Article

The International Journal of Robotics Research
2016, Vol. 35(5) 501–513

DOI: 10.1177/0278364915615688
ijr.sagepub.com

Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning

Kiril Solovey*, Oren Salzman* and Dan Halperin

Abstract

We present a sampling-based framework for multi-robot motion planning. which combines an implicit representation of roadmaps for multi-robot motion planning with a novel approach for pathfinding in geometrically embedded graphs tailored for our setting. Our pathfinding algorithm, discrete-RRT (dRRT)*, is an adaptation of the celebrated RRT algorithm for the discrete case of a graph, and it enables a rapid exploration of the high-dimensional configuration space by carefully walking through an implicit representation of the tensor product of roadmaps for the individual robots. We demonstrate our approach experimentally on scenarios that involve as many as 60 degrees of freedom and on scenarios that require tight coordination between robots. On most of these scenarios our algorithm is faster by a factor of at least 10 when compared to existing algorithms that we are aware of.*

Keywords

Multi-robot motion planning, sampling-based motion planning

1. Introduction

Multi-robot motion planning is a fundamental problem in robotics and has been extensively studied for several decades. In this work we are concerned with finding paths for a group of robots, operating in the same workspace, moving from start to target positions while avoiding collisions with obstacles, as well as with each other. We consider the continuous formulation of the problem, where the robots and obstacles are geometric entities and the robots operate in a configuration space, e.g. $\mathbb{R}^d$ (as opposed to the discrete variant, sometimes called the *pebble motion* problem, where the robots move on a graph (Auletta et al., 1996; Goraly and Hassin, 2010; Kornhauser, 1984; Luna and Bekris, 2011)). Moreover, we assume that each robot has its own start and target positions, as opposed to the unlabeled case (Adler et al., 2015; Kloder and Hutchinson, 2005; Solovey and Halperin, 2014, 2015; Solovey et al., 2015; Turpin et al., 2014).

1.1. Previous work

We assume familiarity with the basic terminology of motion planning. For background, see Choset et al. (2005) or LaValle (2006). Initial work on motion planning aimed to develop *complete* algorithms, which guarantee to find a solution when one exists or report that none exists otherwise. Such algorithms for the multi-robot case exist (Schwartz and Sharir, 1983; Sharir and Sifrony, 1991; Yap, 1984), yet are exponential in the number of robots. The exponential running time, which may be unavoidable (Hearn and Demaine, 2005; Hopcroft et al., 1984; Solovey et al., 2015; Spirakis and Yap, 1984) can be attributed to the high number of *degrees of freedom* (*DOFs*)—the sum of the DOFs of the individual robots. Aronov et al. (1999) showed that for two or three robots, the number of DOFs may be slightly reduced by constructing a path where the robots move while maintaining contact with each other. A more general approach to reduce the number of DOFs was suggested by van den Berg et al. (2009). In their work, the multi-robot motion-planning problem is decomposed into subproblems, each consisting of a subset of robots, where every subproblem can be solved separately and the results can be combined into a solution for the original problem.

Decoupled planners are an alternative to complete planners trading completeness for efficiency. Typically, decoupled planners solve separate problems for individual robots and combine the individual solutions into a global solution (Leroy et al., 1999; van den Berg and Overmars, 2005).

*Kiril Solovey and Oren Salzman contributed equally to this paper.
Blavatnik School of Computer Science, Tel-Aviv University, Israel

Corresponding author:
Kiril Solovey, Blavatnik School of Computer Science, Tel-Aviv University, PO Box 39040, Tel-Aviv 69978, Israel.
Email: kirilsol@post.tau.ac.il

Although efficient in some cases, the approach usually works only for a restricted set of problems.

The introduction of *sampling-based* algorithms such as the *probabilistic roadmap method (PRM)* by Kavraki et al. (1996), *rapidly-exploring random trees (RRT)* by LaValle and Kuffner (1999), and their many variants, had a significant impact on the field of motion planning due to their efficiency, simplicity and applicability to a wide range of problems. Sampling-based algorithms attempt to capture the connectivity of the *configuration space (C-space)* by sampling collision-free configurations and constructing a *roadmap*—a graph data structure where the free configurations are vertices and the edges represent collision-free paths between nearby configurations. Although these algorithms are not complete, most of them are *probabilistically complete*, that is, they are guaranteed to find a solution, if one exists, given a sufficient amount of time. Recently, Karaman and Frazzoli (2011) introduced several variants of these algorithms such that, with high probability, they produce paths that are *asymptotically optimal* with respect to some quality measure.

Sampling-based algorithms can be easily extended to the multi-robot case by considering the fleet of robots as one composite robot (Sanchez and Latombe, 2002). Such a naïve approach suffers from inefficiency as it overlooks aspects that are unique to the multi-robot problem. More tailor-made sampling-based techniques have been proposed for the multi-robot case by Hirsch and Halperin (2002), Salzman et al. (2015a) and Solovey and Halperin (2014). Particularly relevant to our efforts is the work of Švestka and Overmars (1998) who suggested constructing a *composite* roadmap which is a Cartesian product of roadmaps of the individual robots. Due to the exponential nature of the resulting roadmap, this technique is only applicable to problems that involve a modest number of robots. Recent work by Wagner and Choset (2015) suggests that the composite roadmap does not necessarily have to be explicitly represented. Instead, they maintain an implicitly represented composite roadmap, and apply their M* algorithm to efficiently retrieve paths, while minimizing the explored portion of the roadmap. The resulting technique is able to cope with a large number of robots, for certain types of scenarios. Additional information on these two approaches is provided in Section 2.

1.2. Contribution

We present a sampling-based algorithm for the multi-robot motion-planning problem called *multi-robot discrete RRT (MRdRRT)*. Similar to the approach of Wagner and Choset (2015), we maintain an implicit representation of the composite roadmap. We propose an alternative highly efficient technique for pathfinding in the roadmap, which can cope with scenarios that involve tight coordination of the robots. Our new approach to pathfinding on geometrically embedded graphs, which we call dRRT, is an adaptation of the celebrated RRT algorithm (LaValle and Kuffner, 1999) for the discrete case of a graph, embedded in Euclidean space. dRRT traverses the composite roadmap, which may have exponentially many neighbors (exponential in the number of robots that need to be coordinated). The efficient traversal is achieved by retrieving only partial information on the explored roadmap. Specifically, it considers a single neighbor of a visited vertex at each step. dRRT rapidly explores the C-space represented by the implicit graph. Integrating the implicit representation of the roadmap allows us to solve multi-robot problems while exploring only a small portion of the C-space.

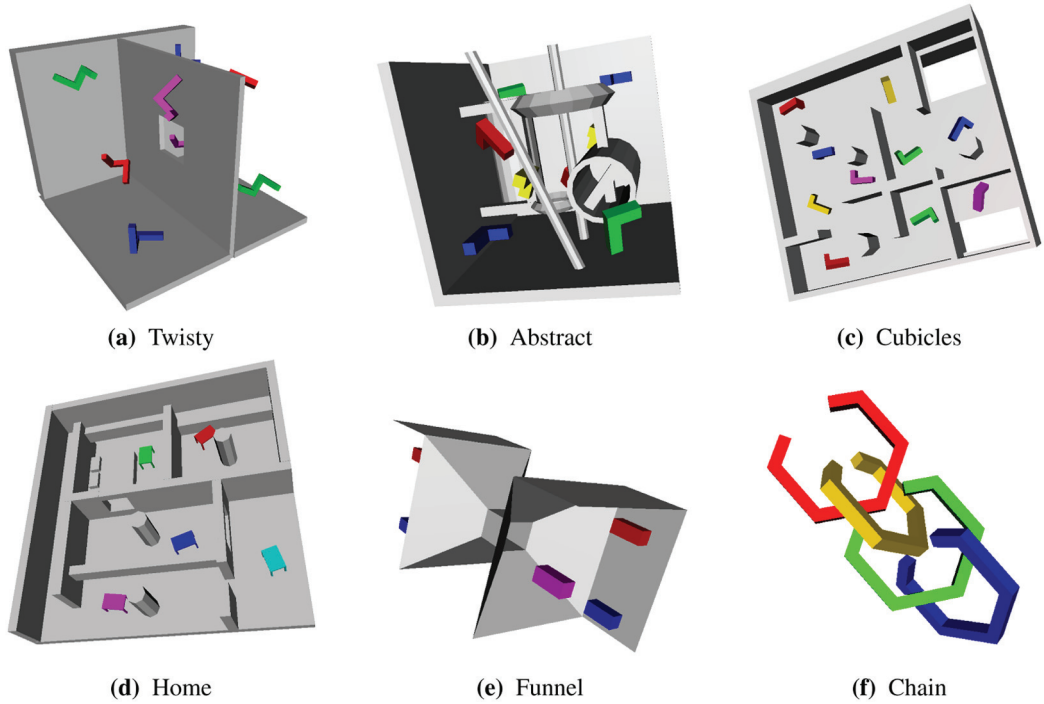
We demonstrate the capabilities of MRdRRT on the setting of polyhedral robots translating and rotating in space amidst polyhedral obstacles. We provide experimental results on several challenging scenarios, for most of which MRdRRT is faster by a factor of at least 10 when compared to existing algorithms that we are aware of. We show that we manage to solve problems of up to 60 DOFs, and scenarios that require tight coordination between the robots (see Figure 1).

