Project number IST-25582

CGL

Computational Geometric Learning

Finding a Needle in an Exponential Haystack: Discrete RRT for Exploration of Implicit Roadmaps in Multi-Robot Motion Planning

STREP

Information Society Technologies

Period covered:	November 1, 2012–October 31, 2013
Date of preparation:	October 21, 2013
Date of revision:	October 21, 2013
Start date of project:	November 1, 2010
Duration:	3 years
Project coordinator name:	Joachim Giesen (FSU)
Project coordinator organisation:	Friedrich-Schiller-Universität Jena Jena, Germany

Finding a Needle in an Exponential Haystack: Discrete RRT for Exploration of Implicit Roadmaps in Multi-Robot Motion Planning

Kiril Solovey[†*] and Oren Salzman[†*] and Dan Halperin[†]
[†] Balvatnic School of Computer Science, Tel-Aviv University, Israel

***Abstract*—We present a sampling-based framework for multi-robot motion planning which incorporates an implicit representation of a roadmap with a novel approach for pathfinding in geometrically embedded graphs. Our pathfinding algorithm, *discrete-RRT* (dRRT), is an adaption of the celebrated RRT algorithm, for the discrete case of a graph. By rapidly exploring the high-dimensional configuration space represented by the implicit roadmap, dRRT is able to reach subproblems where minimal coordination between the robots is required. Integrating the implicit representation of the roadmap, the dRRT algorithm, and techniques that are tailored for such subproblems on the implicit roadmap allows us to solve multi-robot problems while exploring only a small portion of the configuration space. We demonstrate our approach experimentally on scenarios of up to 48 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.**

I. 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 [4, 9, 15, 20], 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., [14, 27, 30]).

A. Previous Work

We assume familiarity with the basic terminology of motion planning. For background, see, e.g., [6, 18]. 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 [25] yet are exponential in the number of robots. The exponential running time, which may be unavoidable [11, 28] 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 [3], 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. [32]. 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., [19, 31]). Although efficient in some cases, the approach usually works only for a restricted set of problems.

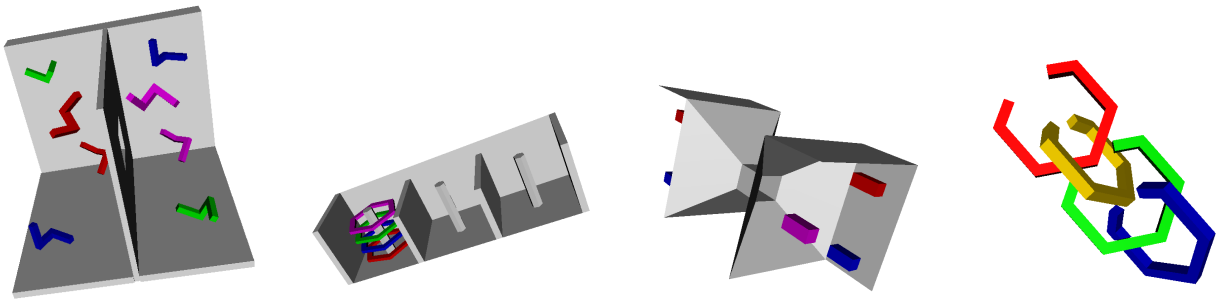
The introduction of *sampling-based* algorithms such as the Probabilistic Roadmap Method (PRM) [13] and the Rapidly-Exploring Random Tree (RRT) [16] and their many variants, had a significant impact on the field of motion planning due to their efficiency, simplicity and applicability to a wide range 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 [12] 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 [24]. 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 [10, 23, 27].

Particularly relevant to our efforts is the work of Švestka

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.



(a) Barrier (b) Hanoi (c) Funnel (d) Chain

Fig. 1. Spatial environments used for the experiments. (a) The Barrier scenario consists of eight robots, in a room with a barrier, where every pair of robots of the same color need to exchange positions. (b) The Hanoi scenario consists of four torus robots. The goal is to move the robots, in the same order, to the rightmost column. (c) 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. (d) 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. The Barrier scenario is provided by the Open Motion Planning Library [7] (OMPL 0.10.2) distribution.

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. A detailed description of these two approaches is provided in Section II below.

