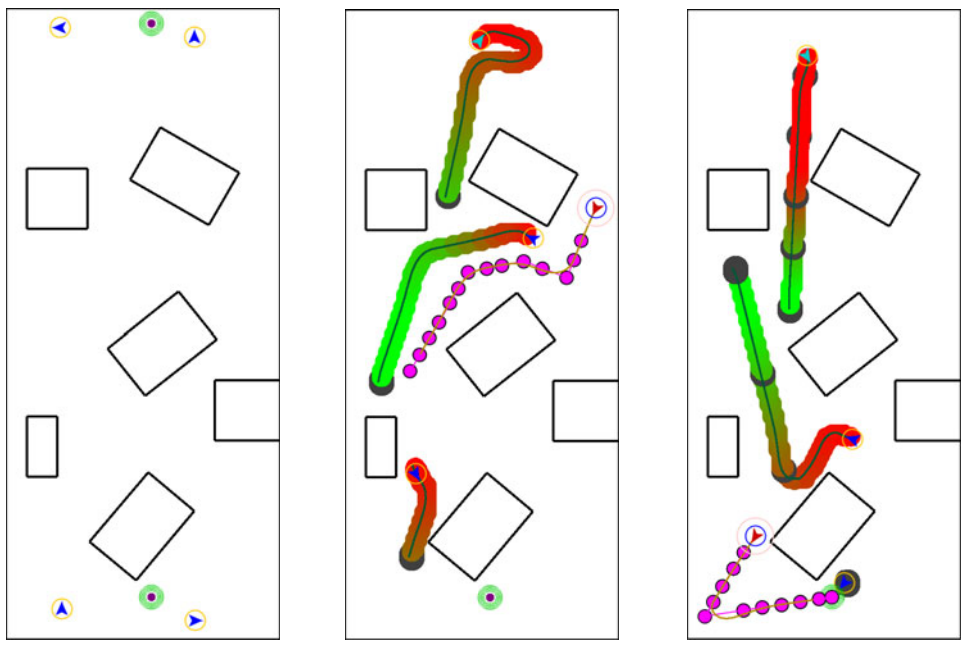


Fig. 7 Snapshots from Four Agents with Obstacles simulation showing (**a**) setup, (**b**) DMA-RRT, and (**c**) Cooperative DMA-RRT

(a) Obstacle layout, four starting positions, two goals

(b) State of the world as seen by agent 1

(c) Emergency stop nodes (in gray) on other agents' paths

the entire team before receiving it again. As Fig. 6(a) shows, the wait time is typically around ten seconds for this ten-agent scenario. In a more complex scenario with additional agents and obstacles, this delay would only increase.

In comparison, merit-based token passing enables agents to regain the token within a second or two of passing it, if needed. An example of this is in Fig. 6(b) around 12 seconds where agent 7 passes the token to agent 3 but wins the bid to get it back from agent 3 immediately after. This reduces the effective "penalty" incurred in the round-robin approach when agents with a large incentive to replan may be blocked from doing so by agents with little incentive. Thus, agents that are able to update to shorter paths can do so much more quickly, reducing the travel time between goals and increasing the number of goals that can be reached in a fixed amount of time. The simulation results reflect this quite clearly, with a 20% increase in the average number of goals per agent, from 12.5 to 15.1.

7.3 Scenario: four agents with obstacles

The second scenario is in a complex environment with several obstacles, as shown in Fig. 7(a). Four agents are initialized at the positions shown in the figure, and each agent alternates between the two goals.

7.3.1 DMA-RRT without merit-based token passing

Again, this scenario is first run using the round-robin approach. Figure 7(b) shows a snapshot of agent 1's knowledge of the world during a sample run. Recall that the magenta dots are agent 1's waypoints and the gray circles are the invariant terminal sets for each path. On average, each agent reached a total of 11.9 goals in a span of ten minutes.