The organization of this paper is as follows. In Section 2 we elaborate on two sampling-based multi-robot motion-planning algorithms, namely the composite roadmap approach by Švestka and Overmars (1998) and the work on subdimensional expansion and M* by Wagner and Choset (2015). In Section 3 we introduce the dRRT algorithm. For clarity of exposition, we first describe it as a general pathfinding algorithm for geometrically embedded graphs. In Section 4 we describe the MRdRRT method where dRRT is used in the setting of multi-robot motion planning for the exploration of the implicitly represented composite roadmaps. We show in Section 5 experimental results for the algorithm on different scenarios. We provide a discussion on the advantages and shortcomings of our algorithm in Section 6 and conclude with possible future research directions. On the theoretical side, we provide rigorous proofs for the *probabilistic completeness* of dRRT and MRdRRT in Sections 3 and 4, respectively.

Remark. We mention that we are not the first to consider RRTs in discrete domains. Branicky et al. (2003) applied the RRT algorithm to a discrete graph. However, a key difference between the approaches is that we assume that the graph is *geometrically embedded*, hence we use *random points* as samples while they use nodes of the graph as samples. Additionally, their technique requires that all the neighbors of a visited vertex will be considered—a costly operation in our setting, as mentioned above.

2. Composite roadmaps for multi-robot motion planning

We describe the composite roadmap approach introduced by Švestka and Overmars (1998). Here, a Cartesian product of PRM roadmaps of individual robots is considered as a means of devising a roadmap for the entire fleet of



(a) Twisty **(b)** Abstract **(c)** Cubicles

(d) Home **(e)** Funnel **(f)** Chain

Fig. 1. 3D environments with robots that are allowed to rotate and translate (6 DOFs each). In scenarios (a), (b), (c) and (e) robots of the same color need to exchange positions. (a) Twisty scenario with eight corkscrew-shaped robots, in a room with a barrier. (b) Abstract scenario with eight L-shaped robots. (c) Cubicles scenario with 10 L-shaped robots. (d) Home scenario with five table-shaped robots that are placed in different rooms. The goal is to change rooms in a clockwise order. (e) The Funnel scenario consists of six robots where three robots are located on each side of the funnel. The goal is to move every robot from one side of the funnel to the other. (f) The Chain scenario consists of four C-shaped robots located at the four corners of a room with no obstacles. They need to move to the center to create a highly coupled chain-like formation. Scenarios (a)–(d) are based on meshes provided by the Open Motion Planning Library (OMPL 0.10.2, Sucan et al., 2012) distribution.

robots. However, since they consider an explicit construction of this roadmap, their technique is applicable to scenarios that involve only a small number of robots. To overcome this, Wagner and Choset (2015) suggest representing the roadmap *implicitly* and describe a novel algorithm to find paths on this implicit graph.

Let $r_1, \ldots, r_m$ be m robots operating in a workspace W with robot r_i having start and target configurations s_i, t_i. We wish to find paths for every robot from start to target, while avoiding collision with obstacles as well as with the other robots. Let $G_i = (V_i, E_i)$ be a PRM roadmap for r_i, $|V_i| = n$, and let k denote the maximal degree of a vertex in any G_i. In addition, assume that $s_i, t_i \in V_i$, and that s_i, t_i reside in the same connected component of G_i. Given such a collection of roadmaps $G_1, \ldots, G_m$ a composite roadmap can be defined in two different ways—one is the result of a *Cartesian product* of the individual roadmaps while in the other a *tensor product* is used.

The *composite roadmap* $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ is defined as follows. The vertices $\mathbb{V}$ represent all combinations of collision-free placements of the m robots. Formally, a set of m robot configurations $C = (v_1, \ldots, v_m)$ is a vertex of $\mathbb{G}$ if for every i, $v_i \in V_i$, and in addition, when every robot r_i is placed in v_i the robots are pairwise collision-free. The Cartesian and tensor products differ in the type of edges in the resulting roadmap. If the Cartesian product is used, then $(C, C') \in \mathbb{E}$, where $C = (v_1, \ldots, v_m)$, $C' \in (v_1', \ldots, v_m')$, if there exists i such that $(v_i, v_i') \in E_i$, for every $j \neq i$ it holds that $v_j = v_j'$, and r_i does not collide with the other robots stationed at $v_j = v_j'$ while moving from v_i to v_i'. A tensor product generates many more edges. Specifically, $(C, C') \in \mathbb{E}$ if $(v_i, v_i') \in E_i$ for every i, and the robots remain collision-free while moving on the respective single-graph edges.[1]

Remark. Throughout this work we only consider the *tensor* product composite roadmap.

Note that by the definition of G_i and $\mathbb{G}$ it holds that $S, T \in \mathbb{V}$, where $S = (s_1, \ldots, s_m)$, $T = (t_1, \ldots, t_m)$. The following observation immediately follows (for both product types).

Observation 1. *Let $C_1, \ldots, C_h$ be a sequence of h vertices of $\mathbb{G}$ such that $S = C_1$, $T = C_h$. If for every two consecutive vertices $(C_i, C_{i+1}) \in \mathbb{E}$, then, there exists a path for the robots from S to T.*

Thus, given a composite roadmap $\mathbb{G}$, it is left to find such a path between S and T. Unfortunately, standard pathfinding techniques, which require the full representation of the graph, are prohibitively expensive to use since the number

of vertices of $\mathbb{G}$ alone may reach $O(n^m)$. One may consider the A* algorithm (Pearl, 1984), or its variants, as appropriate for the task, since it may not need to traverse all the vertices of graph. A central property of A* is that it needs to consider all the neighbors of a visited vertex in order to guarantee that it will find a path eventually. Alas, in our setting, this turns out to be a significant hurdle, since the number of neighbors of every vertex is $O(k^m)$.

Wagner and Choset (2015) introduced an adaptation of A* to the case of composite roadmaps called M*. Their approach exploits the observation that only the motion of some robots has to be coordinated in typical scenarios. Thus, planning in the joint C-space is only required for robots whose motion has to be coordinated, while the motion of the remaining robots can be planned individually. Hence, their method dynamically explores low-dimensional search spaces embedded in the full C-space, instead of the joint high-dimensional C-space. This technique is highly effective for scenarios with a low degree of coordination, and can cope with large fleets of robots in such settings. However, when the degree of coordination was increased, we observed a sharp increase in the running time of this algorithm, as it has to consider many neighbors of a visited vertex. This makes M* less effective when the motion of many robots needs to be tightly coordinated.

3. Discrete RRT

We describe a technique which we call *discrete RRT (dRRT)* for pathfinding in implicit graphs embedded in Euclidean space. For clarity of exposition, we first describe dRRT without the technicalities related to motion planning. We add these details in the subsequent section. As the name suggests, dRRT is an adaptation of the RRT algorithm (LaValle and Kuffner, 1999) for the purpose of exploring discrete geometrically embedded graphs, instead of a continuous space.

Since the graph serves as an approximation of some relevant portion of the Euclidean space, traversal of the graph can be viewed as a process of exploring the subspace. The dRRT algorithm rapidly explores the graph by biasing the search towards vertices embedded in unexplored regions of the space.

Let $G = (V, E)$ be a graph where every $v \in V$ is embedded in a point in the Euclidean space $\mathbb{R}^d$ and every edge $(v, v') \in E$ is a line segment connecting the points. Given two vertices $s, t \in V$, dRRT searches for a path in G from s to t. For simplicity, assume that the graph is embedded in $[0, 1]^d$.

Similarly to its continuous counterpart, dRRT grows a tree rooted in s and attempts to connect it to t to form a path from s to t. As in RRT, the growth of the tree is achieved by extending it towards random samples in $[0, 1]^d$. In our case though, vertices and edges that are added to the tree are taken from G, and we do not generate new vertices and edges along the way.

As G is represented implicitly, the algorithm uses an oracle to retrieve information regarding neighbors of visited vertices. We first describe this oracle and then proceed with a full description of the dRRT algorithm. Finally, we show that this technique is *probabilistically complete*, namely guaranteed to find a solution given a sufficient number of samples.