B. Contribution

We present a sampling-based algorithm for the multi-robot motion-planning problem. 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 [16] for the discrete case of a graph, embedded in Euclidean space. 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. It is able to reach subproblems where minimal coordination between the robots is required. Integrating the implicit representation of the roadmap, the dRRT algorithm, and techniques tailored for such subproblems on the implicit roadmap allows us to solve multi-robot problems while exploring only a small portion of the C-space.

We mention that we are not the first to consider RRTs in discrete domains. Branicky et al. [5] 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.

We demonstrate the capabilities of our technique on the setting of disk robots moving amidst polygonal obstacles in the plane and for the case of polyhedral robots translating and rotating in space amidst polyhedral obstacles. This is shown experimentally on various challenging scenarios where our algorithm 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 48 dofs for highly coupled scenarios (Figures 1 , 3).

The organization of this paper is as follows. In Section II 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 Section III we introduce the dRRT algorithm. For clarity of exposition, we first describe it as a general pathfinding algorithm for geometrically embedded graphs and only in the following section (Section IV) we utilize dRRT in the setting of multi-robot motion-planning problem for the exploration of implicitly represented composite roadmaps. We show in Section V experimental results for the algorithm on different scenarios and conclude the paper in Section VI with possible future research directions.

II. Composite Roadmaps for Multi-Robot Motion Planning

This section details 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 barrier, Wagner et al. suggest [34, 35] to represent the roadmap *implicitly*. In their work on *subdimensional expansion*

they describe a novel algorithm to find paths on this implicit data structure. The composite roadmap will be used in our method as well and we therefore describe it in detail in the next subsection. In the following subsection we review the subdimensional expansion framework.

A. Composite Roadmaps

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 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 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 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, 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 [22], 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 $O(k^m)$.

B. Subdimensional Expansion and M*

Wagner et al. propose a general strategy called *subdimensional expansion* [35] for motion planning of multiple robots. 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 concept comes into practice when they apply their M* algorithm [34] for pathfinding on the composite roadmap. M* can be described as a variant of the A* algorithm that exploits the fact that the explored roadmap is a result of a tensor product of individual-robot roadmaps. Recall that A* maintains an open list of vertices, which are candidates for expansion towards the target. At each step a vertex is chosen for expansion according to some heuristic function and all the vertex's neighbors are inserted into the open list. M* behaves similarly except that at a vertex chosen for expansion, a limited set of neighbors is inserted into the open list. The edges in this set describe (i) the individual *optimal*[2] motion for robots that do not require coordination and, (ii) all possible motions for robots whose motion has to be coupled.

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, this algorithm suffers from poor running times, as it has to consider many neighbors of a visited vertex. Specifically, the number of neighbors of a vertex is exponential in the degree of coupling. As an example, for a scenario with $k = 15$ (recall that k is the maximal degree of a vertex in a single-robot roadmap) and a degree of coupling equal to five, a single vertex considered by M* in this setting may have up to $15^5 \sim 2^{20}$ neighbors. Thus, M* becomes intractable when the degree of coupling exceeds five.

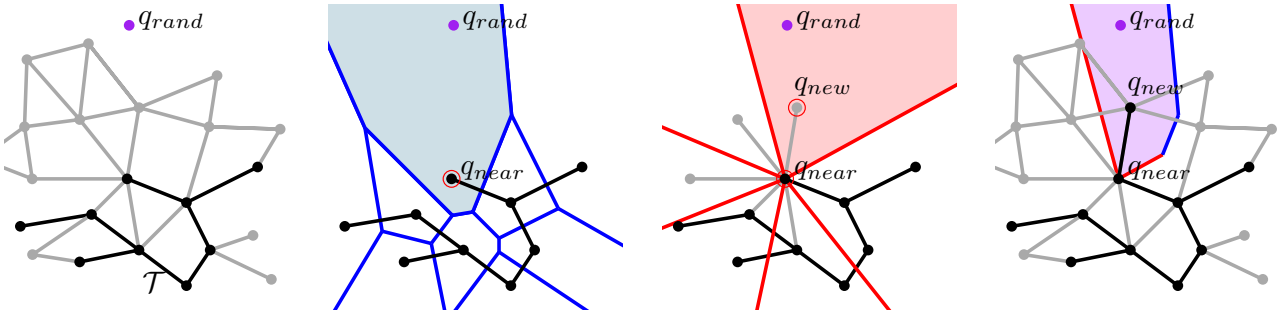
Notice that this occurs when the tensor product is considered, while in the Cartesian case, the degree of each vertex is much smaller ($O(km)$). Although Wagner et al. did not discuss the use of a Cartesian composite roadmap we demonstrate in Section V that M* exhibits very large running times for this case as well.