7.3.2 DMA-RRT

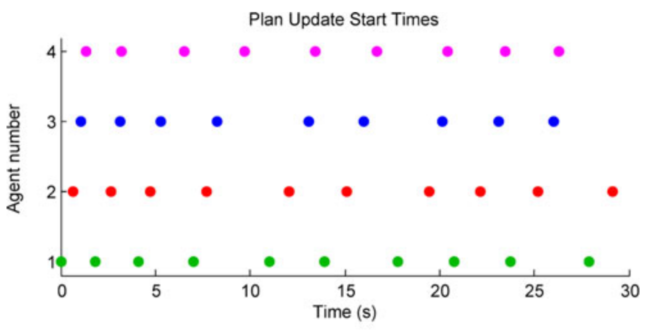
The same scenario is re-run with the full DMA-RRT. In this case, even with merit-based token passing, each agent averages 11.9 goals in ten minutes. As Fig. 8(b) shows, the frequency with which each agent receives the token is not substantially different from that of Fig. 8(a).

7.3.3 Cooperative DMA-RRT

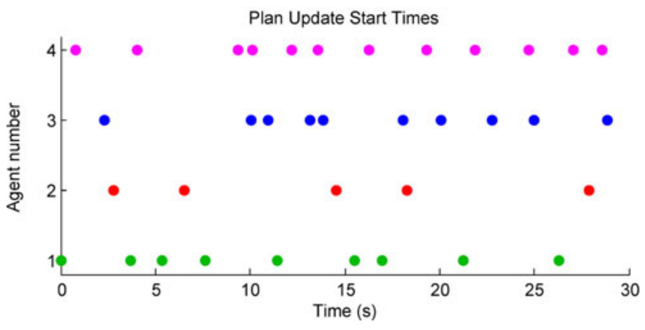
Finally, the emergency stop cooperation strategy is enabled, and the same scenario is run again. The stop nodes are placed approximately every four seconds along each trajectory, as seen in Fig. 7(c). With this cooperation strategy, each agent averaged 13.0 goals in ten minutes.

7.3.4 Analysis

For a team of only four agents, the merit-based token passing strategy is not expected to provide as significant an improvement in performance, in general (except in cases where, for example, one agent has a much more challenging route to plan). This can be seen from a comparison of Fig. 8(a) and Fig. 8(b). Even with the dynamic planning order, there are so few agents that each receives the token almost as regularly as in the round-robin case. Since each agent's ability

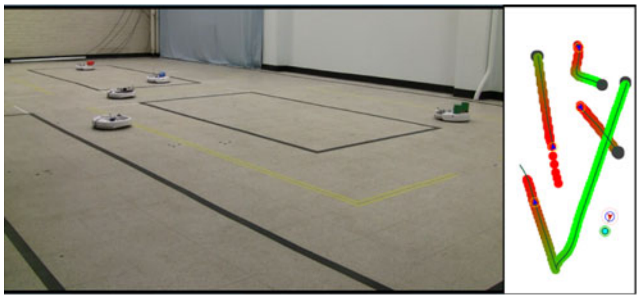


(a) Sequential planning order - again form a consistent pattern

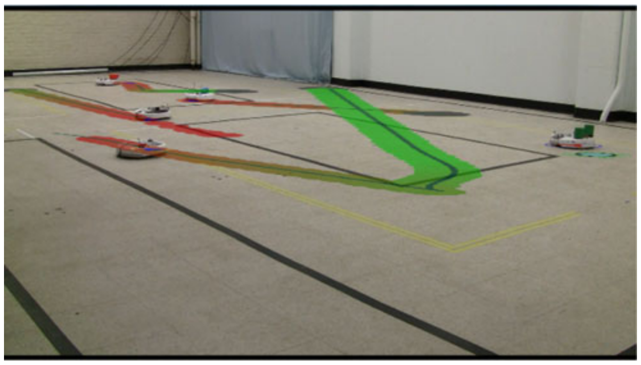


(b) Merit-based planning order - replan frequency not significantly changed from sequential case

Fig. 8 Replan start times (**a**) without and (**b**) with merit-based token passing (Four Agents with Obstacles)



(a) Snapshot of the vehicles and their trajectories



(b) Overlay of trajectory visualization on actual testbed

Fig. 9 DMA-RRT in RAVEN

to update its path is not significantly altered, the overall performance cannot be expected to improve just by adding this token passing strategy.

