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 a roadmap 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 a tensor product of roadmaps for the individual robots. We demonstrate our approach experimentally on scenarios of up to 60 degrees of freedom where our algorithm is faster by a factor of at least ten when compared to existing algorithms that we are aware of.

1 Introduction

Multi-robot motion planning is a fundamental problem in robotics and has been extensively studied. 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 [5, 12, 18, 23], where the robots move on a graph). Moreover, we assume that each robot has its own start and target positions, as opposed to the unlabeled case (see, e.g., [3, 17, 30, 32]).

This work has been supported in part by the 7th Framework Programme for Research of the European Commission, under FET-Open grant number 255827 (CGL—Computational Geometry Learning), by the Israel Science Foundation (grant no. 1102/11), by the German-Israeli Foundation (grant no. 1150-82.6/2011), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

K. Solovey and O. Salzman contributed equally to this paper.

K. Solovey (✉) · O. Salzman · D. Halperin
Blavatnik School of Computer Science, Tel-Aviv University, Tel Aviv, Israel
e-mail: kirilsol@post.tau.ac.il

H.L. Akin et al. (eds.), *Algorithmic Foundations of Robotics XI*,
Springer Tracts in Advanced Robotics 107, DOI 10.1007/978-3-319-16595-0_34

1.1 Previous Work

We assume familiarity with the basic terminology of motion planning. For background, see, e.g., [10, 21]. 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 [28, 29, 36] yet are exponential in the number of robots. The exponential running time, which may be unavoidable [14, 31] can be attributed to the high number of *degrees of freedom* (*dof*)—the sum of the dofs of the individual robots.

For two or three robots, the number of dofs may be slightly reduced [4], 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. [7]. In their work, the 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 (see, e.g., [6, 22]). 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) [16], the *rapidly-exploring random trees* (RRT) [19] 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 [15] 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 [27]. Such a naive 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 [13, 26, 30]. Particularly relevant to our efforts is the work of Švestka and Overmars [33] who suggested to construct 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. A recent work by Wagner et al. [35] 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 [34] 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 Sect. 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 et al. [35], 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 coupling of the robots. Our new approach, which we call dRRT, is an adaptation of the celebrated RRT algorithm [19] for the discrete case of a graph, embedded in Euclidean space.[1] dRRT traverses a composite roadmap that 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 of 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, where MRdRRT is faster by a factor of at least ten when compared to existing algorithms that we are aware of. We show that we manage to solve problems of up to 60 dofs for highly coupled scenarios (Fig. 1).

The organization of this paper is as follows. In Sect. 2 we elaborate on two sampling-based multi-robot motion planning algorithms, namely the composite roadmap approach by Švestka and Overmars [33] and the work on subdimensional expansion and M* by Wagner et al. [34, 35]. In Sect. 3 we introduce the dRRT algorithm. For clarity of exposition, we first describe it as a general pathfinding algorithm for geometrically embedded graphs. In the following section (Sect. 4) we describe the MRdRRT method where dRRT is used in the setting of multi-robot motion-planning problem for the exploration of the implicitly represented composite roadmaps. We show in Sect. 5 experimental results for the algorithm on different scenarios and conclude the paper in Sect. 6 with possible future research directions.

[1] We mention that we are not the first to consider RRTs in discrete domains. Branicky et al. [9] 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.

