Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Multi-Robot Motion Planning with Dynamics Guided by Multi-Agent Search*

Duong Le and **Erion Plaku**
Department of Electrical Engineering and Computer Science, Catholic University of America
Washington DC, USA
{05le, plaku}@cua.edu

Abstract

This paper presents an effective multi-robot motion planner that enables each robot to reach its desired location while avoiding collisions with the other robots and the obstacles. The approach takes into account the differential constraints imposed by the underlying dynamics of each robot and generates dynamically-feasible motions that can be executed in the physical world. The crux of the approach is the sampling-based expansion of a motion tree in the continuous state space of all the robots guided by multi-agent search over a discrete abstraction. Experiments using vehicle models with nonlinear dynamics operating in complex environments show significant speedups over related work.

1 Introduction

Robotic teams provide a viable venue to enhance automation, increase productivity, and reduce operational costs in transportation, exploration, inspection, surveillance, search-and-rescue, and many other applications. In these settings, robots are often required to move to different locations while avoiding collisions with obstacles and other robots.

Multi-robot motion planning is challenging since it is no longer sufficient to plan the motions of the robots independently as the plans of one robot can interfere with the plans of another robot or even prevent it from reaching the goal. The planned motions should also obey the underlying dynamics of each robot, which impose differential constraints on the velocity, acceleration, direction, and turning radius. This is essential to ensure that the planned motions can be executed by the robots in the physical world.

Related work has considered various settings for the multi-robot motion-planning problem. In a discrete setting, i.e., multi-agent pathfinding, each robot is a point with no dynamics that in one step can move to an adjacent vertex in a graph abstraction of the world. To solve this NP-hard problem [Yu and LaValle, 2013], decoupled approaches often plan the paths for each agent one at a time, ensuring that agent i avoids conflicts with the plans of agents $1, \ldots, i-1$ [Silver, 2005; Sturtevant and Buro, 2006; Barer *et al.*, 2014]. Decoupled approaches are fast but suboptimal. To provide optimality, centralized approaches based on A* [Standley and Korf, 2011; Goldenberg *et al.*, 2014; Svancara and Surynek, 2017], M* [Wagner and Choset, 2015], increasing-cost tree search [Sharon *et al.*, 2013], and conflict-based search [Sharon *et al.*, 2015] operate over the composite space of all the agents [Felner *et al.*, 2017]. Solvers have also used auctions [Amir *et al.*, 2015], answer-set programming [Erdem *et al.*, 2013], or rules [Khorshid *et al.*, 2011; Botea and Surynek, 2015].

In a continuous setting, sampling-based motion planning has often been used. To account for dynamics, `RRT` [LaValle and Kuffner, 2001], `KPIECE` [Şucan and Kavraki, 2012], `GUST` [Plaku, 2015], and others expand a motion tree by adding collision-free and dynamically-feasible trajectories as branches. For multi-robot problems, sampling-based motion planners can be used in a decoupled or centralized framework. Decoupled approaches plan the motions one robot at a time, treating robots $1, \ldots, i-1$ as moving obstacles when planning the motions of robot i. Decoupled approaches work well in open environments but have difficulty finding solutions in cluttered environments where coordination among robots becomes necessary since the motions planned for robots $1, \ldots, i-1$ could make it impossible for robot i to reach its goal. Centralized approaches treat the robots as one system and operate over the resulting composite state space. They are probabilistically complete, but do not scale well due to the high dimensionality of the composite state space.

To develop an effective multi-robot motion planner, we present a framework that couples the ability of sampling-based motion planning to handle the complexity arising from geometric and differential constraints imposed by the obstacles and underlying robot dynamics with the ability of multi-agent pathfinding to find non-conflicting routes over discrete abstractions. The underlying idea is to use multi-agent search as a heuristic to guide sampling-based motion planning as it expands a motion tree in the composite state space of all the robots. The expansion in the composite state space allows the approach to obtain probabilistic completeness, which ensures that a solution will be found, when it exists, with probability approaching one. Scalability and efficiency are obtained by using multi-agent search as a heuristic to guide the motion-tree expansion. As with any heuristic, a critical aspect is the

*This paper is an abridged version of the paper "Cooperative Multi-Robot Sampling-Based Motion Planning with Dynamics" that won the Best Robotics Paper award at ICAPS, 2017.

design of an appropriate relaxed problem setting. We obtain the discrete abstraction by ignoring the vehicle dynamics and using probabilistic roadmaps to create a network of roads that captures the connectivity of the environment, making it easy to reach each goal from any location.