However, introducing cooperation does make a difference due to the narrow passages formed by the obstacles. As in the example from Sect. 6.2, these passages are difficult to plan through, often resulting in agents stopping near the openings. Therefore, an agent that is able to find a path through one of the openings can use the cooperation strategy to prevent others from obstructing this path. Furthermore, a standard error analysis shows the 95% confidence interval for DMA-RRT is [11.39, 12.49] (average goals per agent), while with cooperation it is [12.53, 13.47]. These non-overlapping intervals are a strong indicator that the cooperation strategy provides a statistically significant improvement in performance. From a practical perspective, this improvement translates to the team completing an additional 11 tasks (goals) in that time-span.

8 Experimental results

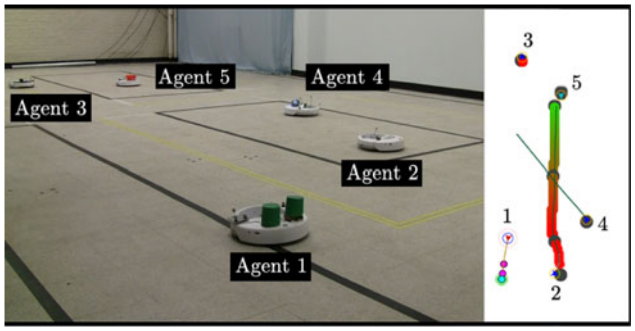
8.1 Setup

Hardware tests of the DMA-RRT algorithms were performed in the Real-time indoor Autonomous Vehicle test ENvironment (RAVEN) at the MIT Aerospace Controls Laboratory (How et al. 2008). The team consisted of five iRobot Creates, which can be modeled accurately by skid-steer dynamics (11). The algorithms run on dedicated vehicle computers that send drive commands to the vehicles.

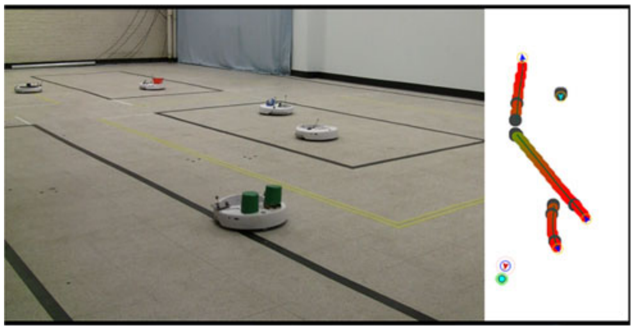
8.2 Results

Both the DMA-RRT and Cooperative DMA-RRT were tested on an open map with ten goals (Fig. 5(a)), with the agents starting at arbitrary locations. Figure 9 shows a snapshot from a run of the DMA-RRT algorithm. As in the simulation, the color gradients on the trajectory visualization indicate the time at which that area of the environment is expected to be occupied by a given agent. The overlay of this visualization onto the image (Fig. 9(b)) shows the physical locations of these time-parameterized obstacles.