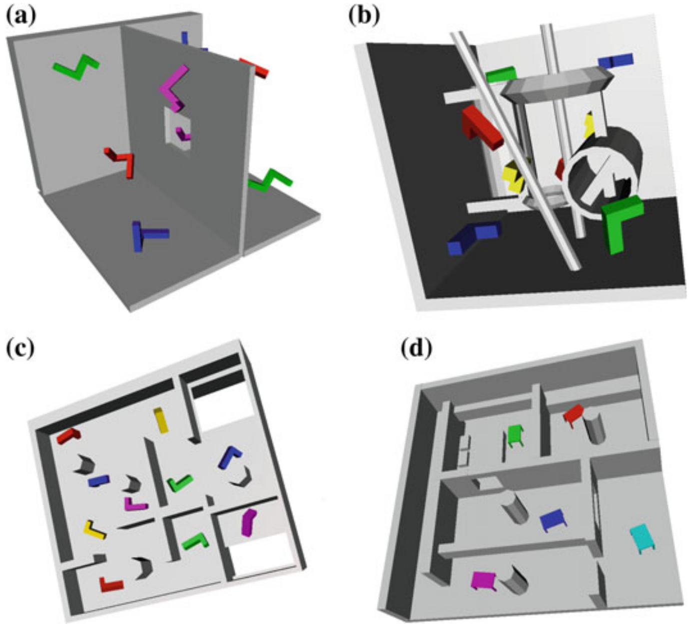

Fig. 1 3D environments with robots that are allowed to rotate and translate (6DOFs). In scenarios **a**–**c** robots of the same color need to exchange positions. **a** Twisty scenario with 8 corkscrew-shaped robots, in a room with a barrier. **b** Abstract scenario with 8 L-shaped robots. **c** Cubicles scenario with 10 L-shaped robots. **d** Home scenario with 5 table-shaped robots that are placed in different rooms. The goal is to change rooms in a clockwise order. The scenario were constructed using meshes that are provided by the Open Motion Planning Library [11] (OMPL 0.10.2) distribution

2 Composite Roadmaps for Multi-robot Motion Planning

We describe the composite roadmap approach introduced by Švestka and Overmars [33]. Here, a Cartesian product of PRM roadmaps of individual robots is considered as a means of devising a roadmap for the entire fleet of 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 et al. suggest [34, 35] to represent 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 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 [2].

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 pair-wise 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.[2]

Remark Throughout this work, unless stated otherwise, we refer to 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 $S = C_1$, $T = C_h$ and 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, cannot be used since the number of vertices of $\mathbb{G}$ alone may reach $O(n^m)$. One may consider the A* algorithm [25] , 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 drawback, since the number of neighbors of every vertex is $O(k^m)$.

Wagner et al. propose an adaptation of A* to the case of a composite roadmap called M* [34]. Their approach exploits the observation that only the motion of some robots has to be coupled in typical scenarios. Thus, planning in the joint C-space is only required for robots that have to be coupled, while the motion of the rest of the 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 coupling, and can cope with large fleets of robots in such settings. However, when the degree of coupling increases, we observed sharp increase in the running time of this algorithm, as it has to consider many neighbors of a visited vertex.

[2] 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.

3 Discrete RRT

We describe a technique which we call *discrete RRT* (dRRT) for pathfinding in implicit graphs that are embedded in a 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 [19] 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 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 trees 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*.

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 (*Direction Oracle*) Given a vertex $v \in V$, and a point $u \in [0, 1]^d$ we define

$$\mathcal{O}_D(v, u) := \underset{v'}{\operatorname{argmin}} \left\{ \angle_v(u, v') | (v, v') \in E \right\}.$$

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

3.2 Description of dRRT

At a high level, dRRT proceeds similar to the RRT algorithm, and we repeat it here for completeness. The dRRT algorithm (Algorithm 1) grows a trees $\mathcal{T}$ which is a subgraphs 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 repeats 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} 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 $q_{\text{new}} \in V$, that is the neighbor of q_{near} in G in the direction of q_{rand} (by the direction oracle $\mathcal{O}_D$). Once q_{new} is found (line 4), it is added to the tree (line 6) with the edge $(q_{\text{near}}, q_{\text{new}})$ (line 7). See an illustration of this process in Fig. 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 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 Sect. 3.3).

Finally, given a path from some node q of $\mathcal{T}$ to t the method RETRIEVE _PATH (Algorithm 1, line 6) returns the concatenation of the path from s to q, with Π.

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 initial problem and here we make a

Algorithm 1 dRRT_PLANNER (s, t)

```
1: T.init(s)
2: loop
3:    EXPAND(T)
4:    Π ← CONNECT_TO_TARGET(T, t)
5:    if not_empty(Π) then
6:       return  RETRIEVE_PATH(T, Π)
```