Multi-agent search and sampling-based motion planning work in tandem. At each iteration of a core loop, multi-agent search is first invoked to find non-conflicting routes over the discrete abstraction. At a second step, sampling-based motion planning seeks to expand the motion tree along the routes. When little or no progress is made, the routes are penalized and the multi-agent search is invoked again to find alternative routes. This synergistic combination makes it possible to effectively plan collision-free and dynamically-feasible motions that enable each robot to reach its goal. Experiments using vehicle models with nonlinear dynamics operating in unstructured, obstacle-rich, environments demonstrate the scalability and efficiency of the approach, yielding significant speedups over related work.

2 Problem Formulation

Each robot model $\mathcal{M}_i = \langle \mathcal{P}_i, \mathcal{S}_i, \mathcal{A}_i, f_i \rangle$ is defined by its geometric shape $\mathcal{P}_i$, state space $\mathcal{S}_i$, action space $\mathcal{A}_i$, and underlying dynamics f_i expressed as a set of differential equations

$$\dot{s} = f_i(s, a), s \in \mathcal{S}_i, a \in \mathcal{A}_i. \tag{1}$$

As an example, a vehicle model defines the state in terms of the position (x, y), orientation θ, steering angle ψ, and velocity v. The vehicle is controlled by setting the acceleration *acc* and steering rate ω. The motion equations are defined as

$$\dot{x} = v\cos(\theta)\cos(\psi), \dot{y} = v\sin(\theta)\cos(\psi), \tag{2}$$

$$\dot{\theta} = v\sin(\psi)/L, \dot{v} = acc, \dot{\psi} = \omega, \tag{3}$$

where L is the distance from the back to the front wheels.

The new state $s_{new} \in \mathcal{S}$ obtained by applying a control action $a \in \mathcal{A}_i$ to a state $s \in \mathcal{S}_i$ is computed by a function

$$s_{new} \leftarrow \text{SIMULATE}(s, a, f_i, dt), \tag{4}$$

which numerically integrates f_i for one time step dt. Starting from $s \in \mathcal{S}$ and applying actions $\langle a_1, \ldots, a_{\ell-1} \rangle$ in succession gives rise to a dynamically-feasible trajectory $\zeta : \{1, \ldots, \ell\} \rightarrow \mathcal{S}_i$, where $\zeta(1) \leftarrow s$ and $\forall j \in \{2, \ldots, \ell\}$:

$$\zeta(j) \leftarrow \text{SIMULATE}(\zeta(j-1), a_{j-1}, f_i, dt). \tag{5}$$

The multi-robot motion-planning problem is defined by

$$\langle \mathcal{W}, \mathcal{O}, \mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{M}_1, \ldots, \mathcal{M}_n, s_1^{init}, \ldots, s_n^{init} \rangle. \tag{6}$$

The world $\mathcal{W}$ contains obstacles $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_m\}$ and goals $\mathcal{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$. The objective is to compute trajectories $\zeta_1, \ldots, \zeta_n$, one for each robot, so that $\zeta_i : \{1, \ldots, \ell\} \rightarrow \mathcal{S}_i$ starts at the initial state $s_i^{init} \in \mathcal{S}_i$, reaches the goal $\mathcal{G}_i$, and avoids collisions with the obstacles and the other robots.

3 Method

The approach has several components: (i) a discrete abstraction based on a relaxed problem setting; (ii) multi-agent search over the discrete abstraction to guide sampling-based motion planning; (iii) expansion of a motion tree in the composite state space of all the robots; (iv) using the discrete abstraction to partition the motion tree into equivalence classes; and (v) interplay of sampling-based motion planning and multi-agent search to effectively expand the motion tree along routes obtained by the multi-agent search. A schematic representation is shown in Fig. 1.