III. 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 [16]

[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.

[2] Wagner et al. define the cost function for the multi-robot case as the sum of the individual robots' path lengths.



(a) (b) (c) (d)

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 $\text{Vor}(q_{\text{near}}) \cap \text{Vor}'(q_{\text{near}}, q_{\text{new}})$.

for the purpose of exploring discrete geometrically-embedded graphs, instead of a continuous space.[3]

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 two trees that are rooted in s and t and tries to connect them to form a path from s to t. As in RRT, the growth of the trees is achieved by extending them 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*.

A. 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 $u \in [0,1]^d$ the* direction oracle $\mathcal{O}_D$ *returns*

$$\operatorname*{argmin}_{v'} \{\angle_v(u, v') | (v, v') \in E\}.$$

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.

B. 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 two trees $\mathcal{T}_0, \mathcal{T}_1$ that are subgraphs of G, where the former is rooted in s and the latter in t (line 1). The growth of the trees (line 3) is achieved by an expansion towards random samples. Additionally, an attempt to connect the two trees 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 each tree 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 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 tries to connect the trees using the CONNECT_TREES operation (Algorithm 3). Pairs of vertices from the two trees, $q_0 \in \mathcal{T}_0$, $q_1 \in \mathcal{T}_1$, that are candidates for connection are found by sampling a random point (line 2) and its (Euclidean) nearest neighbors in the two trees (lines 3, 4). Given such a pair q_0, q_1, an attempt is made to connect them using the method LOCAL_CONNECTOR (line 5) which is a crucial part of the dRRT algorithm (see Subsection III-C). This process is repeated P times.

[3]To be precise, our algorithm is based on the *bidirectional* RRT variant [16], namely growing two trees from the start and target configurations.

Finally, given a path Π from q_0 to q_1, which connects the two trees, the method RETRIEVE_PATH (Algorithm 1, line 6) returns the concatenation of the path from s to q_0, with Π and with the path from q_1 to t.

C. Local Connector

We show in the following subsection that it is possible that the two trees will find a common vertex 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 nearby vertices q_0, q_1 (where $q_0 \in \mathcal{T}_0$, $q_1 \in T_1$), 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 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).

To avoid a situation where too much effort is put into finding a path between q_0 and q_1 that cannot be easily connected by LOCAL_CONNECTOR, the search is terminated after a predefined criterion has been met.

As an example for a generic local connector one can use the A* algorithm. A natural termination criterion for this example is a bound on the number vertices we allow A* to consider. This generic local connector can be used by the general dRRT algorithm, when no additional property of the explored graph G is known. In the following section, where the adaptation of the dRRT algorithm for the multi-robot problem is discussed, this component will be modified to take advantage of the fact that G is a composite roadmap.

D. 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.

Algorithm 1 dRRT_PLANNER (s, t)

1: $\mathcal{T}_0$.init(s); $\mathcal{T}_1$.init(t)
2: **loop**
3: EXPAND($\mathcal{T}_0$); EXPAND($\mathcal{T}_1$)
4: $\Pi \leftarrow$ CONNECT_TREES($\mathcal{T}_0, \mathcal{T}_1$)
5: **if** not_empty(Π) **then**
6: **return** RETRIEVE_PATH ($\mathcal{T}_0, \mathcal{T}_1, \Pi$)

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: $q_{\text{new}} \leftarrow \mathcal{O}_D(q_{\text{near}}, q_{\text{rand}})$
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_TREES $(\mathcal{T}_0, \mathcal{T}_1)$

1: **for** $i = 1 \rightarrow P$ **do**
2: $q_{\text{rand}} \leftarrow$ RANDOM_SAMPLE()
3: $q_0 \leftarrow$ NEAREST_NEIGHBOR($\mathcal{T}_0, q_{\text{rand}}$)
4: $q_1 \leftarrow$ NEAREST_NEIGHBOR($\mathcal{T}_1, q_{\text{rand}}$)
5: $\Pi \leftarrow$ LOCAL_CONNECTOR(q_0, q_1)
6: **if** not_empty(Π) **then**
7: **return** Π
8: **return** $\emptyset$

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 and we assume for simplicity that dRRT grows only the first tree $\mathcal{T}_0$, while the other stays static throughout the run.