3.1. Oracle to query the implicit graph

In order to retrieve partial information regarding the neighbors of visited vertices, dRRT consults an oracle described below. We start with several basic definitions.

Given two points $v, v' \in [0, 1]^d$, denote by $\rho(v, v')$ the ray that starts in v and goes through v'. Given three points $v, v', v'' \in [0, 1]^d$, denote by $\angle_v(v', v'')$ the (smaller) angle between $\rho(v, v')$ and $\rho(v, v'')$.

Definition 1. *Given a vertex $v \in V$, and a point $u \in [0, 1]^d$ we define*

$$\mathcal{O}_{\mathrm{D}}(v, u) := \operatorname*{argmin}_{v'} \left\{ \angle_v(u, v') \,|\, (v, v') \in E \right\}$$

In other words, the direction oracle returns the neighbor v' of v in G such that the direction from v to v' is the closest to the direction from v to u, than any other neighbor v'' of v.

3.2. Description of dRRT

At a high level, dRRT (Algorithm 1) proceeds similarly to RRT: it grows a tree $\mathcal{T}$ which is a subgraph of G and is rooted in s (line 1). The growth of $\mathcal{T}$ (line 3) is achieved by an expansion towards random samples. Additionally, an attempt to connect $\mathcal{T}$ with t is made (line 4). The algorithm terminates when this operation succeeds and a solution path is generated (line 6), otherwise the algorithm goes to the next expansion iteration (line 2).

Expansion of $\mathcal{T}$ is performed by the EXPAND operation (Algorithm 2) which performs N iterations that consist of the following steps: A point q_{rand} is sampled uniformly from $[0, 1]^d$ (line 2). Then, a node q_{near} of $\mathcal{T}$ that is the closest to the sample (in Euclidean distance), is selected (line 3). q_{near} is extended towards the sample by locating the vertex $q_{\mathrm{new}} \in V$, that is the neighbor of q_{near} in G in the direction of q_{rand} (by the direction oracle $\mathcal{O}_{\mathrm{D}}$). Once q_{new} is found (line 4), it is added to the tree (line 6) with the edge $(q_{\mathrm{near}}, q_{\mathrm{new}})$ (line 7). An illustration of this process can be seen in Figure 2. This is already different from the standard RRT as we cannot necessarily proceed exactly in the direction of the random point.

After the expansion, dRRT attempts to connect the tree $\mathcal{T}$ with t using the CONNECT_TO_TARGET operation (Algorithm 3). For every vertex q of $\mathcal{T}$, which is one of the K nearest neighbors of t in $\mathcal{T}$ (line 1), an attempt is made to connect q to t using the method LOCAL_CONNECTOR (line 2) which is a crucial part of the dRRT algorithm (see Section 3.3). Finally, given a path from some node q of $\mathcal{T}$

Algorithm 1 dRRT_PLANNER (s, t)

1: $\mathcal{T}$.init(s)
2: **loop**
3: EXPAND($\mathcal{T}$)
4: $\Pi \leftarrow$ CONNECT_TO_TARGET($\mathcal{T}, t$)
5: **if** not_empty(Π) **then**
6: **return** RETRIEVE_PATH($\mathcal{T}, \Pi$)

Algorithm 2 EXPAND ($\mathcal{T}$)

1: **for** $i = 1 \rightarrow N$ **do**
2: $q_{\text{rand}} \leftarrow$ RANDOM_SAMPLE()
3: $q_{\text{near}} \leftarrow$ NEAREST_NEIGHBOR($\mathcal{T}, q_{\text{rand}}$)
4: $q_{\text{new}} \leftarrow \mathcal{O}_{\text{D}}(q_{\text{near}}, q_{\text{rand}})$
5: **if** $q_{\text{new}} \notin \mathcal{T}$ **then**
6: $\mathcal{T}$.add_vertex(q_{new})
7: $\mathcal{T}$.add_edge($q_{\text{near}}, q_{\text{new}}$)

Algorithm 3 CONNECT_TO_TARGET($\mathcal{T}, t,$)

1: **for** $q \in$ NEAREST_NEIGHBORS($\mathcal{T}, t, K$) **do**
2: $\Pi \leftarrow$ LOCAL_CONNECTOR(q, t)
3: **if** not_empty(Π) **then**
4: **return** Π
5: **return** $\emptyset$

to t the method RETRIEVE_PATH (Algorithm 1, line 6) returns the concatenation of the path from s to q, with Π.

We note that the only two parameters needed by our algorithm are the number of iterations N for which to expand the roadmap, and the number K of nearest neighbors used to connect the tree to the target. As we discuss in Section 5, one can set $K = \log i$ at the ith iteration.

3.3. Local connector

We show in the following subsection that it is possible that $\mathcal{T}$ will eventually reach t during the EXPAND stage, and therefore an application of LOCAL_CONNECTOR will not be necessary. However, in practice this is unlikely to occur within a short time frame, especially when G is large. Thus, we employ a heavy duty technique, which, given two vertices q_0, q_1 of G tries to find a path between them. We mention that it is common to assume in sampling-based algorithms that connecting nearby samples will require less effort than solving the original problem and here we make a similar assumption. We assume that a local connector is effective only on *restricted* pathfinding problems, thus in the general case it cannot be applied directly on s, t, as it may be too costly (unless the problem is easy). A concrete example of a local connector is provided in the next section.

3.4. Probabilistic completeness of dRRT

Recall that an algorithm is *probabilistically complete* if the probability that it finds a solution tends to one as the number of iterations tends to infinity, when such a solution exists. For simplicity, the proof does not make use of the local connector, i.e. the entire run of the dRRT algorithm consists of a single call to the EXPAND procedure (Algorithm 2), which terminates when the number of iterations reaches N, or when $q_{\text{new}} = t$ is added to $\mathcal{T}$. With a slight abuse of notation we use the vertices and their embedding interchangeably.

Let $G = (V, E)$ be a connected graph, embedded in $\mathbb{R}^d$, with a fixed number of vertices. Let $V' \subset V$ be a connected subset in G. For a given $v \in V'$, denote by $\text{Vor}(v, V')$ the *Voronoi cell* (de Berg et al., 2008) of the site v, in the Euclidean (standard) Voronoi diagram of point sites in $\mathbb{R}^d$, where the sites are point-embeddings of the vertices V' (Figure 2(b)). Now, denote by $\text{nbr}(v)$ the set of neighbors of v in G. For every $v^* \in \text{nbr}(v)$ denote by $\text{Vor}'(v, v^*)$ the Voronoi cell of $\rho(v, v^*)$, in the Voronoi diagram of the ray sites $\{\rho(v, v') \,|\, v' \in \text{nbr}(v)\}$ (see Figure 2(c)).

Lemma 2. *For a given G there exists $\xi > 0$ such that for every $V' \subseteq V, v \in V', v^* \in \text{nbr}(v)$, it holds that $|\text{Vor}(v, V') \cap \text{Vor}'(v, v^*)| \geq \xi$, where $|\cdot|$ denotes the* Lebesgue measure.

Proof. For every $V' \subseteq V$ and $v \in V'$ it holds that $\text{Vor}(v, V) \subseteq \text{Vor}(v, V')$, as adding a site u to the Voronoi diagram either leaves a cell intact or removes a portion of the cell $\text{Vor}(v, V')$: the portion inside the halfspace determined by the bisector of v and u, on the side of u. Consequently

$$\text{Vor}(v, V) \cap \text{Vor}'(v, v^*) \subseteq \text{Vor}(v, V') \cap \text{Vor}'(v, v^*)$$

for every $v^* \in \text{nbr}(v)$. Now, we set $v \in V, v^* \in \text{nbr}(v)$ to be the vertices for which the area of $\text{Vor}(v, V) \cap \text{Vor}'(v, v^*)$ is minimized and set

$$\xi := |\text{Vor}(v, V) \cap \text{Vor}'(v, v^*)|$$

For the remainder of the proof we assume that every two distinct vertices of G are embedded into two distinct points in $\mathbb{R}^d$, and every pair of edges of G incident to the same (embedded) vertex does not overlap in their interior. If several vertices collapse into a single Voronoi site, one can choose one of the vertices with equal probability, by slightly modifying the NEAREST_NEIGHBOR routine, in Algorithm 2. Overlapping edges can be treated in a similar manner.