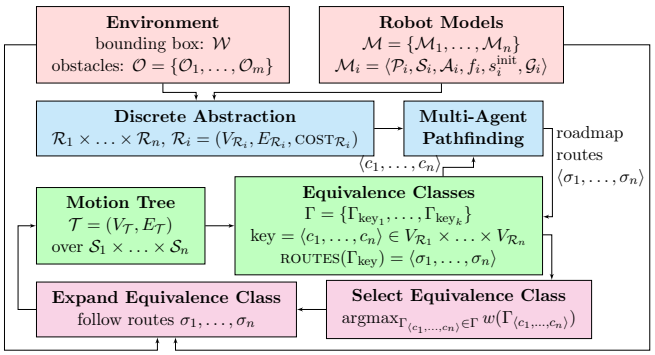

Figure 1: Schematic illustration of the approach.

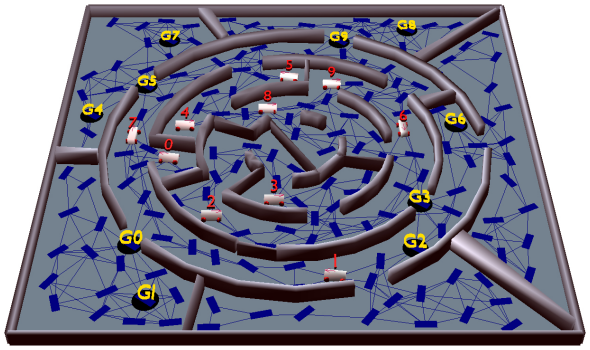

Figure 2: An example of a roadmap.

3.1 Discrete Abstraction via Roadmaps

The discrete abstraction provides a relaxed problem representation, obtained by ignoring the robot dynamics. Specifically, the robot state is reduced to just the position and orientation (referred to as configuration) and each robot is allowed to translate and rotate in any direction. To solve the multi-robot motion-planning problem in this relaxed setting, a roadmap $\mathcal{R}_i = (V_{\mathcal{R}_i}, E_{\mathcal{R}_i})$ is constructed for each robot $\mathcal{M}_i$, seeking to make it easy to reach each goal from any location in the environment. The roadmap is initially populated by adding collision-free configurations from each goal $\mathcal{G}_1, \ldots, \mathcal{G}_n$. Leveraging PRM [Kavraki *et al.*, 1996], the roadmap is further populated by sampling collision-free configurations and connecting neighboring configurations with collision-free paths whenever possible, as shown in Fig. 2. Dense roadmaps are preferred in order to provide alternative routes along which the robots can move to reach their goals. Robots that have the same shape can use the same roadmap.

3.2 Multi-Agent Search over the Discrete Abstraction

MULTIAGENTSEARCH$(\mathcal{R}_1, \ldots, \mathcal{R}_n, c_1, \ldots, c_n, \mathcal{G}_1, \ldots, \mathcal{G}_n)$ searches over the roadmaps $\mathcal{R}_1, \ldots, \mathcal{R}_n$ to compute non-conflicting routes $\sigma_1, \ldots, \sigma_n$ for each robot such that σ_i is over $\mathcal{R}_i$, starts at c_i, and ends at a roadmap vertex associated with the goal $\mathcal{G}_i$. Since in multi-agent search, each robot moves from one vertex to the next in one step, each roadmap edge is divided into several parts to ensure that the distance from one vertex to the next is no more than a user-specified step size. The edge division is done only once after each roadmap is constructed. Note that the roadmap construction guarantees that robots will not collide with the obstacles as they move from one roadmap vertex to the next. Thus, the multi-agent search has to avoid only robot-robot conflicts. The overall approach is agnostic to the inner workings of the multi-agent search, so it can be used in conjunction with any available method that operates over graphs.

3.3 Motion Tree in the Composite State Space

Note that solutions over the relaxed problem setting do not necessarily correspond to dynamically-feasible trajectories since the discrete abstraction ignores the robot dynamics. As a result, geometric and differential constraints imposed by the obstacles and the underlying dynamics may make it difficult or impossible for a robot to follow its route σ_i.