Theorem 2. *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.*

Proof: Denote by U_0 the set of vertices of $\mathcal{T}_0$ after the completion of an iteration of the algorithm. Let $v^* \in V \setminus U_0$ be an unvisited vertex such that there exists $(v, v^*) \in E$, where $v \in U_0$. We wish to show that the probability that $\mathcal{T}_0$ will be expanded on the edge (v, v^*), and thus v^* will be added to U_0, is bounded away from zero. For simplicity we assume that there exists a single vertex $v \in U_0$ that has an edge to v^*.

Denote by $\text{Vor}(v)$ 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_0 (Figure 2(b)). In addition, denote by $\text{Vor}'(v, v^*)$ 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 $\mathcal{T}_0$, not including v^* (Figure 2(c)).

Notice that in order to extend $\mathcal{T}_0$ from v to v^* the random sample q_{rand} in EXPAND (Algorithm 2) has to fall inside $\text{Vor}(v) \cap \text{Vor}'(v, v^*)$. Thus, in order to guarantee that v^* will be added to $\mathcal{T}_0$, 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. ■

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.

IV. Applying dRRT to Composite Roadmaps

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. Specifically, 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.

A. Oracle $\mathcal{O}_D$

Recall that given $C \in \mathbb{V}$ and a random sample q, $\emptyset_D(C, q)$ 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 $\rho(C, C')$, where ρ is as defined in Section III-D.

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—traversing all the neighbors of c_i in G_i Let $C' = (c'_1, \ldots, c'_m)$ be a candidate for the result of $\emptyset_D(C, q)$. If (C, C') represents a valid edge in $\mathbb{G}$, i.e., no robot-robot collision occurs, we return C'. Otherwise, $\emptyset_D(C, q)$ returns $\emptyset$. In this case, the new sample is ignored and another sample is drawn in the EXPAND phase (Algorithm 2).

The generalization of the completeness proof of the dRRT (Theorem 2) for the 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 $\text{Vor}(c_i) \cap \text{Vor}'(c_i, c'_i)$ (where in the original proof we required that q will lie in $\text{Vor}(C) \cap \text{Vor}'(C, C')$). Since we restrict the proof for the vertices of the individual roadmaps, we can safely assume that the points (in $\mathcal{C}(r_i)$) are in general position (as required by Theorem 2), since they were randomly sampled by the PRM.

B. Local Connector using M*

Recall that in the general dRRT algorithm the local connector is used for connecting two vertices of a graph when it is believed that the pathfinding problem is simpler, as the distance between the two vertices is smaller than the distance between the original problem's source and target. In our setting, it is natural to measure the hardness of a given problem instance by the required degree of coupling. Thus, we assume that the problem of connecting two vertices by the local connector requires a lower degree of coupling than the input problem.

As our local connector we use M*, as this method is known to be extremely efficient on scenarios where the degree of coupling is low. We mention that it is still possible that the local problem will turn out too challenging for the M* algorithm. In order to avoid situations where M* spends too much time on problems it is incapable to solve (due to prohibitive memory requirements), we terminate M* when it considers the coupling of too many robots. Alternatively, it is also possible to detect decoupled instances a-priori using the *optimal decoupling method* [32], and only then use M*.

C. Probabilistic Completeness of the Framework

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 2 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. Alas, 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 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).

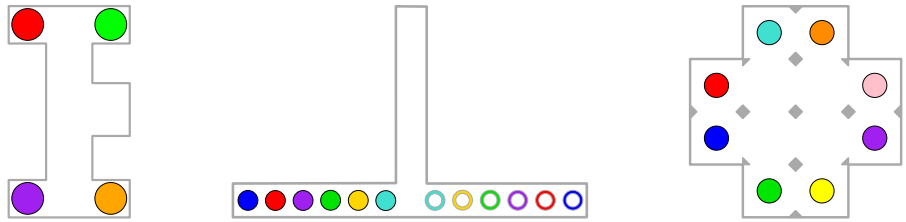
Although the above modification ensures completeness of dRRT and hence probabilistic completeness of our motion-planning framework, we are currently looking for an alternative proof that does not require altering the dRRT algorithm.