It remains to show that $\xi > 0$. Let $v \in V, v^* \in \text{nbr}(v)$ be the vertices from which the value ξ is attained. As vertices are embedded into distinct points, and as edges do not overlap, we can deduce that $|\text{Vor}(v, V)| > 0$ and $|\text{Vor}'(v, v^*)| > 0$. In addition, the intersection between the two cells is clearly non-empty: There is a ball with radius $r > 0$ whose center is v and is completely contained in

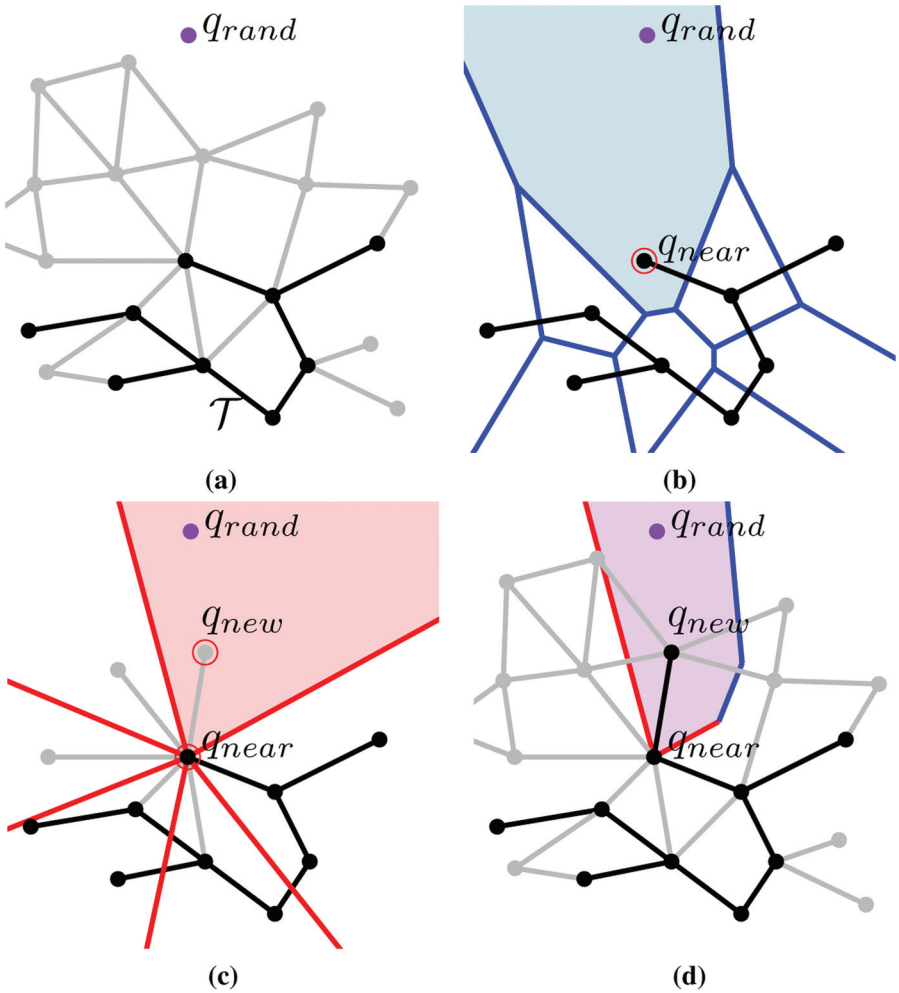


Fig. 2. An illustration of the expansion step of dRRT. The tree $\mathcal{T}$ is drawn with black vertices and edges, while the gray elements represent the unexplored portion of the implicit graph G. (a) A random point q_{rand} (purple) is drawn uniformly from $[0,1]^d$. (b) The vertex q_{near} of $\mathcal{T}$ that is the Euclidean nearest neighbor of q_{rand} is extracted. (c) The neighbor q_{new} of q_{near}, such that its direction from q_{near} is the closest to the direction of q_{rand} from q_{near}, is identified. (d) The new vertex and edge are added to $\mathcal{T}$. *Additional information for Theorem 2*: In (b) the Voronoi diagram of the vertices of $\mathcal{T}$ is depicted in blue, and the Voronoi cell of q_{near} ,$\text{Vor}(q_{\text{near}}, V')$, is shaded light blue, where V' denotes the vertices of $\mathcal{T}$. In (c) the Voronoi diagram of the rays that leave q_{near} and pass through its neighbors is depicted in red, and the Voronoi cell of $\rho(q_{\text{near}}, q_{\text{new}})$, $\text{Vor}'(q_{\text{near}}, q_{\text{new}})$, is shaded pink. The purple region in (d) represents $\text{Vor}(q_{\text{near}}, V') \cap \text{Vor}'(q_{\text{near}}, q_{\text{new}})$.

$\text{Vor}(v, V)$; similarly, there is a cone of solid angle $\alpha > 0$ with apex at v fully contained in $\text{Vor}'(v, v^*)$. Hence, it holds that $\xi > 0$, otherwise v and v^* are embedded in the same point. □

Theorem 3. *Let G be a connected graph embedded in $\mathbb{R}^d$, and denote by $s, t \in V$ the start and target vertices, respectively. The probability that dRRT finds a path from s to t converges to 1 as $N \to \infty$, where N represents the number of iterations (steps) performed by dRRT in the EXPAND procedure.*

Proof. Denote by $L := (s = v_1, v_2, \ldots, v_\ell = t)$ the shortest path between $s, t \in V$ in G, in terms of the number of vertices, and let $\mathcal{V}_j := \text{Vor}(v_j, V) \cap \text{Vor}'(v_j, v_{j+1})$ for $1 \le j \le \ell - 1$. Additionally, denote by $q_1, \ldots, q_N$ the samples drawn consecutively by dRRT, and by $V_i^{\mathcal{T}}$ the vertices of $\mathcal{T}$ at step $1 \le i \le N$.

The main observation here is that for a given step i of dRRT, if $q_i \in \mathcal{V}_j$ and $v_j \in V_i^{\mathcal{T}}$, for $1 \le j \le \ell - 1$ then the edge (v_j, v_{j+1}) will be added to $\mathcal{T}$ at the end of this step. Thus, with probability at least $\xi > 0$ dRRT will make an advancement along L toward t (Lemma 2).

The process of uncovering L by dRRT can be modeled as a *Markov chain*, where for every $v_j \in L$ we have a state S_j, which represents the event that the farthest vertex along L that was reached is v_j. Given that we arrived at state S_j, the probability that we will move to state S_{j+1} given the next sample is denoted by $\beta_j > \xi$. We stay put at S_j with probability at most $1 - \beta_j$. Note that once $t = v_\ell$ is reached, dRRT will not proceed to explore G for the remainder of

the iterations. Thus, the probability of leaving the state S_ℓ is equal to 0. The process can be viewed as an *absorbing* Markov chain with the absorbing state S_ℓ, in which every state can reach S_ℓ, but once S_ℓ is entered, it cannot be left. It is known that the probability of reaching an absorbing state within N steps tends to 1 as N tends to ∞ (see Theorem 11.3 in Chapter 11.2 of Grinstead and Snell, 2012). □

Note that this is true only when the values of the transition probabilities do not depend on N, which is indeed the case in our setting. Specifically, as the number of vertices of G, and the dimension in which it is embedded, are assumed to be fixed, ξ is fixed as well. The same applies for the length of L, namely the value ℓ.

4. Multi-robot motion planning with dRRT

In this section we describe the MRdRRT algorithm. Specifically, we discuss the adaptation of dRRT for pathfinding in a composite roadmap $\mathbb{G}$, which is embedded in the joint C-space of m robots. In particular, we show an implementation of the oracle $\mathcal{O}_\mathrm{D}$, which relies solely on the representation of $G_1, \ldots, G_m$. Additionally, we discuss an implementation of the local-connector component, which takes advantage of the fact that $\mathbb{G}$ represents a set of valid positions and movements of multiple robots. Finally, we discuss the probabilistic completeness of our entire approach to multi-robot motion planning.

4.1. Direction oracle $\mathcal{O}_\mathrm{D}$

Recall that given $C \in \mathbb{V}$ and a random sample q, $\mathcal{O}_\mathrm{D}(C, q)$ returns $C' \in \mathbb{V}$ that is a neighbor of C in $\mathbb{G}$, and for every other neighbor C'' of C, $\rho(C, q)$ forms a smaller angle with $\rho(C, C')$ than with $\rho(C, C'')$, where ρ is as defined in Section 3.4.

Denote by $\mathcal{C}(r_i)$ the C-space of r_i. Let $q = (q_1, \ldots, q_m)$ where $q_i \in \mathcal{C}(r_i)$, and let $C = (c_1, \ldots, c_m)$ where $c_i \in V_i$. To find a suitable neighbor for C we first find the most suitable neighbor for every individual robot and combine the m single-robot neighbors into a candidate neighbor for C. We denote by $c_i' = \mathcal{O}_\mathrm{D}(c_i, q_i)$ the neighbor of c_i in G_i that is in the direction of q_i. Notice that the implementation of the oracle for individual roadmaps is trivial—for example, by traversing all the neighbors of c_i in G_i. Let $C' = (c_1', \ldots, c_m')$ be a candidate for the result of $\mathcal{O}_\mathrm{D}(C, q)$. If (C, C') represents a valid edge in $\mathbb{G}$, i.e. no robot–robot collision occurs, we return C'. Otherwise, $\mathcal{O}_\mathrm{D}(C, q)$ returns $\emptyset$. In this case, the new sample is ignored and another sample is drawn in the EXPAND phase (Algorithm 2).