To account for the dynamics, a motion tree $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ is rooted at the initial state $\langle s_1^{init}, \ldots, s_n^{init} \rangle$ and is incrementally expanded in the composite state space $\mathcal{S}_1 \times \ldots \times \mathcal{S}_n$. Thus, each vertex $v \in V_\mathcal{T}$ corresponds to a composite state, denoted by STATE(v), where STATE$_i(v)$ represents the state associated with the i-th robot. By construction, v is added to $V_\mathcal{T}$ only if STATE(v) is collision-free, i.e., when the robots are placed according to STATE(v), there is no robot-robot or robot-obstacle collision. To generate a successor v' from v, some control actions $\langle a_1, \ldots, a_n \rangle$ are applied component-wise to STATE(v) and the motion equations of each robot are integrated for one time step dt, i.e., $\forall i \in \{1, \ldots, n\}$: STATE$_i(v') \leftarrow$ SIMULATE(STATE$_i(v), a_i, f_i, dt$). When STATE(v') is not in collision, v' and the edge (v, v') are added to the motion tree $\mathcal{T}$.

The motion tree $\mathcal{T}$ is expanded until a vertex v_{new} is added where each robot has reached its goal, i.e., $\forall i \in \{1, \ldots, n\}$: STATE$_i(v_{new}) \in \mathcal{G}_i$. The path $\langle v_1, \ldots, v_\ell \rangle$, with $v_1 = v_{init}$ and $v_\ell = v_{new}$, from the root of $\mathcal{T}$ to v_{new} is retrieved and the solution trajectory for each robot i is constructed as $\zeta_i = \langle$STATE$_i(v_1), \ldots,$ STATE$_i(v_\ell) \rangle$.

3.4 Partitioning the Motion Tree into Equivalence Classes Based on the Discrete Abstraction

The motion tree $\mathcal{T}$ could be expanded in an uninformed way by selecting at each iteration a vertex $v \in V_\mathcal{T}$ at random and applying controls to generate a successor v' from v. Such an expansion, however, would have difficulty finding solutions.

In contrast, we introduce multi-agent search to guide the motion-tree expansion. Since multi-agent search operates over the discrete abstraction, a mapping from the composite state space to the discrete abstraction must be first defined. As the discrete abstraction is obtained via roadmaps, a robot state $s_i \in \mathcal{S}_i$ is mapped to the nearest configuration in the roadmap $\mathcal{R}_i$. In this way, a composite state $\langle s_1, \ldots, s_n \rangle \in \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$ is mapped component-wise to a composite configuration $\langle c_1, \ldots, c_n \rangle$, i.e.,

$$\text{MAP}(\langle s_1, \ldots, s_n \rangle) = \langle c_1, \ldots, c_n \rangle, \tag{7}$$

where $c_i =$ NEAREST$(\mathcal{R}_i, s_i)$.

The approach leverages this mapping to partition the motion tree $\mathcal{T}$ into equivalence classes. The premise is that vertices in $\mathcal{T}$ that provide the same discrete information should belong to the same equivalence class. In other words, vertices $v, v' \in \mathcal{T}$ belong to the same equivalence class $\Gamma_{\langle c_1, \ldots, c_n \rangle}$ if and only if MAP(STATE(v)) = MAP(STATE(v')) = $\langle c_1, \ldots, c_n \rangle$. So, an equivalence class $\Gamma_{\langle c_1, \ldots, c_n \rangle}$ contains all the vertices in $\mathcal{T}$ that map to $\langle c_1, \ldots, c_n \rangle$.

The significance of an equivalence class $\Gamma_{\langle c_1, \ldots, c_n \rangle}$ is that it makes it possible to leverage multi-agent search to compute non-conflicting routes $\sigma_1, \ldots, \sigma_n$, where each σ_i starts at c_i and reaches the goal $\mathcal{G}_i$. These routes then serve to guide the motion planner as it seeks to expand the motion tree from vertices associated with the equivalance class $\Gamma_{\langle c_1, \ldots, c_n \rangle}$ along $\sigma_1, \ldots, \sigma_n$.

3.5 Putting it all Together: Using Multi-Agent Search to Guide the Motion-Tree Expansion

The overall approach starts by creating the discrete abstraction, rooting the motion tree $\mathcal{T}$ at the initial state, and creating the first equivalence class $\Gamma_{\text{STATE}(v_{init})}$, which contains the root vertex v_{init}. Afterwards, the approach enters a core loop, where each iteration consists of the following:

- select an equivalence class $\Gamma_{\langle c_1, \ldots, c_n \rangle}$ from Γ;
- use multi-agent search over the discrete abstraction to obtain non-conflicting routes $\sigma_1, \ldots, \sigma_n$ that reach the goals starting from $c_1, \ldots, c_n$; and
- expand the motion tree $\mathcal{T}$ from vertices in $\Gamma_{\langle c_1, \ldots, c_n \rangle}$ along the routes $\sigma_1, \ldots, \sigma_n$.