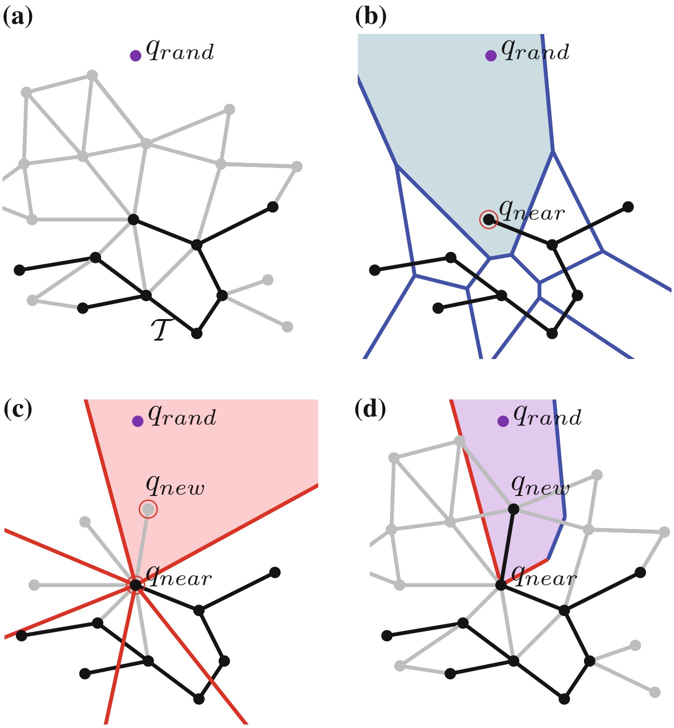

Fig. 2 An illustration of the expansion step of dRRT. The tree $\mathcal{T}$ is drawn with *black* vertices and edges, while the *gray* elements represent the unexplored portion of the graph G. **a** A random point q_{rand} (*purple*) is drawn uniformly from $[0, 1]^d$. **b** The vertex q_{near} of $\mathcal{T}$ 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}, $\text{Vor}(q_{\text{near}})$, is filled with *light blue*. 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 filled with *pink*. The *purple* region in **d** represents $\text{Vor}(q_{\text{near}}) \cap \text{Vor}'(q_{\text{near}}, q_{\text{new}})$

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 highly 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 it finds a solution tends to one as the run-time of the algorithm tends to infinity (when such a solution exists). For simplicity, we show that dRRT possesses a stronger property and with high probability will reveal all the vertices of the traversed graph, assuming this graph is connected.

The proof relies on the assumption that the vertices of the traversed graph G are in *general position*, that is, every pair of distinct vertices are embedded in two distinct points in $\mathbb{R}^d$, and for every triplet of distinct vertices the points in which they are embedded are non-collinear. This issue will be addressed in the following section, where we consider the application of dRRT on a specific type of graphs. The proof does not need to take into consideration the local connector.

Theorem 1 *Let $G = (V, E)$ be a connected graph embedded in $[0, 1]^d$ where the vertices are in general position. Then, with high probability, every vertex of G will be revealed by the dRRT algorithm, given sufficient amount of time.*

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

```
1: for i = 1 → N do
2:    q_rand ← RANDOM_SAMPLE()
3:    q_near ← NEAREST_NEIGHBOR(T, q_rand)
4:    q_new ← O_D(q_near, q_rand)
5:    if q_new ∉ T then
6:       T.add_vertex(q_new)
7:       T.add_edge(q_near, q_new)
```

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

```
1: for q ∈ NEAREST_NEIGHBORS(T, t, K) do
2:    Π ← LOCAL_CONNECTOR(q, t)
3:    if not_empty(Π) then
4:       return Π
5: return ∅
```

Proof Denote by U the set of vertices of $\mathcal{T}$ after the completion of an iteration of the algorithm. Let $v^* \in V \setminus U$ be an unvisited vertex such that there exists $(v, v^*) \in E$, where $v \in U$. We wish to show that the probability that $\mathcal{T}$ will be expanded on the edge (v, v^*), and thus v^* will be added to U, is bounded away from zero. For simplicity we assume that there exists a single vertex $v \in U$ that has an edge to v^*.

Denote by $\mathrm{Vor}(v)$ the *Voronoi cell* [8] of the site v, in the Euclidean (standard) Voronoi diagram of point sites, where the sites are the vertices of U (Fig. 2b). In addition, denote by $\mathrm{Vor}'(v, v^*)$ the Voronoi cell of $\rho(v, v^*)$, in a Voronoi diagram of the ray sites $\rho(v, v^*), \rho(v, u_1), \ldots, \rho(v, u_j)$, where $u_1, \ldots, u_j$ are the neighbors of v in $\mathcal{T}$, not including v^* (Fig. 2c).