The completeness proof of the dRRT (Theorem 3) for this specific implementation of $\mathcal{O}_\mathrm{D}$, is straightforward. Notice that in order to extend $C = (c_1, \ldots, c_m)$ to $C' = (c_1', \ldots, c_m')$ the sample $q = (q_1, \ldots, q_m)$ must obey the following restriction: For every robot r_i, q_i must lie in $\mathrm{Vor}(c_i, V_i) \cap \mathrm{Vor}'(c_i, c_i')$ (where in the original proof we required that q will lie in $\mathrm{Vor}(C, \mathbb{V}) \cap \mathrm{Vor}'(C, C')$).

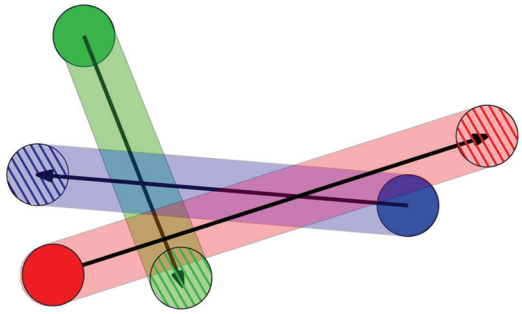


Fig. 3. Visualization of the local connector used to connect two vertices of the graph based on the method described by van den Berg et al. (2009). Three disk robots (red, green and blue) need to move from a start placement to a target placement (depicted by shaded disks and dashed disks, respectively) by following a straight line. At every given point of time, one robot moves while the other robots are stationary either in their start or target configurations. Note that the path of the red robot is blocked both by the start position of the blue robot and the target position of the green robot. Thus, to allow for a collision-free path based on prioritizing the movements, the red robot should move *before* the green robot moves, and *after* the blue one moves.

4.2. Local-connector implementation

Recall that in the general dRRT algorithm the local connector is used for connecting two given vertices of a graph. For our local connector we rely on a framework described by van den Berg et al. (2009). Given two vertices $\mathbb{V} = (v_1, \ldots, v_m), \mathbb{V}' = (v_1', \ldots, v_m')$ of $\mathbb{G}$ we find for each robot i a path π_i on G_i from v_i to v_i'. The connector attempts to find an ordering of the robots such that robot i does not leave its start position on π_i until robots with higher priority reached their target positions on their respective path, and of course that it also avoids collisions. When these robots reach their destination, robot i moves along π_i from $\pi_i(0)$ to $\pi_i(1)$. During the movement of this robot the other robots stay put.

The priorities are assigned according to the following rule: if moving robot i along π_i causes a collision with robot j that is placed in v_j then robot i should move *after* robot j. Similarly, if i collides with robot j that is placed in v_j' then robot i should move *before* robot j. This prioritization induces a directed graph $\mathcal{I}$. In case this graph is acyclic we generate a solution according to the prioritization of the robots. Otherwise, we report failure. For a visualization of the method, see Figure 3. We decided to use this simple technique in our experiments due to its low cost, in terms of running time.

We remark that we also experimented with M* as a local connector. The motivation here is the common assumption in sampling-based motion planning that connecting close-by configurations is often an easy problem. As the local connector is applied multiple times, it should be simple and extremely efficient on easy instances, even at the cost of not finding a complex solution when one exists. Indeed,

for easy problems with a *low degree of coordination* M* meets the above criteria. Now, to use M* as a local connector, one needs to modify it such that when a local path is not easy to find, the running time and the memory consumption of M* will not be high. Obviously, this comes at the price of relaxing the completeness of M*. Thus, we bounded both the degree of coordination that M* could consider (to avoid considering exponentially many neighbors for a certain node) and the number of nodes that M* could consider (to bound the running time of M*). While very efficient at times, this approach introduced the need to tune the above parameters and for certain scenarios did not yield satisfactory results. The ordering algorithm of van den Berg et al. (2009) turned out to be considerably more efficient and obviously it is parameter-free.

4.3. Probabilistic completeness of MRdRRT

In order for the motion-planning framework to be probabilistically complete, we need to show that (i) as the number of samples used for each single-robot roadmap tends to ∞, the probability that the composite roadmap contains a path tends to 1, and (ii) that the proof of Theorem 3 still holds when the size of the graph tends to ∞.

A motion-planning problem is said to be *robustly feasible* (Karaman and Frazzoli, 2011), if there exists a solution with positive clearance. In the setting of multi-robot motion planning, robust feasibility implies that there exists a solution in which the robots do not move in contact with obstacles or with the other robots. Švestka and Overmars show that the composite-roadmap approach is probabilistically complete, given that the problem is robustly feasible, and assuming that the underlying graph-search algorithm is complete.

Theorem 4. *(Švestka and Overmars, 1998) Given a robustly feasible problem, the probability that the composite roadmap, constructed from PRM roadmaps with n vertices each, denoted by* $\mathbb{G}(n)$*, contains a solution that tends to* 1 *as n tends to* ∞*, i.e.*

$$\lim_{n\to\infty} \Pr[\mathbb{G}(n) \text{ contains a solution}] = 1$$

In our setting we employ a probabilistic algorithm to query $\mathbb{G}$, namely dRRT, whose success rate tends to 1 as the number of its samples N tends to ∞, under the assumption that n is fixed (see Theorem 3). We first prove a weaker property of MRdRRT, and then proceed to the main theorem.

Proposition 5. *Given a robustly feasible problem, for every* $0 < p < 1$ *there exists* $n_0 \in \mathbb{N}^+$ *such that for every* $n > n_0$ *it holds that*

$$\lim_{N\to\infty} \Pr[\text{MRdRRT finds a solution using } \mathbb{G}(n) \text{ and } N \text{ samples}] \geq p$$

Proof. Let $0 < p < 1$ be some constant. By Theorem 4, $\mathbb{G}(n)$ contains a solution with probability 1 as n tends to ∞. Then, there also exists $n_0 \in \mathbb{N}^+$ such that for every $n > n_0$ it follows that

$$\Pr[\mathbb{G}(n) \text{ contains a solution}] \geq p$$

Denote by $\mathcal{P}(n, N)$ the event that MRdRRT finds a solution using $\mathbb{G}(n)$ and N samples. We fix some $n_1 > n_0$. From Theorem 3 it follows that

$$\begin{aligned}\lim_{N\to\infty} \Pr[\mathcal{P}(n_1, N)] &= \lim_{N\to\infty} \Pr[\mathcal{P}(n_1, N) \,|\, \mathbb{G}(n_1) \\ &\quad \text{contains a solution}] \cdot \Pr[\mathbb{G}(n_1) \text{ contains a solution}] \\ &= \lim_{N\to\infty} \Pr[\mathcal{P}(n_1, N) \,|\, \mathbb{G}(n_1) \text{ contains a solution}] \cdot p = p\end{aligned}$$

□

Theorem 6. *Given a robustly feasible problem, the probability of MRdRRT to find a solution tends to* 1*, as n and N tend to* ∞*.*

Proof. For every positive integer i, define $\varepsilon_i := 1/i$. Let n_i be the minimal value such that for every $n \geq n_i$ there exists N such that $\Pr[\mathcal{P}(n, N)] \geq 1 - \varepsilon_i$. Note that such n_i exists by Proposition 5. For a given n, let $i(n)$ be such that $n_i \leq n < n_{i+1}$. Also, let $N(n)$ be the minimal value for which $\Pr[\mathcal{P}(n, N(n))] \geq 1 - \varepsilon_{i(n)}$. Then, it follows that

$$\lim_{n\to\infty} \Pr[\mathcal{P}(n, N(n)] \geq \lim_{n\to\infty} 1 - \varepsilon_{i(n)} = 1$$

□

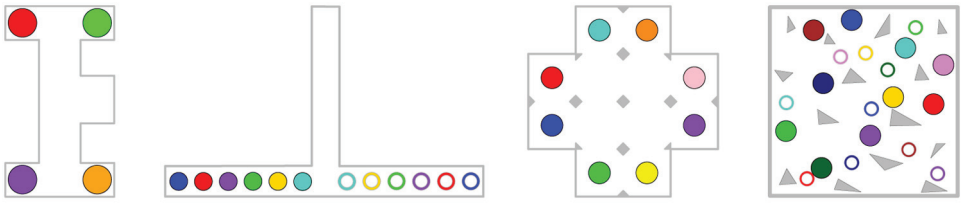
5. Experimental results

We implemented MRdRRT for the case of polygonal and polyhedral robots translating and rotating among polygonal and polyhedral obstacles, respectively. We compared the performance of MRdRRT with RRT and an improved (recursive) version of M* that appears in Wagner and Choset (2015). To make the comparison as equitable as possible, we used the *inflated* version of M* which has relaxed optimality guarantees (as dRRT does not take into consideration the quality of the solution).