MULTIAGENTSEARCH$(\mathcal{R}_1, \ldots, \mathcal{R}_n, c_1, \ldots, c_n, \mathcal{G}_1, \ldots, \mathcal{G}_n)$ searches the discrete abstraction to obtain non-conflicting routes $\langle \sigma_1, \ldots, \sigma_n \rangle$, where σ_i starts at c_i and reaches the goal $\mathcal{G}_i$. To save computation time, the routes are stored with $\Gamma_{\langle c_1, \ldots, c_n \rangle}$ and retrieved when the multi-agent search is invoked again for $\Gamma_{\langle c_1, \ldots, c_n \rangle}$. During the selection of an equivalence class, priority is given to equivalence classes associated with short routes to promote rapid expansions toward the goals. The expansion adds new vertices to $\mathcal{T}$, which could result in creating new equivalence classes. When the expansion from $\Gamma_{\langle c_1, \ldots, c_n \rangle}$ fails or has difficulty following the routes $\sigma_1, \ldots, \sigma_n$, which could happen due to the geometric and differential constraints imposed by the obstacles and the underlying robot dynamics, the approach penalizes $\Gamma_{\langle c_1, \ldots, c_n \rangle}$ in order to promote expansions from other equivalence classes. This process of selecting and expanding from an equivalence class is repeated until a solution is found or a runtime limit is reached.

Selecting an Equivalence Class

A weight is defined for each equivalence class and the equivalence class with the maximum weight is selected for expan-

sion. Specifically, the weight for $\Gamma_{\langle c_1,\ldots,c_n\rangle}$ is defined as

$$w(\Gamma_{\langle c_1,\ldots,c_n\rangle}) = \frac{\alpha^{\text{NRSEL}(\Gamma_{\langle c_1,\ldots,c_n\rangle})}}{\sum_{i=1}^{n}(\text{COST}(\sigma_i))^2}, \quad (8)$$

where $\sigma_1,\ldots,\sigma_n$ denote the routes associated with $\Gamma_{\langle c_1,\ldots,c_n\rangle}$, $0 < \alpha < 1$, and $\text{NRSEL}(\Gamma_{\langle c_1,\ldots,c_n\rangle})$ denotes the number of times $\Gamma_{\langle c_1,\ldots,c_n\rangle}$ has been previously selected.

In this way, to promote expansions towards the goals, preference is given to equivalence classes associated with short routes. This constitutes the greedy aspect of the selection process. To mitigate the potential pitfalls of a greedy selection, a penalty factor α ($0 < \alpha < 1$) is introduced, which reduces the weight of an equivalence class each time it is selected. In fact, repeatedly selecting $\Gamma_{\langle c_1,\ldots,c_n\rangle}$ will continue to reduce its weight so that eventually some other equivalence class will end up having a greater weight and thus be selected for expansion. This is essential to ensure probabilistic completeness and avoid becoming stuck when the motion-tree expansion from $\Gamma_{\langle c_1,\ldots,c_n\rangle}$ repeatedly fails due to the constraints imposed by the obstacles and the underlying robot dynamics.

Expanding the Motion Tree from the Equivalence Class
After selecting an equivalence class $\Gamma_{\langle c_1,\ldots,c_n\rangle}$, the objective becomes to expand the motion tree $\mathcal{T}$ from a vertex $v \in \Gamma_{\langle c_1,\ldots,c_n\rangle}$ along the routes $\sigma_1,\ldots,\sigma_n$ associated with $\Gamma_{\langle c_1,\ldots,c_n\rangle}$. The vertex v could be selected at random to promote expansions from different vertices. Attempts are then made to expand $\mathcal{T}$ from $\text{STATE}(v)$ so that each robot i moves from $\text{STATE}_i(v)$ toward the next configuration in σ_i. We can use proportional-integrative-derivative (PID) controllers [Spong *et al.*, 2005] to select controls that would steer each robot toward its target, e.g., by turning the wheels and then moving toward the target. The expansion from v continues for several steps until the target is reached or a robot collides with an obstacle or another robot. If a new state is in collision, the trajectory generation stops and the approach goes back to selecting an equivalence class. Otherwise, the new state is added to tree $\mathcal{T}$ and the partition Γ is updated accordingly. If in the new state each robot has reached its goal, then a solution is found and the search terminates successfully.