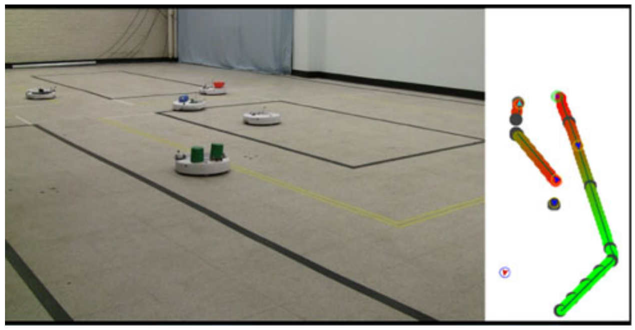
Figure 10 shows one instance of the emergency stop cooperation strategy activating. Agent 2 first plans a path terminating just short of its goal due to the position of agent 5. Agent 4 then receives the token and finds a path to its goal, but it identifies a conflict with Agent 2's path. Since paths that reach the goal are preferred, Agent 4 sends an emergency stop message to Agent 2 to terminate its path at its first stop node. Agent 2 complies and upon receiving the token from Agent 4, finds an updated plan that allows it to stop closer to its goal without interfering with Agent 4's path. The agents continue along these paths, with Agent 2 waiting at its terminal node for Agent 4 to pass.



(a) Agent 2 plans a path, then agent 4 find a path to goal that conflicts with agent 2



(b) Agent 4 sends an emergency stop message to agent 2, agent 2 terminates its path early, and agent 4 is able to proceed to the goal



(c) Agent 2 reaches the end of its plan and waits for agent 4 to pass

Fig. 10 Emergency stop logic in action

Table 1 shows the minimum separation distance between agents over the course of a five minute run of the DMA-RRT algorithm. The distance between two agents placed side by side is measured by the Vicon system to be 0.30 m. The minimum separation between each pair of agents is greater than this threshold for all combinations of agents, indicating that the collision avoidance constraint is satisfied for the entire run. Table 2 shows similar results for Cooperative DMA-RRT. Furthermore, these experimental results demonstrate that the DMA-RRT algorithms are viable path planners for real-world multi-agent systems.

Table 1 Minimum distance (m) between agent pairs (DMA-RRT)

Agent	1	2	3	4
5	0.720	0.554	0.436	0.732
4	0.586	0.417	0.476	–
3	0.365	0.505	–	–
2	0.509	–	–	–

Table 2 Minimum distance (m) between agent pairs (Cooperative DMA–RRT)

Agent	1	2	3	4
5	0.343	0.344	0.528	0.312
4	0.564	0.641	0.441	–
3	0.313	0.308	–	–
2	0.565	–	–	–

9 Conclusions and future work

This paper presented two new multi-agent path planning strategies capable of handling complex environments and agent dynamics: the Decentralized Multi-Agent Rapidly-exploring Random Tree (DMA-RRT) and Cooperative DMA-RRT. These algorithms extend the CL-RRT path planner to multiple agents by introducing a merit-based token passing coordination strategy to improve team performance while guaranteeing that all constraints (such as collision avoidance) are satisfied. A cooperation strategy is also introduced to further improve team performance and avoid certain common deadlock scenarios by allowing agents to modify teammates' paths while continuing to satisfy all constraints. The strength of these algorithms lies in the use of the RRT tree to quickly identify and compare the cost of potential paths for each agent. This allows an agent to update its plans sooner than others if it has a greater incentive to do so and also enables it to truncate other agents' paths to reduce the overall team cost. The benefits of the coordination and cooperation strategies are verified through several simulation and hardware results, demonstrating their utility as real-world planners.

There are several directions for future work to improve these algorithms. The primary challenge to address is that of avoiding deadlock scenarios, especially with many agents navigating cluttered or narrow environments. Although Cooperative DMA-RRT avoids common pair-wise deadlock scenarios, there are other causes of deadlock stemming from the decentralized, finite-horizon nature of the planners, such as multiple robots entering a dead-end hallway. Extending DMA-RRT to build a roadmap online based on all the shared waypoint data may help avoid these deadlock scenarios. This roadmap of feasible paths through the environment could be used to identify challenging regions and find paths that clear these regions. DMA-RRT could also be extended to account for communication limits (see Sect. 4.1) and dynamic obstacles (see Sect. 6.4). Larger simulations and field tests could also be conducted to further validate these algorithms in practical scenarios.