5.1. Implementation details

The algorithms were implemented in C++. The experiments were conducted on a laptop with an Intel i5-3230M 2.60 GHz processor with 16 GB of memory, running 64-bit Windows 7. We implemented a generic framework for multi-robot motion planning based on composite roadmaps.

The implementation relies on the CGAL Project (2011) and on PQP (see `http://gamma.cs.unc.edu/SSV/` for collision detection in the two- and three-dimensional workspaces, respectively, and performs nearest-neighbor queries using the Fast Library for Approximate Nearest Neighbors (FLANN) by Muja and Lowe (2009). Metrics,



(a) Bay **(b)** T-shape **(c)** Room **(d)** Random

Fig. 4. Highly coupled settings in planar environments used for the experiments. (a) The red and green robots (at the top) need to exchange positions with the purple and orange robots (at the bottom), (b) the robots need to reverse their order by entering the narrow corridor (start and target positions are depicted by shaded disks and annuli, respectively), (c) opposite robots need to interchange positions, (d) 10 randomly placed robots that move to random goals (start and target locations are depicted by shaded disks and annuli, respectively).

sampling and interpolation in the 3D environments follow the guidelines of Kuffner (2004). To eliminate the dependence of dRRT on parameters we assigned them according to the number of iterations the algorithm performed so far, i.e. the number of times that the main loop has been repeated. Specifically, in the ith iteration each EXPAND (Algorithm 2) call performs 2^i iterations ($N = 2^i$), while CONNECT_TO_TARGET uses $K = i$ candidates that are connected with t. Let k_{nn} denote the number of nearest neighbors that is used to construct the roadmaps of the individual robots. We used the values $k_{\text{nn}} = 15$ and $k_{\text{nn}} = 12$ for the two- and three-dimensional workspaces, respectively.

5.2. Test scenarios

We report results for two sets of tests. In the first set we test M* and MRdRRT on challenging scenarios (Figures 1 and 4) to illustrate the performance of the two techniques. In the second set of tests we study how the difficulty of the scenarios, in terms of the necessary amount of coordination between the robots, affects the running time of M* and MRdRRT.

Basic scenarios. The two-dimensional workspaces (see Figure 4) represent scenarios where the motion of the robot needs to be tightly coordinated. In all the scenarios a collection of between 4 and 10 identical disk robots move in a polygonal workspace with relatively little clearance. The 3D workspaces used (see Figure 1) are constructed using meshes provided by the OMPL. The Twisty scenario (Figure 1(a)) contains eight corkscrew-shaped robots, in a room with a barrier. There are four robots on each side of the barrier and each robot needs to pass through the small hole in the barrier to reach the other side. This results in a tight coordination of the robots' motion in the vicinity of the barrier. The Abstract scenario (Figure 1(b)) and the Cubicles scenario (Figure 1(c)) contain 8 and 10 L-shaped robots, respectively. While the shape of the obstacles differ, these two scenarios are similar as both require coordinated motion of the robots due to the large size of the robots and the obstacles with respect to the small size of the room. The Home scenario (Figure 1(d)) contains five table-shaped robots that are placed in different rooms. The goal is to move each table to a different room. This problem contains little coordination as only one robot needs to pass in every narrow passage connecting the rooms. Next, the Funnel scenario (Figure 1(e)) consists of six robots where three robots are located on each side of the funnel and the goal is to move every robot from one side of the funnel to the other. Finally, the Chain scenario (Figure 1(f)) consists of four C-shaped robots located at the four corners of a room with no obstacles. In this "bug trap"-like scenario the robots need to move to the center to create a highly coupled chain-like formation.

We report in Table 1 the running times of M* and dRRT for the scenarios. We ran each of the 3 algorithms 10 times on each scenario. We remark that RRT proved incapable of solving any of the test scenarios, running for several 10s of minutes until terminating due to exceeding the memory limits. We believe that RRT as-is is unsuitable for multi-robot motion planning in cases where the scenarios consist of more than a couple of robots. M* exhibited better performance than RRT. However, for almost all of the attempts in all 2D scenarios, it failed to find a solution (even in scenarios with a relatively small number of robots). Only one solution was found for the Random scenario, with running times exceeding those of MRdRRT by roughly 50%. For the Twisty, Funnel and Chain scenarios, which involve multiple robots and require a substantial amount of coordination, M* failed to find a solution. For the Abstract and the Cubicles scenarios, it never exceeded a success rate of 40%. In particular, it often ran out of memory or ran for a very long duration, and was terminated if its running time exceeded 10× the running time of MRdRRT. On the other hand, MRdRRT was stable in its results and managed to solve all the scenarios, except for the Chain scenario, for each of the 10 attempts. When M* did manage to solve 1 of the first 3 scenarios, it explored between 2.5 to 10× the number of vertices that dRRT explored. For the Home scenario the results of MRdRRT and M* were comparable and in general we found M* more suitable for situations where only a small number of robots have to be coordinated at any

given time. We mention that MRdRRT was unable to solve scenarios that consist of a substantially larger number of robots than we used in our experiments. We believe that it would be beneficial to consider a stronger *local connector* in such cases.

As mentioned, both algorithms failed to find a solution for the Chain scenario. We believe that this is due to the fact that this problem can be interpreted as a bug trap, where a search algorithm may benefit from performing *bidirectional search.* Indeed, we implemented such a variant (see Section 7), which solved the problem (results presented in Table 1). We mention that such an approach may be applicable to M* as well.

Gradual increase of coordination. We studied how the difficulty of scenarios affects the performance of M* and MRdRRT. For this purpose we fixed a workspace environment and the collection of start and target positions of the robots, and gradually increased the size of the robots (Figure 5). The increase gradually eliminated pathways through which robots can avoid one another. As a result, the motion of robots became increasingly coupled. The two algorithms exhibited similar performance for the two most simple cases, in which a scale of 1 and 1.3 was used for the robots. However, using a scaling factor of 1.7 and higher, MRdRRT outperformed M* in terms of the running time. Additionally, the success rate of the former stands on 100% throughout the entire test set, while the latter's success rate was at %50 for a scaling factor of 2.1, and decreased even more for the harder cases. In particular, for scaling 3.7, M* had a success rate of 0%.

6. Discussion

In this section we review the advantages and limitations of MRdRRT. Recall that the implicitly represented composite roadmap $\mathbb{G}$ results from a tensor product of m PRM roadmaps $G_1, \ldots, G_m$. The reliance on the precomputed individual roadmaps eliminates the need to perform additional collision checking between robots and obstacles while querying $\mathbb{G}$. This has a substantial impact on the performance of MRdRRT as it is often the case that checking whether m robots collide with obstacles is much more costly than checking whether the m robots collide between themselves. This is in contrast with more naïve approaches, such as RRT, which consider the group of robots as one large robot. In such cases, checking whether a configuration (or an edge) is collision-free requires checking for the two types of collisions simultaneously.

The M* algorithm, which also uses the underlying structure of $\mathbb{G}$, performs very well in situations where only a small subset of the robots need to coordinate. In these situations it can cope, almost effortlessly, with several 10s of robots while outperforming our framework. This points indeed to a weakness of dRRT in easy scenarios. However, in scenarios where a substantial amount of coordination is required between the robots (see, e.g. Figure 4(b)), M* suffers from a disadvantage, since it is forced to consider exponentially many neighbors when performing the search on $\mathbb{G}$. In contrast, dRRT performs a "minimalistic" search and advances in small steps, little by little, regardless of the difficulty of the problem at hand. Moreover, dRRT strives to reach unknown regions in $\mathbb{G}$ while avoiding spending too much time in the exploration of regions that are in the vicinity of explored vertices. This is done via the Voronoi bias, as shown in the proof of Theorem 3. This is extremely beneficial when working on $\mathbb{G}$ since it contains vertices which densely cover the space, and thus considering many vertices within a small region would not lead to a better understanding of the problem at hand. To justify this claim, consider the following example. Suppose that for every robot r_i, v_i is a vertex of V_i that has k neighbors in G_i at distance at most ε. Then the vertex $(v_1, ..., v_m) \in \mathbb{V}$ might have as many as k^m neighbors that are at distance at most $\varepsilon\sqrt{m}$ in $\mathbb{G}$.