V. Experimental Results

We implemented our framework for the case of disc robots translating in the plane amidst polygonal obstacles and for the case of polyhedral robots translating and rotating among polyhedral obstacles. We compared the performance of our algorithm with M*. The first test case studies the effect the degree of coupling on the algorithms, while the second examines the behavior of the algorithms on highly coupled scenarios and scenarios with a large number of dofs.

A. Implementation Details

The algorithm was implemented in C++. The experiments were conducted on a laptop with an Intel i5-3230M 2.60GHz processor with 16GB of memory, running 64-bit Windows 7. We implemented a generic framework for multi-robot motion planning based on composite roadmaps. It has an implementation of the single-robot PRM algorithm (using the Boost Graph Library (BGL) [26]), collision detection (using the Computational Geometry Algorithms Library (CGAL) [29] for the



(a) Bay (b) T-shape (c) Room

Fig. 3. Highly-coupled settings in planar environments used for the experiments. (a) The red and green robots (at the top) need to change positions with the purple and orange robots (at the bottom), (b) the robots need to interchange their order by entering the narrow corridor (start and target positions are depicted by fully-colored disks and annuli, respectively), (c) opposite robots need to interchange positions

2D scenarios and PQP [1] for the 3D scenarios), and nearest neighbor queries (using the Fast Library for Approximate Nearest Neighbors (FLANN) [21]). Metrics, sampling and interpolation in the 3D environments followed the guidelines presented by Kuffner [17]. The framework's design allowed us to compare different search strategies on the same composite roadmap.

Parameters of the dRRT Algorithm. 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 iteration i each EXPAND (Algorithm 2) call performs 2^i iterations ($N = 2^i$), while CONNECT_TREES performs i iterations ($P = i$). A call to the local connector was terminated when the number of robots that had to be coordinated exceeded 4.

Recursive M*. Throughout this paper we addressed the M* algorithm, yet Wagner et al. also described a more efficient variant named recursive M* or rM* that subdivides separated but simultaneously interacting sets of robots to individual subproblems solved recursively. They showed that rM* is faster and has better memory consumption and hence we compare the dRRT running times with rM*.

Relaxing the Optimality of rM*. M* and rM* retrieve the optimal path between two vertices of the composite roadmap. As dRRT does not take into account path quality, we use the inflated version [34] of rM* that biases the search towards the leaves of the search tree that are close to the goal. These are candidates where a solution is more likely to be found quickly.

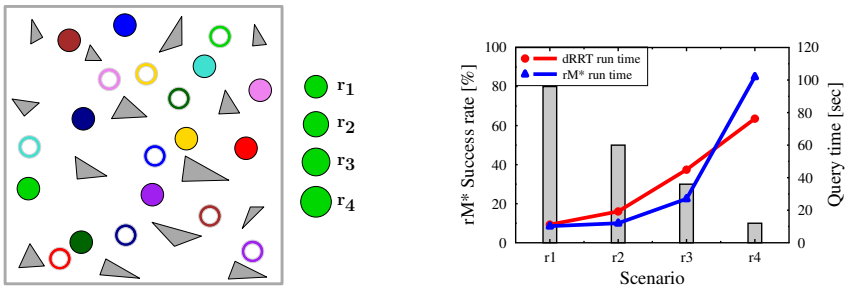
Cartesian vs. Tensor Composite Roadmap. As we described earlier, M* and rM* may exhibit high memory consumption when using a tensor composite roadmap. If one considers using a Cartesian roadmap, this limitation vanishes but the algorithm's running times increase drastically. This creates an interesting tradeoff between a fast, memory-consuming algorithm and a slow, yet memory-efficient variant. When using a tensor composite roadmap our approach is not affected by the high degree of each vertex, due to the use of the oracle. Thus, such a tradeoff does not exist and there is no motivation to consider using dRRT with a Cartesian composite roadmap. We ran rM* in all scenarios with both types of composite roadmaps and note that in all cases when rM* ran on a Cartesian roadmap, we either had to terminate the run after a timeout of 15 minutes or the average running times were larger than ours by a factor of 15. Due to this fact we only present the results when using a tensor composite roadmap.