4 Experiments and Results

Experiments are conducted using vehicle models with nonlinear dynamics operating in complex environments, as shown in Fig. 2 and 3. In many of these scenarios, the robots are required to cooperate in order to reach their goals.

The approach is compared to a centralized and a prioritized version of RRT [LaValle and Kuffner, 2001], denoted by cRRT and pRRT, as RRT is one of the most popular sampling-based motion planners. The approach is also compared to a prioritized version of GUST [Plaku, 2015], denoted by pGUST, as GUST has been shown to be significantly faster than RRT and other sampling-based motion planners.

Results in Fig. 4 show significant speedups over cRRT, pRRT, and pGUST. As RRT-based planners lack guidance, they have difficulty finding solutions. pGUST is faster than cRRT and pRRT since it uses discrete search to guide the motion-tree expansion. In open scenes (scene 2), where cooperation among the robots is not essential, pGUST is even faster than our approach since the robot trajectories do not interfere with each other. In scenes where cooperation is important, pGUST, as any prioritized approach, has difficulty finding solutions, even timing out in scenes 4, 5, 6. In contrast, due to the guidance by multi-agent search, our approach is able to find solutions in a matter of 1-3s.

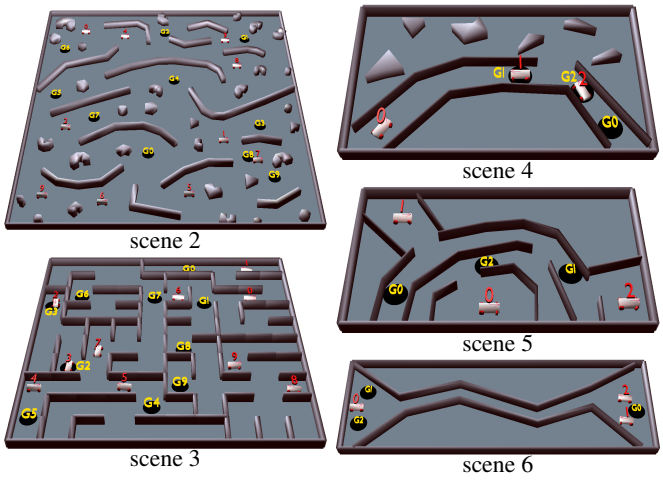

Figure 3: Scenes used in the experiments (scene 1 shown in Fig. 2). Videos of solutions can be found at https://goo.gl/sQ6irb. Figure best viewed in color and on screen.

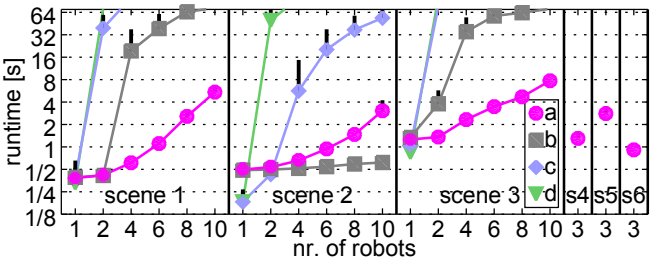

Figure 4: Runtime results when comparing (a) our approach to (b) pGUST, (c) pRRT, and (d) cRRT. Runtime measures everything from reading the input file to reporting that a solution is found (including for our approach the time to build the roadmaps). Missing entries indicate failure by the planner to solve the problem instances within the runtime limit (set to 90s per run). Note in particular that only our approach could solve scenes 4, 5, 6 (denoted as s4, s5, s6).

5 Discussion

This paper presented an effective multi-robot motion planner that enables each robot to reach its desired location while avoiding collisions. The approach also took into account the dynamics of each robot and generated dynamically-feasible trajectories that can be executed in the physical world.