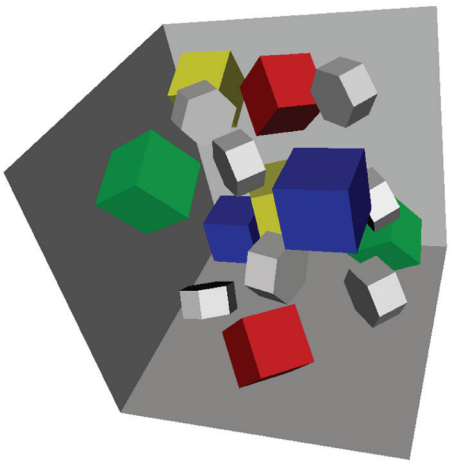
7. Future work

Towards optimality. Currently, our algorithmic framework is concerned with finding *some* solution. Our immediate future goal is to modify it to provide a solution with quality guarantees, possibly by taking an approach similar to the continuous RRT* algorithm (Karaman and Frazzoli, 2011), which is known to be asymptotically optimal. A fundamental difference between RRT* and the original formulation of RRT is in a rewiring step, where the structure of the tree is revised to improve previously examined paths. Specifically, when a new node is added to the tree, it is checked as to whether it will be more beneficial for some of the existing nodes to point to the new vertex instead of their current parent in the tree. This can be adapted, to some extent, to the discrete case, although it is not clear how to efficiently perform rewiring on an implicitly represented graph.

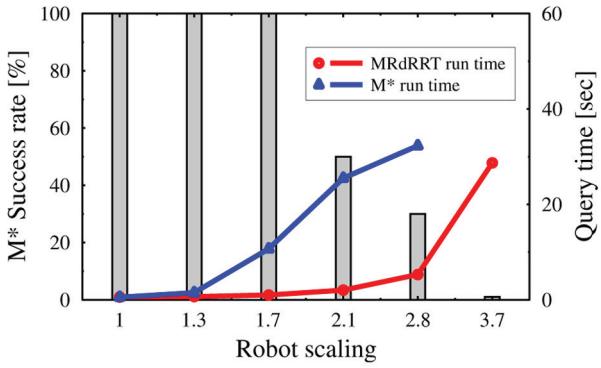
dRRT in other settings of motion planning. In this paper we combined the dRRT algorithm with implicit composite roadmaps to provide an efficient algorithm for multi-robot motion planning. One of the benefits of our framework comes from the fact that it reuses some of the already computed information to avoid performing costly operations. In particular, it refrains from checking collisions between robots and obstacles by forcing the individual robots to move on precalculated individual roadmaps (i.e. G_i). A similar approach can be used in other settings of motion planning. In particular, we have recently developed a dRRT-based approach for motion planning of free-flying multi-link robots (Salzman et al., 2015b). The new approach generates an implicitly represented roadmap, which encapsulates information on configurations and paths that do not induce self-intersections for the robot, while ignoring the existence of obstacles. Then, we overlay this roadmap on the workspace, an operation which invalidates some of the nodes and edges of the roadmap. Thus, we know only which configurations are self-collision-free, but not obstacle collision-free. Then we employ dRRT for pathfinding

Table 1. Results for M* and MRdRRT on the scenarios depicted in Figures 1 and 4. We first report the number of vertices used in the construction of the single-robot PRM roadmaps and the elapsed time (all times are reported in seconds). Then we report the number of visited vertices, the averaged total running time over the number of successful attempts (out of 10), the standard deviation, and the success rate of M*. A similar report is given for dRRT, but we also specify the duration of the connection phase (using the local connector) and the expansion phase. The running times and the number of explored vertices are averaged over the number of successful attempts. The results for MRdRRT in the Chain scenario are presented for a bidirectional variant of dRRT. See Section 7.

		2D Scenarios					3D Scenarios				
		Bay	T-shape	Room	Rand	Twisty	Abstract	Cubicles	Home	Funnel	Chain
PRM	Vertices number	300	300	300	300	8k	10k	10k	5k	500	15k
	Generation time	0.1 s	0.1 s	0.1 s	0.1 s	10 s	24.8 s	16.2 s	10.1 s	17 s	8.7 s
M*	Visited vertices	∞	∞	∞	450k	∞	300k	27k	2k	∞	∞
	Total time (avg.)	∞	∞	∞	106 s	∞	267 s	31 s	4 s	∞	∞
	Total time (s.d.)	∞	∞	∞	0	∞	205	13	1.8	∞	∞
	Success rate	0%	0%	0%	10%	0%	30%	40%	100%	0%	0%
MRdRRT	Visited vertices	380k	5.3k	320k	330k	8k	34k	12k	8k	120k	600
	Connect time	5.4 s	3.1 s	9.7 s	7.9 s	3.3 s	30.4 s	16.3 s	1.5 s	110 s	3.3 s
	Expand time	119 s	1.2 s	27 s	69.4 s	6.7 s	25.5 s	36.8 s	2.9 s	373 s	5.4 s
	Total time (avg.)	124.4 s	4.3 s	36.7 s	77.3 s	11 s	55.9 s	53.1 s	4.4 s	500 s	17.4 s
	Total time (s.d.)	57.3	1.8	3.3	69.8	2.0	38	78	1.6	221	9.3
	Success rate	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%



(a) Gradual coordination scenario



(b) Comparison of results

Fig. 5. Comparing the affect of a gradual increase in the degree of coordination on the performance of MRdRRT and M*. (a) Eight cubical robots, capable of translating and rotating, are placed in a scenario cluttered with polyhedral obstacles (drawn in gray). Robots of the same color need to exchange positions. The robots' size is gradually increased, which results in an increase in the difficulty of the problem. In this figure, the most difficult setting is depicted, where the robots are scaled by a factor of 3.7 from their original size. (b) Results for the different scaling factors for the scenario. Success rate for M* is depicted by shaded bars (MRdRRT had a 100% success rate in all cases). The average query time for MRdRRT and M* is depicted by colored graphs. A run of M* was considered a failure if its running time exceeded 10× the average running time of MRdRRT, or it ran out of memory.

on the new roadmap, while avoiding self-collision tests and while exploring a small portion of a massive roadmap. A key benefit of our approach is the ability to cache local-planner motions, which are particularly costly to compute, when the robot at hand has many DOFs. A visualization of the approach can be seen in Figure 6.

dRRT variants. Our dRRT algorithm conceptually mimics the celebrated RRT algorithm for the case of a discrete graph embedded in some geometric space. However, the motion-planning literature contains many more algorithms, some more suitable for specific problem instances. For example, the RRT-Connect by Kuffner and LaValle (2000) which grows two RRT-trees—one rooted at the source and one at the target. As mentioned in Section 5, the Chain scenario is an example of such a problem. We implemented a simple variant of dRRT that performs a bidirectional search similar to RRT-Connect. While M* or dRRT did not find a solution, when we used the bidirectional variant of

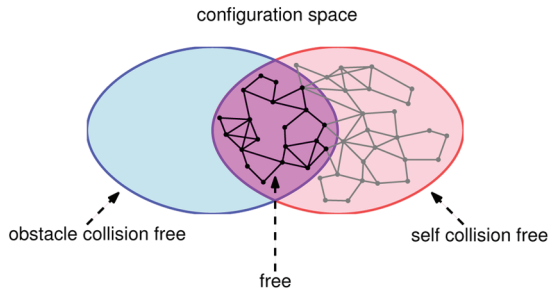


Fig. 6. Visualization of the dRRT-based approach for motion planning of a free-flying multi-link robot. The entire configuration space of the problem is illustrated as the bounding box. The red ellipse represents the set of configurations which do not induce self-collisions. Similarly, the blue ellipse represents the set of configurations which do not induce collisions between the robot and the workspace obstacles. The purple area represents the set of configurations which are fully free. In our dRRT-based approach for this setting we generate an implicit roadmap which provides a tilling of the self collision-free space, and using dRRT we uncover the portion of this roadmap which is also obstacle collision-free. Note that in this case the implicit roadmap is generated for a specific type of robot, and is independent of the structure of the workspace.

dRRT, a solution was quickly found (see results in Table 1). Other motion-planning algorithms can be adapted to a discrete variant and we give two examples to demonstrate how one may transform a continuous motion-planning algorithm to search in a geometrically embedded roadmap. Consider the Expansive Space-Tree planner (EST) by Hsu et al. (1999) and the Resolution Independent Density Estimation for Motion Planning in High-Dimensional Spaces planner (STRIDE) by Gipson et al. (2013). Both algorithms maintain a tree as a roadmap and expand the tree by selecting a node from the tree. While the details on how to select a node differs between algorithms, they both rely on a geometric embedding of the tree's vertices to perform the selection. Once a node x is chosen from the tree, a random configuration y in the vicinity of x is selected (again, the algorithms differ in the way the node is selected). The planners then attempt to connect x to y. To apply each of the above algorithms for the case of a discrete graph $\mathcal{G}$, one needs to simply replace y with the vertex $y' \in \mathcal{G}$ such that y' is the neighbor of x which is the closest to y.