Notice that in order to extend $\mathcal{T}$ from v to v^* the random sample q_{rand} in EXPAND (Algorithm 2) has to fall inside $\mathrm{Vor}(v) \cap \mathrm{Vor}'(v, v^*)$. Thus, in order to guarantee that

v^* will be added to $\mathcal{T}$, with non-zero probability, we show that the shared region between these two cells has non-zero measure, namely $|\text{Vor}(v) \cap \text{Vor}'(v, v^*)| > 0$, where $|\Gamma|$ denotes the volume of Γ.

By the general position assumption we can deduce that $|\text{Vor}(v)| > 0$ 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 $\text{Vor}(v)$; 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 $|\text{Vor}(v) \cap \text{Vor}'(v, v^*)| > 0$, otherwise v and v^* are embedded in the same point. $\square$

We note that a more careful analysis can yield an explicit bound on the convergence rate of dRRT. Such a bound may be computed using the size of the smallest cell in the Voronoi diagram of all nodes of G.

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 $\mathcal{O}_D$, which relies solely on the representation of $G_1, \ldots, G_m$. Additionally, we discuss an implementation of the local connector component that 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 Oracle $\mathcal{O}_D$

Recall that given $C \in \mathbb{V}$ and a random sample q, $\mathcal{O}_D(C, q)$ returns C' such that C' 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 Sect. 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 $c'_i = \mathcal{O}_D(c_i, q_i)$ 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 $\mathcal{O}_D(C, q)$. If (C, C') represents a valid edge in $\mathbb{G}$, i.e., no robot-robot collision occurs, we return C'. Otherwise, $\mathcal{O}_D(C, q)$ 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 1) for this specific implementation of $\mathcal{O}_D$, 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 $\mathrm{Vor}(c_i) \cap \mathrm{Vor}'(c_i, c'_i)$ (where in the original proof we required that q will lie in $\mathrm{Vor}(C) \cap \mathrm{Vor}'(C, C')$). Also note that the points in $\mathcal{C}(r_i)$ are in general position, as required by Theorem 1, since they were uniformly sampled by PRM.

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. As our local connector we rely on a framework described by van den Berg et al. [7]. 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.

We decided to use this simple technique in our experiments due to its low cost, in terms of running time, regardless of whether it succeeds finding a solution or not. We wish to mention that we also experimented with M* with a bounded degree of coupling (to avoid considering exponentially many neighbors) as the local connector in our algorithm. However, the ordering algorithm of [7] turned out to be considerably more efficient.

4.3 Probabilistic Completeness of MRdRRT

In order for the motion-planning framework to be probabilistically complete, we still need to show that (i) as the number of samples used for each single-robot roadmap tends to infinity, the composite roadmap will contain a path (if such a path exists) and (ii) that the proof of Theorem 1 still holds when the size of the graph tends to infinity. Indeed, Švestka and Overmars [33] show that the composite roadmap approach is probabilistically complete when the graph-search algorithm is complete. However, in our setting, the graph-search algorithm is only probabilistically complete and the proof may need to be refined as the size of each Voronoi cell tends to zero.

We note that as the composite roadmap is finite, it is easy to modify the dRRT algorithm such that it will be complete. This may be done by keeping a list of exposed nodes that still have unexposed edges. At the end of every iteration of the main loop of dRRT (Algorithm 1, line 2) one node is picked from the list and one of its unexposed edges is exposed (finding an unexposed edge is done in a brute force manner). Although the above modification ensures completeness of dRRT and hence probabilistic completeness of MRdRRT, we are currently looking for an alternative proof that does not require altering the dRRT algorithm.

5 Experimental Results

We implemented MRdRRT for the case of polyhedral robots translating and rotating among polyhedral obstacles (see Fig. 1). We compared the performance of MRdRRT with RRT and an improved (recursive) version of M* that appears in [34]. To make the comparison as equitable as possible, as dRRT does not take into consideration the quality of the solution, we use the *inflated* version of M* [34] with relaxed optimality guarantees.

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 PQP [1] for collision detection, and performs nearest-neighbor queries using the Fast Library for Approximate Nearest Neighbors (FLANN) [24]. Metrics, sampling and interpolation in the 3D environments followed the guidelines presented by Kuffner [20]. 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 i'th 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.