B. Test Scenarios

The first set of tests consists of a workspace with 10 randomly placed disk-robots that move to random goals amidst random obstacles. We gradually increased the coupling by increasing the disk's radii (see Figure 4 (a)). This allowed us to compare the performance of dRRT and rM* for different levels of coupling. The second set of scenarios consists of highly coupled scenarios and scenarios with a large number of dofs (see Figure 1 and Figure 3) that no other pathfinding algorithm was able to solve within a reasonable time frame. We tried to run A*, M* and rM* on the implicit roadmap generated for these scenarios but the program terminated due to insufficient memory resources (or due to high running times). We mention that it has been shown that other sampling-based methods such as PRM and RRT, are incapable of dealing with such scenarios [27, 35], and therefore were not tested. All times reported are the average of 10 runs on each scenario after removing minimal and maximal outliers.

Gradual Increase of Coupling. For the scenarios requiring a gradual increase of coupling, the rM* failed to find a path on several instances (either due to insufficient memory or due to running times exceeding 15 minutes). The results are presented in Figure 4. dRRT's success rate is 100% for all tests while rM*'s success rate diminishes as the problem becomes highly coupled. When the rM* succeeds in finding a path, both algorithm's running times are comparable, and for the easier cases rM* has slightly better running times.

Highly Coupled Scenarios. We report in Table I the running times of our framework for the highly coupled scenarios depicted in Figure 1 and Figure 3. We stress that these scenarios represent problems considered intractable for the other planners. The amount of vertices expanded by dRRT may seem quite large yet one should notice the very high (bound on the) number of vertices represented by the implicit composite roadmap. As mentioned earlier, the dRRT algorithm explores a tiny fraction of the C-space while reaching locally simple problems that are solvable by M*.



(a) Random polygons scenario (b) Comparison results

Fig. 4. Comparing dRRT and rM*. (a) Ten randomly placed robots that move to random goals (start and target locations are depicted by colored disks and annuli, respectively). The robots' radii are gradually increased creating an increase in the coupling required. Different radii are depicted on the right-hand side of the figure ranging from smallest (r1) to largest (r4). (b) Results for the different radii for the scenario. Success rate for rM* is depicted by shaded bars (dRRT had 100% success rate in all cases). Query time for the dRRT and rM* algorithms is depicted by line segments.

Sceanrio	num of robots	dof	total time	PRM time	expand time	connect time	PRM size	expanded vertices	$O(\|\mathbb{V}\|)$
Bay	4	8	45.1	0.1	39.7	5.4	3×10^2	3.8×10^5	8.1×10^9
T-shape	6	12	3.5	0.1	0.4	3.1	3×10^2	5.3×10^3	7.3×10^{14}
Room	8	16	22.1	0.1	12.4	9.7	3×10^2	3.2×10^5	6.5×10^{19}
Barrier	8	48	350	16	196	138	10^4	3×10^4	3.9×10^{29}
Hanoi	4	24	572	55	355	161	2×10^4	4.5×10^5	1.6×10^{17}
Funnel	6	36	500	17	373	110	5×10^3	1.2×10^5	1×10^{16}
Chain	4	24	17.4	8.7	5.4	3.3	1.5×10^4	6×10^2	6.25×10^{14}

TABLE I

Results for our dRRT algorithm applied to the scenarios depicted in Figure 1 and Figure 3. Total query time consist of single PRM construction time, expand time by the dRRT algorithm and connect time by the local connector (all times reported are in seconds)). Additionally we report the number of vertices in a single PRM roadmap G_i, number of vertices (of $\mathbb{G}$) expanded by the algorithm, and a bound on the total number of vertices represented by the implicit composite roadmap. We mention that the maximal vertex degree of a vertex in G_i was fixed to be 15 throughout the experiment.

VI. Future Work - Towards Optimality

Currently, our algorithmic framework is concerned with finding *a* solution, and our immediate future goal is to modify it to provide a solution with quality guarantees, possibly by making dRRT optimal by taking an approach similar to the continuous RRT* algorithm [12], 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.

VII. Acknowledgements

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*. URL http://gamma.cs.unc.edu/SSV/.

[2] Graph Product — Wikipedia, The Free Encyclopedia, 2013. URL http://en.wikipedia.org/wiki/Graph_product.

[3] B. Aronov, M. de Berg, A. F. van der Stappen, P. Švestka, and J. Vleugels. Motion Planning for Multiple Robots. *Discrete & Computational Geometry*, 22(4):505–525, 1999.