Note

1. There is wide consensus on the term *tensor product* as defined here, and less so on the term *Cartesian product*. As the latter has already been used before in the context of motion planning, we will keep using it here as well.

Acknowledgements

We wish to thank Glenn Wagner for advising on the M* algorithm and providing helpful additional information. We also thank Ariel Felner for advice regarding pathfinding algorithms on graphs. We note that the title name "Finding a Needle in an Exponential Haystack" has been previously used in a talk by Joel Spencer in a different context.

Funding

This work was supported in part by the 7th Framework Programme for Research of the European Commission (FET-Open grant number 255827, CGL—Computational Geometry Learning), by the Israel Science Foundation (grant number 1102/11), by the German–Israeli Foundation (grant number 1150-82.6/2011), and by the Hermann Minkowski–Minerva Center for Geometry at Tel-Aviv University.

References

Adler A, de Berg M, Halperin D and Solovey K (2015) Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Transactions on Automation Science and Engineering* 12(4): 1309–1317. DOI: 10.1109/TASE.2015.2470096.

Aronov B, de Berg M, van der Stappen AF, Švestka P and Vleugels J (1999) Motion planning for multiple robots. *Discrete & Computational Geometry* 22(4): 505–525.

Auletta V, Monti A, Parente M and Persiano P (1996) A linear time algorithm for the feasibility of pebble motion on trees. In: Karlsson R and Lingas A (eds) *Algorithm Theory – SWAT'96*. New York: Springer, vol. 1097, pp. 259–270.

Branicky MS, Curtiss MM, Levine JA and Morgan SB (2003) RRTs for nonlinear, discrete, and hybrid planning and control. In: *42nd IEEE conference on decision and control*, Maui, HI, 9–12 December 2003, vol. 1, pp. 656–663. Piscataway: IEEE Press.

Choset H, Lynch K, Hutchinson S, Kantor G, Burgard G, Kavraki L et al. (2005) *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge: MIT Press.

de Berg M, van Kreveld M, Overmars M and Schwarzkopf O (2008) *Computational Geometry: Algorithms and Applications*. Berlin, Germany: Springer-Verlag.

Gipson B, Moll M and Kavraki L (2013) Resolution independent density estimation for motion planning in high-dimensional spaces. In: *2003 IEEE international conference on robotics and automation (ICRA'03)*, Karlsruhe, Germany, 6–10 May 2013, pp. 2437–2443. Piscataway: IEEE Press. Berlin, Germany.

Goraly G and Hassin R (2010) Multi-color pebble motion on graphs. *Algorithmica* 58(3): 610–636.

Grinstead C and Snell J (2012) *Introduction to Probability*. Providence, RI: American Mathematical Society. Providence, RI, USA.

Hearn R and Demaine E (2005) PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science* 343(1–2): 72–96.

Hirsch S and Halperin D (2002) Hybrid motion planning: coordinating two discs moving among polygonal obstacles in the plane. In: Boissonnat J-D, Burdick J, Goldberg K and Hutchinson S (eds) *Algorithmic Foundations of Robotics V*. New York: Springer, pp. 239–255.

Hopcroft J, Schwartz J and Sharir M (1984) On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "Warehouseman's problem". *International Journal of Robotics Research* 3(4): 76–88.

Hsu D, Latombe JC and Motwani R (1999) Path planning in expansive configuration spaces. *International Journal of Computational Geometry & Applications* 9(4–5): 495–512.

Karaman S and Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research* 30(7): 846–894.

Kavraki LE, Švestka P, Latombe JC and Overmars M (1996) Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4): 566–580.

Kloder S and Hutchinson S (2005) Path planning for permutation-invariant multi-robot formations. In: *2005 IEEE international conference on robotics and automation (ICRA'05)*, Barcelona, Spain, 18–22 April 2005, pp. 1797–1802. Piscataway: IEEE Press.

Kornhauser D (1984) *Coordinating pebble motion on graphs, the diameter of permutation groups, and applications*. MSc Thesis, Massachusetts Institute of Technology, USA.

Kuffner J (2004) Effective sampling and distance metrics for 3D rigid body path planning. In: *2004 IEEE international conference on robotics and automation (ICRA'04)*, New Orleans, LA, 26 April–1 May 2004, pp. 3993–3998. Piscataway: IEEE Press.

Kuffner J and LaValle S (2000) RRT-Connect: An efficient approach to single-query path planning. In: *2000 IEEE international conference on robotics and automation (ICRA'00)*, April 24–28, 2000, San Francisco, CA, USA. pp. 995–1001. Piscataway: IEEE Press.

LaValle SM (2006) *Planning Algorithms*. Cambridge: Cambridge University Press.

LaValle SM and Kuffner J (1999) Randomized kinodynamic planning. In: *1999 IEEE international conference on robotics and automation (ICRA'99)*, Detroit, MI, 10–15 May 1999, pp. 473–479. Piscataway: IEEE Press.

Leroy S, Laumond JP and Simeon T (1999) Multiple path coordination for mobile robots: A geometric algorithm. In: *16th international joint conference on artificial intelligence (IJCAI'99)*, Barcelona, Catalonia, Spain, 16–22 July 2011, pp. 1118–1123.

Luna R and Bekris KE (2011) Push and swap: Fast cooperative path-finding with completeness guarantees. In: *22nd international joint conference on artificial intelligence (IJCAI'11)*, Barcelona, Catalonia, Spain, July 16–22, 2011. pp. 294–300.

Muja M and Lowe DG (2009) Fast approximate nearest neighbors with automatic algorithm configuration. In: *4th international conference on computer vision theory and applications (VISAPP'09)*, Lisbon, Portugal, 5–8 February 2009, pp. 331–340. Lisbon, Portugal: INSTICC Press.

Pearl J (1984) *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Boston, MA: Addison-Wesley.

Salzman O, Hemmer M and Halperin D (2015a) On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages. *IEEE Transactions on Automation Science and Engineering* 12(2): 529–538.

Salzman O, Solovey K and Halperin D (2015b) Motion planning for multi-link robots by implicit configuration-space tiling. *Computing Research Repository*, abs/1504.06631.

Sanchez G and Latombe JC (2002) Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. In: *2002 IEEE international conference on robotics and automation (ICRA'02)*, Washington, DC, 11–15 May 2002, pp. 2112–2119. Piscataway: IEEE Press.

Schwartz JT and Sharir M (1983) On the piano movers' problem: III. Coordinating the motion of several independent bodies. *International Journal of Robotics Research* 2(3): 46–75.

Sharir M and Sifrony S (1991) Coordinated motion planning for two independent robots. *Annals of Mathematics and Artificial Intelligence* 3(1): 107–130.

Solovey K and Halperin D (2014) *k*-Color multi-robot motion planning. *International Journal of Robotics Research* 33(1): 82–97.

Solovey K and Halperin D (2015) On the hardness of unlabeled multi-robot motion planning. In: *Robotics: science and systems XI* (ed LE Kavraki, D Hsu and J Buchli), Rome, Italy, 13–17 July 2015.

Solovey K, Yu J, Zamir O and Halperin D (2015) Motion planning for unlabeled discs with optimality guarantees. In: *Robotics: science and systems XI* (ed LE Kavraki, D Hsu and J Buchli), Rome, Italy, 13–17 July 2015.

Spirakis P and Yap C (1984) Strong NP-hardness of moving many discs. *Information Processing Letters* 19(1): 55–59.

Sucan IA, Moll M and Kavraki LE (2012) The open motion planning library. *IEEE Robotics & Automation Magazine* 19(4): 72–82.

The CGAL Project (2011) *CGAL User and Reference Manual*. xxx: CGAL Editorial Board, edition 3.9.

Turpin M, Mohta K, Michael N and Kumar V (2014) Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots* 37(4): 401–415.

van den Berg J and Overmars M (2005) Prioritized motion planning for multiple robots. In: *2005 IEEE/RSJ international conference on intelligent robots and systems (IROS'05)*, pp. 430–435. Piscataway: IEEE Press.

van den Berg J, Snoeyink J, Lin M and Manocha D (2009) Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In: *Robotics: science and systems V* (ed J Trinkle, Y Matsuoka and JA Castellanos), Seattle, USA, 28 June–1 July 2009. Cambridge: MIT Press.

Švestka P and Overmars M (1998) Coordinated path planning for multiple robots. *Robotics and Autonomous Systems* 23(3): 125–152.

Wagner G and Choset H (2015) Subdimensional expansion for multirobot path planning. *Artificial Intelligence* 219: 1–24.

Yap C (1984) Coordinating the motion of several discs. Technical Report 105, Courant Institute of Mathematical Sciences, New York, USA.