Test scenarios. We report in Table 1 the running times of M* and dRRT for the scenarios. The first three scenarios are especially challenging as they consist of a large number of robots, and require a substantial amount of coordination between them. The fourth scenario ("Home") is more relaxed and consists of only five robots and requires little coordination.

We ran each of the three algorithms 10 times on each scenario. RRT proved incapable of solving any of the test scenarios, running for several tens of minutes until terminating due to exceeding the memory limits. We believe that RRT as-is is not suitable for high-dimensional, coupled, multi-robot motion planning. M* exhibited slightly better performance. For the first three scenarios, which involve multiple robots and require a substantial amount of coordination, it never exceeded a success rate of 40 %. In particular, it often ran out of memory or ran for a very long duration (we terminated it if its running time exceeded ten times the running time of MRdRRT). On the other hand, MRdRRT was stable in its results and managed to solve all the scenarios for each of the 10 attempts. When M* did manage to solve

Table 1 Results for M* and MRdRRT on the scenarios depicted in Fig. 1

Scenario	PRM		M*			MRdRRT				
	n (k)	Time (s)	Visited vertices	Total time	Success rate (%)	Visited vertices (k)	Connect time (s)	Expand time (s)	Total time (s)	Success rate (%)
Twisty	8	10	DNF	DNF	0	8	3.3	6.7	11	100
Abstract	10	24.8	300 k	267 s	30	34	30.4	25.5	55.9	100
Cubicles	10	16.2	27 k	31 s	40	12	16.3	36.8	53.1	100
Home	5	10.1	2 k	3.9 s	100	8	1.5	2.9	4.4	100

We first report the number of vertices (reported in thousands) used in the construction of the single-robot PRM roadmaps and the elapsed time (all times reported are in seconds). Then we report the number of visited vertices, the total running time, and the success rate of M*. A similar report is given for dRRT, but we also specify the duration of the connection phase (using local connector) and the expansion phase. The running times and the amount of explored vertices are averaged over the number of successful attempts one of the first three scenarios, it explored between 2.5 and 10 times the number of vertices that dRRT explored. For the fourth 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 interact at any given time. We mention that MRdRRT was unable to solve scenarios that consist of a substantially larger number of robot than we used in our experiments. We believe that it would be beneficial to consider a stronger *local connector* in such cases.

6 Discussion

In this section we state the benefits of MRdRRT, which consists of an implicitly represented roadmaps for multi-robot motion planning combined with an efficient approach for pathfinding for such roadmaps.

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 naive 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 tens of robots while outperforming our framework. However, in scenarios where a substantial amount of coordination is required between the robots 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 1. This is extremely beneficial when working on $\mathbb{G}$ since it contains vertices which represent essentially the same conformation of the robots, 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 i, v_i is a vertex of V_i that has k neighbors in G_i at distance at most ε. Then the vertex $(v_1, \ldots, v_m) \in \mathbb{V}$ might have as much 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 [15], 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 whether this indeed will lead to optimal paths.

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 with obstacles by forcing the individual robots to move on precalculated individual roadmaps (i.e., G_i). We believe that a similar approach can be used in other settings of motion planning. In particular, we are currently working on a dRRT-based approach for motion planning of a multi-linked robot. The new approach generates an implicitly-represented roadmap, which encapsulates information on configurations and paths between configuration that do not induce self-intersections of 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 obstacles collision-free. Then we use dRRT for pathfinding on the new roadmap, while avoiding self-collision tests and while exploring a small portion of the infinite roadmap.

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

References

1. PQP—A Proximity Query Package. http://gamma.cs.unc.edu/SSV/
2. Graph Product: Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/wiki/Graph_product (2013)
3. Adler, A., de Berg, M., Halperin, D., Solovey, K.: Efficient multi-robot motion planning for unlabeled discs in simple polygons. CoRR arXiv:1312.1038 (2013)
4. Aronov, B., de Berg, M., van der Stappen, A.F., Švestka, P., Vleugels, J.: Motion planning for multiple robots. Discret. Comput. Geom. **22**(4), 505–525 (1999)