The crux of the approach was coupling the ability of sampling-based motion planning to handle the complexity arising from obstacles and robot dynamics with the ability of multi-agent pathfinding to find non-conflicting routes over discrete abstractions. The underlying idea was to use multi-agent search over the discrete abstraction as a heuristic to guide sampling-based motion planning as it expands a mo-

tion tree in the composite state space of all the robots.

This work opens up several research directions. Advances in multi-agent search can significantly improve the efficiency and scalability of the framework by reducing the runtime to compute the routes. Moreover, the quality of the routes can also be improved, for example, by requiring multi-agent search to compute non-conflicting routes that maximize the separation among the robots and the clearance from the obstacles. Such routes would be easier to follow, which would reduce the runtime for the motion-tree expansion. Another direction is to enable the team of robots to perform complex tasks given by PDDL or some other high-level formalism.

Acknowledgments

This work is supported by NSF IIS1548406.

References

[Amir *et al.*, 2015] Ofra Amir, Guni Sharon, and Roni Stern. Multi-agent pathfinding as a combinatorial auction. In *AAAI Conference on Artificial Intelligence*, pages 2003–2009, 2015.

[Barer *et al.*, 2014] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Symposium on Combinatorial Search*, pages 19–27, 2014.

[Botea and Surynek, 2015] Adi Botea and Pavel Surynek. Multi-agent path finding on strongly biconnected digraphs. In *AAAI Conference on Artificial Intelligence*, pages 2024–2030, 2015.

[Şucan and Kavraki, 2012] Ioan A. Şucan and Lydia E. Kavraki. A sampling-based tree planner for systems with complex dynamics. *IEEE Transactions on Robotics*, 28(1):116–131, 2012.

[Erdem *et al.*, 2013] Esra Erdem, Doga Gizem Kisa, Umut Öztok, and Peter Schueller. A general formal framework for pathfinding problems with multiple agents. In *AAAI Conference on Artificial Intelligence*, pages 290–296, 2013.

[Felner *et al.*, 2017] Ariel Felner, Roni Stern, E Shimony, Eli Boyarski, M Goldenerg, Guni Sharon, Nathan R Sturtevant, Glenn Wagner, and Pavel Surynek. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Symposium on Combinatorial Search*, pages 29–37, 2017.

[Goldenberg *et al.*, 2014] Meir Goldenberg, Ariel Felner, Roni Stern, Guni Sharon, Nathan R Sturtevant, Robert C Holte, and Jonathan Schaeffer. Enhanced partial expansion a. *Journal of Artificial Intelligence Research*, 50:141–187, 2014.

[Kavraki *et al.*, 1996] Lydia E. Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[Khorshid *et al.*, 2011] Mokhtar M Khorshid, Robert C Holte, and Nathan R Sturtevant. A polynomial-time algorithm for non-optimal multi-agent pathfinding. In *Symposium on Combinatorial Search*, pages 76–83, 2011.

[LaValle and Kuffner, 2001] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[Plaku, 2015] Erion Plaku. Region-guided and sampling-based tree search for motion planning with dynamics. *IEEE Transactions on Robotics*, 31:723–735, 2015.

[Sharon *et al.*, 2013] Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195:470–495, 2013.

[Sharon *et al.*, 2015] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[Silver, 2005] David Silver. Cooperative pathfinding. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 1, pages 117–122, 2005.

[Spong *et al.*, 2005] Mark W. Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot Modeling and Control*. John Wiley and Sons, 2005.

[Standley and Korf, 2011] Trevor Standley and Richard Korf. Complete algorithms for cooperative pathfinding problems. In *International Joint Conference on Artificial Intelligence*, pages 668–673, 2011.

[Sturtevant and Buro, 2006] Nathan R Sturtevant and Michael Buro. Improving collaborative pathfinding using map abstraction. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 80–85, 2006.

[Svancara and Surynek, 2017] Jiri Svancara and Pavel Surynek. New flow-based heuristic for search algorithms solving multi-agent path finding. In *International Conference on Agents and Artificial Intelligence*, pages 451–458, 2017.

[Wagner and Choset, 2015] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.

[Yu and LaValle, 2013] Jingjin Yu and Steven M LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI Conference on Artificial Intelligence*, pages 1443–1449, 2013.