Acknowledgements The authors would like to thank Brandon Luders and Georges Aoude for their contributions to this research effort. Research funded in part by the Office of Secretary of Defense under Air Force Contract FA8721-05-C-0002.

References

Amir, Y., Moser, L., Melliar-Smith, P., Agarwal, D., & Ciarfella, P. (1993). Fast message ordering and membership using a logical token-passing ring. In *Proceedings the 13th international conference on distributed computing systems* (pp. 551–560). New York: IEEE Press.

Aoude, G. S., Luders, B. D., Levine, D. S., & How, J. P. (2010). Threat-aware path planning in uncertain urban environments. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Taipei, Taiwan (pp. 6058–6063).

Ayanian, N., & Kumar, V. (2010). Decentralized feedback controllers for multiagent teams in environments with obstacles. *IEEE Transactions on Robotics*, *26*(5), 878–887.

Azarm, K., & Schmidt, G. (1997). Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. In *IEEE international conference on robotics and automation* (Vol. 4, pp. 3526–3533).

Banerjee, S., & Chrysanthis, P. (1996). A new token passing distributed mutual exclusion algorithm. In *Proceedings of the 16th international conference on distributed computing systems* (pp. 717–724). New York: IEEE Press.

Barraquand, J., Langlois, B., & Latombe, J. C. (1992). Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*(2), 224–241.

Basar, T. (1988). Asynchronous algorithms in non-cooperative games. *Journal of Economic Dynamics and Control*, *12*(1), 167–172.

Bertsekas, D., & Gallager, R. (1992). *Data networks*. Englewood Cliffs: Prentice-Hall.

Chun, L., Zheng, Z., & Chang, W. (1999). A decentralized approach to the conflict-free motion planning for multiple mobile robots. In *IEEE international conference on robotics and automation (ICRA)* (Vol. 2, pp. 1544–1549).

Desaraju, V., Ro, H. C., Yang, M., Tay, E., Roth, S., & Del Vecchio, D. (2009). Partial order techniques for vehicle collision avoidance: application to an autonomous roundabout test-bed. In *IEEE international conference on robotics and automation (ICRA)* (pp. 82–87).

Dunbar, W., & Murray, R. (2006). Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, *42*(4), 549–558.

Erdmann, M., & Lozano-Pérez, T. (1987). On multiple moving objects. *Algorithmica*, *2*, 477–521.

Frazzoli, E., Dahleh, M. A., & Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, *25*(1), 116–129.

Hoffmann, G., & Tomlin, C. (2008). Decentralized cooperative collision avoidance for acceleration constrained vehicles. In *IEEE conference on decision and control (CDC)* (pp. 4357–4363).

How, J. P., Bethke, B., Frank, A., Dale, D., & Vian, J. (2008). Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, *28*(2), 51–64.

Inalhan, G., Stipanovic, D., & Tomlin, C. (2002). Decentralized optimization, with application to multiple aircraft coordination. In *IEEE conference on decision and control (CDC)*, Las Vegas, NV.

Jager, M., & Nebel, B. (2001). Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (Vol. 3, pp. 1213–1219).

Johnson, E., Tang, Z., Balakrishnan, M., Rubio, J., Zhang, H., & Sreepuram, S. (2003). Robust token management for unreliable networks. In *Military communications conference. MILCOM 2003* (Vol. 1, pp. 399–404). New York: IEEE Press.

Karaman, S., & Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. In *Robotics: science and systems (RSS)*.

Karaman, S., Walter, M., Perez, A., Frazzoli, E., & Teller, S. (2011). Anytime motion planning using the rrt*. In *Proceedings of the IEEE international conference on robotics and automation, institute of electrical and electronics engineers*.

Kavraki, L. E., Svestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, *12*(4), 566–580.

Keviczky, T., Borrelli, F., & Balas, G. J. (2004). A study on decentralized receding horizon control for decoupled systems. In *American control conference (ACC)*, Boston, MA (pp. 4921–4926).