5. Auletta, V., Monti, A., Parente, M., Persiano, P.: A linear time algorithm for the feasibility of pebble motion on trees. In: SWAT, pp. 259–270 (1996)
6. van den Berg, J., Overmars, M.: Prioritized motion planning for multiple robots. In: IROS, pp. 430–435 (2005)
7. van den Berg, J., Snoeyink, J., Lin, M., Manocha, D.: Centralized path planning for multiple robots: optimal decoupling into sequential plans. In: RSS (2009)
8. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications, 3rd edn. Springer, Heidelberg (2008)
9. Branicky, M.S., Curtiss, M.M., Levine, J.A., Morgan, S.B.: RRTs for nonlinear, discrete, and hybrid planning and control. In: Decision and Control, pp. 9–12 (2003)
10. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, G., Kavraki, L., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge (2005)
11. Şucan, I.A., Moll, M., Kavraki, L.E.: The open motion planning library. IEEE Robot. Autom. Mag. **19**(4), 72–82 (2012)
12. Goraly, G., Hassin, R.: Multi-color pebble motion on graphs. Algorithmica **58**(3), 610–636 (2010)
13. Hirsch, S., Halperin, D.: Hybrid motion planning: coordinating two discs moving among polygonal obstacles in the plane. In: WAFR, pp. 239–255. Springer, New York (2002)
14. Hopcroft, J., Schwartz, J., Sharir, M.: On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "Warehouseman's Problem". IJRR **3**(4), 76–88 (1984)
15. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. IJRR **30**(7), 846–894 (2011)
16. Kavraki, L.E., Švestka, P., Latombe, J.C., Overmars, M.: probabilistic roadmaps for path planning in high dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
17. Kloder, S., Hutchinson, S.: Path planning for permutation-invariant multi-robot formations. In: ICRA, pp. 1797–1802 (2005)
18. Kornhauser, D.: Coordinating Pebble motion on graphs, the diameter of permutation groups, and applications. M.Sc. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (1984)
19. Kuffner, J.J., LaValle, S.M.: RRT-connect: an efficient approach to single-query path planning. In: ICRA, pp. 995–1001 (2000)
20. Kuffner, J.J.: Effective sampling and distance metrics for 3D rigid body path planning. In: ICRA, pp. 3993–3998 (2004)
21. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
22. Leroy, S., Laumond, J.P., Simeon, T.: Multiple path coordination for mobile robots: a geometric algorithm. In: IJCAI, pp. 1118–1123 (1999)
23. Luna, R., Bekris, K.E.: Push and swap: fast cooperative path-finding with completeness guarantees. In: IJCAI, pp. 294–300 (2011)
24. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: VISSAPP, pp. 331–340. INSTICC Press (2009)
25. Pearl, J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, Reading (1984)
26. Salzman, O., Hemmer, M., Halperin, D.: On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages. In: WAFR, pp. 313–329 (2012)
27. Sanchez, G., Latombe, J.C.: Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. In: ICRA, pp. 2112–2119 (2002)
28. Schwartz, J.T., Sharir, M.: On the piano movers' problem: III. Coordinating the motion of several independent bodies. IJRR **2**(3), 46–75 (1983)
29. Sharir, M., Sifrony, S.: Coordinated motion planning for two independent robots. Ann. Math. Artif. Intell. **3**(1), 107–130 (1991)

30. Solovey, K., Halperin, D.: k-color multi-robot motion planning. In: WAFR, pp. 191–207 (2012)
31. Spirakis, P.G., Yap, C.K.: Strong NP-hardness of moving many discs. Inf. Process. Lett. **19**(1), 55–59 (1984)
32. Turpin, M., Michael, N., Kumar, V.: Computationally efficient trajectory planning and task assignment for large teams of unlabeled robots. In: ICRA, pp. 834–840 (2013)
33. Švestka, P., Overmars, M.: Coordinated path planning for multiple robots. Robot. Auton. Syst. **23**, 125–152 (1998)
34. Wagner, G., Choset, H.: M*: a complete multirobot path planning algorithm with performance bounds. In: IROS, pp. 3260–3267. IEEE (2011)
35. Wagner, G., Kang, M., Choset, H.: Probabilistic path planning for multiple robots with subdimensional expansion. In: ICRA, pp. 2886–2892 (2012)
36. Yap, C.: Coordinating the motion of several discs. Technical report, Courant Institute of Mathematical Sciences, Michigan State University, New York (1984)