[4] Vincenzo Auletta, Angelo Monti, Mimmo Parente, and Pino Persiano. A linear time algorithm for the feasibility of pebble motion on trees. In *SWAT*, pages 259–270, 1996.

[5] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan. RRTs for Nonlinear, Discrete, and Hybrid Planning and Control. In *Decision and Control*, pages 9–12, 2003.

[6] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, G. Burgard, L. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[7] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[8] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, third edition, 2008.

[9] Gilad Goraly and Refael Hassin. Multi-color pebble motion on graphs. *Algorithmica*, 58(3):610–636, 2010.

[10] S. Hirsch and D. Halperin. Hybrid Motion Planning: Coordinating Two Discs Moving among Polygonal Obstacles in the Plane. In *WAFR*, pages 239–255. Springer, 2002.

[11] J.E. Hopcroft, J.T. Schwartz, and M. Sharir. On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE-Hardness of the "Warehouseman's Problem". *IJRR*, 3(4):76–88, 1984.

[12] S. Karaman and E. Frazzoli. Sampling-Based Algorithms for Optimal Motion Planning. *IJRR*, 30(7):846–894, 2011.

[13] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. Overmars.

probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
[14] S. Kloder and S. Hutchinson. Path Planning for Permutation-Invariant Multi-Robot Formations. In *ICRA*, pages 1797–1802, 2005.
[15] Daniel Kornhauser. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. M.Sc. thesis, Department of Electrical Engineering and Computer Scienec, Massachusetts Institute of Technology, 1984.
[16] J. J. Kuffner and S. M. LaValle. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *ICRA*, pages 995–1001, 2000.
[17] James J. Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In *ICRA*, pages 3993–3998, 2004.
[18] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
[19] S. Leroy, J. P. Laumond, and T. Simeon. Multiple Path Coordination for Mobile Robots: A Geometric Algorithm. In *IJCAI*, pages 1118–1123, 1999.
[20] R. Luna and K. E. Bekris. Push and Swap: Fast Cooperative Path-Finding with Completeness Guarantees. In *IJCAI*, pages 294–300, 2011.
[21] M. Muja and D. G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *VISSAPP*, pages 331–340. INSTICC Press, 2009.
[22] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.
[23] O. Salzman, M. Hemmer, and D. Halperin. On the Power of Manifold Samples in Exploring Configuration Spaces and the Dimensionality of Narrow Passages. *WAFR*, pages 313–329, 2012.
[24] G. Sanchez and J.-C. Latombe. Using a PRM Planner to Compare Centralized and Decoupled Planning for Multi-Robot Systems. In *ICRA*, pages 2112–2119, 2002.
[25] J. T. Schwartz and M. Sharir. On the Piano Movers' Problem: III. Coordinating the Motion of Several Independent Bodies. *IJRR*, 2(3): 46–75, 1983.
[26] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. *The Boost Graph Library User Guide and Reference Manual*. Addison-Wesley, 2002.
[27] K. Solovey and D. Halperin. k-Color Multi-Robot Motion Planning. *WAFR*, pages 191–207, 2012.
[28] Paul G. Spirakis and Chee-Keng Yap. Strong NP-Hardness of Moving Many Discs. *Inf. Process. Lett.*, 19(1):55–59, 1984.
[29] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 3.9 edition, 2011. URL http://www.cgal.org/.
[30] M. Turpin, N. Michael, and V. Kumar. Computationally Efficient Trajectory Planning and Task Assignment for Large Teams of Unlabeled Robots. In *ICRA*, pages 834–840, 2013.
[31] J. van den Berg and M. Overmars. prioritized motion planning for multiple robots. In *IROS*, pages 430 – 435, 2005.
[32] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha. Centralized Path Planning for Multiple Robots: Optimal Decoupling into Sequential Plans. In *RSS*, 2009.
[33] P. Švestka and M. Overmars. Coordinated Path Planning for Multiple Robots. *Robotics and Autonomous Systems*, 23:125–152, 1998.
[34] G. Wagner and H. Choset. M*: A Complete Multirobot Path Planning Algorithm with Performance Bounds. In *IROS*, pages 3260–3267. IEEE, 2011.
[35] G. Wagner, M. Kang, and H. Choset. Probabilistic Path Planning for Multiple Robots with Subdimensional Expansion. In *ICRA*, pages 2886–2892, 2012.