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE international conference on robotics and automation* (Vol. 2, pp. 500–505).

Kim, J., & Kim, C. (1997). A total ordering protocol using a dynamic token-passing scheme. *Distributed Systems Engineering*, *4*, 87.

Koenig, S., & Likhachev, M. (2002). D^* lite. In *Proceedings AAAI national conference on artificial intelligence* (pp. 476–483).

Koenig, S., Likhachev, M., & Furcy, D. (2004). Lifelong planning A*. *Artificial Intelligence*, *155*(1–2), 93–146.

Koren, Y., & Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE international conference on robotics and automation* (pp. 1398–1404).

Kuwata, Y., & How, J. P. (2011). Cooperative distributed robust trajectory optimization using receding horizon MILP. *IEEE Transactions on Control Systems Technology*, *19*(2), 423–431.

Kuwata, Y., Richards, A., Schouwenaars, T., & How, J. P. (2007). Distributed robust receding horizon control for multivehicle guidance. *IEEE Transactions on Control Systems Technology*, *15*(4), 627–641.

Kuwata, Y., Teo, J., Karaman, S., Fiore, G., Frazzoli, E., & How, J. P. (2008). Motion planning in complex environments using closed-loop prediction. In *AIAA guidance, navigation, and control conference (GNC) (AIAA-2008-7166)*, Honolulu, HI.

Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., & How, J. P. (2009). Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, *17*(5), 1105–1118.

LaValle, S. M. (1998). *Rapidly-exploring random trees: a new tool for path planning* (Tech. Rep. 98–11). Iowa State University.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge: Cambridge University Press.

Lee, J., Nam, H., & Lyou, J. (1995). A practical collision-free trajectory planning for two robot systems. In *Proceedings of the IEEE international conference on robotics and automation* (Vol. 3, pp. 2439–2444). New York: IEEE Press.

Leonard, N., Paley, D., Lekien, F., Sepulchre, R., Fratantoni, D., & Davis, R. (2007). Collective motion, sensor networks, and ocean sampling. *Proceedings of the IEEE*, *95*(1), 48–74.

Leonard, J., How, J. P., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M., Barrett, D., Epstein, A., Maheloni, K., Moyer, K., Jones, T., Buckley, R., Antone, M., Galejs, R., Krishnamurthy, S., & Williams, J. (2008). A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, *25*(10), 727–774.

Likhachev, M., & Stentz, A. (2008). R* search. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 344–350).

Luders, B., Karaman, S., Frazzoli, E., & How, J. P. (2010). Bounds on track error using closed-loop rapidly-exploring random trees. In *American control conference (ACC)*, Baltimore (pp. 5406–5412).

Mueller, F. (2001). Fault tolerance for token-based synchronization protocols. In *Proceedings 15th international parallel and distributed processing symposium* (pp. 1257–1264). New York: IEEE Press.

Murray, R. (2007). Recent research in cooperative control of multivehicle systems. *ASME Journal of Dynamic Systems, Measurement, and Control*, *129*(5), 571–583.

Narváez, P., Siu, K., & Tzeng, H. (2000). New dynamic algorithms for shortest path tree computation. *IEEE/ACM Transactions on Networking*, *8*(6), 734–746.

Olfati-Saber, R., Fax, J., & Murray, R. (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, *95*(1), 215–233.

Pallottino, L., Scordio, V., & Bicchi, A. (2004). Decentralized cooperative conflict resolution among multiple autonomous mobile agents. In *IEEE conference on decision and control (CDC)* (Vol. 5, pp. 4758–4763).

Parker, L. (2002). Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, *12*(3), 231–255.

Purwin, O., D'Andrea, R., & Lee, J. (2008). Theory and implementation of path planning by negotiation for decentralized agents. *Robotics and Autonomous Systems*, *56*(5), 422–436.

Richards, A. G. (2005). *Robust constrained model predictive control*. Ph.D. thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA.

Richards, A., Schouwenaars, T., How, J. P., & Feron, E. (2002). Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *Journal of Guidance, Control, and Dynamics*, *25*(4), 755–764.

Sae-Hau, C. (2003). *Multi-vehicle rover testbed using a new indoor positioning sensor*. S.M. thesis draft, Massachusetts Institute of Technology.

Scerri, P., Owens, S., Yu, B., & Sycara, K. (2007). A decentralized approach to space deconfliction. In *Proc. 10th int. information fusion conf.* (pp. 1–8).

Schouwenaars, T., How, J., & Feron, E. (2004). Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees. In *AIAA guidance, navigation, and control conference (GNC)*, Providence, RI.

Si, J. (2004). *Handbook of learning and approximate dynamic programming* (Vol. 2). New York: Wiley–IEEE Press.

Swigart, J., & Lall, S. (2010). An explicit dynamic programming solution for a decentralized two-player optimal linear-quadratic regulator. In *Proceedings of mathematical theory of networks and systems*.

Tanner, H., Loizou, S., & Kyriakopoulos, K. (2001). Nonholonomic stabilization with collision avoidance for mobile robots. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (Vol. 3, pp. 1220–1225). New York: IEEE Press.

Teller, S., Walter, M. R., Antone, M., Correa, A., Davis, R., Fletcher, L., Frazzoli, E., Glass, J., How, J. P., Huang, A. S., Jeon, J., Karaman, S., Luders, B., Roy, N., & Sainath, T. (2010). A voice-commanded robotic forklift working alongside humans in minimally-prepared outdoor environments. In *Proceedings of the IEEE international conference on robotics and automation*, Anchorage, AK.

Trodden, P. (2009). *Robust distributed control of constrained linear systems*. Ph.D. thesis, University of Bristol.

Trodden, P., & Richards, A. (2006). Robust distributed model predictive control using tubes. In *Proceedings of the American control conference*, Minneapolis, MN (pp. 2034–2039).

van den Berg, J., Snoeyink, J., Lin, M., & Manocha, D. (2009). Centralized path planning for multiple robots: optimal decoupling into sequential plans. In *Proceedings of robotics: science and systems*, Seattle, USA.

Velagapudi, P., Sycara, K., & Scerri, P. (2010). Decentralized prioritized planning in large multirobot teams. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4603–4609).

Venkat, A. N., Rawlings, J. B., & Wright, S. J. (2005). Stability and optimality of distributed model predictive control. In *Proceedings of the IEEE conference on decision and control*, Seville, Spain (pp. 6680–6685).

Wang, P., & Zhuang, W. (2008). A token-based scheduling scheme for wlans supporting voice/data traffic and its performance analysis. *IEEE Transactions on Wireless Communications*, *7*(5), 1708–1718.

Yershova, A., Jaillet, L., Siméon, T., & LaValle, S. M. (2005). Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *IEEE international conference on robotics and automation (ICRA)*, Barcelona, Spain (pp. 3856–3861).

Vishnu R. Desaraju received the B.S.E. degree in Electrical Engineering from the University of Michigan in 2008 and the S.M. degree in Aeronautics and Astronautics from MIT in 2010. From 2009 to 2010, he worked on the Agile Robotics project to develop semi-autonomous field robotics for the US Army. Research interests include real-time motion planning, multi-agent control, adaptive control, and collision avoidance strategies. He is a member of IEEE and HKN.

Jonathan P. How is the Richard Maclaurin Professor of Aeronautics and Astronautics at MIT. He received a B.A.Sc. from the U. of Toronto in 1987 and his S.M. and Ph.D. in Aeronautics and Astronautics from MIT in 1990 and 1993. Prior to joining MIT in 2000, he was an Assistant Professor at Stanford University. Research interests include robust coordination and control of autonomous vehicles. He is an Associate Fellow of AIAA, and a senior member of IEEE.