An Effective Algorithmic Framework for Near Optimal Multi-Robot Path Planning

Jingjin Yu  
Daniela Rus

Abstract

We present an algorithmic framework for solving multi-robot path planning problems in general, two-dimensional, continuous environments while seeking to minimize globally the task completion time. The framework obtains its high levels of effectiveness through the composition of an optimal discretization of the continuous environment and the subsequent fast, near-optimal resolution of the resulting discrete planning problem. This principled approach achieves orders of magnitudes better performance with respect to both speed and the supported robot density. For a wide variety of environments, our method is shown to compute globally near-optimal solutions for 50 robots in seconds with robots packed close to each other. In the extreme, the method can consistently solve problems with hundreds of robots that occupy over 30% of the free space.

I. INTRODUCTION

We study the problem of planning collision-free paths for multiple labeled disc robots operating in two-dimensional, multiply-connected, continuous environments (i.e., environments with holes). The primary goal of this work is to develop a practical, extensible framework toward the efficient resolution of multi-robot path planning (MPP) problem, in which the robots are densely packed, while simultaneously seeking to minimize globally the task completion time. The framework is composed of two key algorithmic components, executed in an sequential order. Using the example illustrated in Fig. 1(a), first, we compute the configuration space for a single robot, over which an optimal lattice structure is overlaid (Fig. 1(b)). Using the lattice structure as a roadmap, each start (resp., goal) location is assigned to a nearby node of the roadmap as its unique discrete start (resp., goal) location, which translates the continuous problem into a discrete one (Fig. 1(c)). Then, a state-of-the-art discrete planning algorithm is applied to solve the roadmap-based problem near-optimally (Fig. 1(d)). Through the tight composition of these two algorithmic components, our framework proves to be highly effective in a variety of settings, pushing the boundaries on optimal multi-robot path planning to new grounds in terms of the number of robots supported and the allowed robot density.

Fig. 1. An illustrative example of our algorithmic framework. a) A problem instance with three disc robots. The start and goal locations are indicated by the colored vehicle cartoons and their faded versions, respectively. b) The configuration space (shaded area) for a single robot and the fitted hexagonal lattice. The blue circles are the start positions, and the red circles are the goal positions. c) The discrete abstraction of the original problem. d) Solution to the original problem.

Jingjin Yu and Daniela Rus are with the Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology. E-mails: {jingjin, rus}@csail.mit.edu.
Related work. MPP finds applications in a wide spectrum of domains such as navigation [Alonso-Mora et al. (2014), Snape et al. (2014), manufacturing and assembly [Knepper and Rus (2012)], warehouse automation [Wurman et al. (2008)], computer video games Snape (2012), and microfluidics Griffith and Akella (2005). Given the important role it holds in robotics-related applications, MPP problems has received considerable attention in robotics research with dedicated study on the subject dating back at least three decades [Schwartz and Sharir (1983), in which a centralized approach is taken that considers all robots as a single robot in a high dimensional configuration space. Because the search space in such problems grows exponentially as the number of robots increases linearly, a centralized approach [Schwartz and Sharir (1983), although complete, would be extremely inefficient in practice. As such, most ensuing research take the approach of decomposing the problem. One way to achieve this is by assigning priorities to the robots so that robots with higher priority take precedence over robots with lower priority [Buckley (1989), Erdmann and Lozano-Pérez (1986)]. Another often adopted partitioning method is to plan a path for each robot separately without considering robot-robot interaction. The paths are then coordinated to yield collision free paths [Bjen and Lee (1992), O’Donnell and Lozano-Pérez (1989)]. Following these initial efforts, the decomposition scheme is further exploited and improved [Christ et al. (2005), LaValle and Hutchinson (1998), Peng and Akella (2002), van den Berg and Overmars (2005), van den Berg et al. (2009), Svestka and Overmars (1998)]. Many of the mentioned works also consider optimality in some form. We emphasize that, since finding feasible solution for MPP is already PSPACE-hard [Hopcroft et al. (1984), i.e., no polynomial-time complete algorithm may even exist for such problems unless P = PSPACE, computing globally near-optimal solution for a large number of robots is an extremely challenging problem.

Recent years have witnessed a great many new approaches being proposed for solving MPP. One such method, velocity-obstacles [van den Berg et al. (2008, 2011), which can be traced back to Kant and Zucker (1986), explicitly looks at velocity-time space for coordinating robot motions. In Griffith and Akella (2005), mixed integer programming (MIP) models are employed to encode the interactions between the robots. A method based on network-flow is explored in I. Karamouzas (2012). In Peasgood et al. (2008), similar to our approach, an $A^*$-based search is performed over a discrete roadmap abstracted from the continuous environment. In Solovey et al. (2014), discrete-RRT (d-RRT) is proposed for the efficient search of multi-robot roadmaps. Lastly, as a special case of MPP in continuous settings, efficient algorithms are proposed [Solovey et al. (2015), Turpin et al. (2013)] for interchangeable robots (i.e., in the end, the only requirement is that each goal location is occupied by an arbitrary robot). At the same time, discrete (e.g., graph-based) MPP has also been a subject of active investigation. This line of research originates from the mathematical study of the 15-puzzle and related pebble motion problems [Kornhauser et al. (1984), Wilson (1974). Since then, many heuristics augmenting the $A^*$ algorithm have been proposed for finding optimal solution, e.g., Ryan (2008), Standley and Kort (2011), Wagner and Choset (2011), to name a few. These heuristics essentially explore the same decoupling idea used in the continuous case to trim down the search space. A method based on network-flow also exists here [Yu and LaValle (2013)]. Some of these discrete solutions, such as [Kornhauser et al. (1984)], have helped solving continuous problems [Krontiris et al. (2012), Solovey and Halperin (2012),

Contributions. Our work brings two contributions toward solving MPP effectively and optimally. First, we introduce a two-phase framework that allows any roadmap building (i.e., discretization) method to be combined with any suitable discrete MPP algorithm for solving continuous MPP problems. The framework achieves this by imposing a partial collision avoidance constraint during the roadmap building phase while preserving near-path optimality. Second, we deliver a practical integrated algorithmic solution for computing near optimal paths for a large number of robots. The effectiveness of the overall approach depends critically on a general purpose, highly effective discrete MPP algorithm. To the best of our knowledge, we present the first such algorithm that can quickly plan near optimal, continuous paths for hundreds of robots densely populated in multiply-connected environments.\footnote{Warehousing systems from Kiva Systems [Wurman et al. (2008)] can work effectively with hundreds of robots. However, these robots essentially live on a grid within a structured environment.}

In addition, the software developed along the paper will be made available upon the publication of this paper, which can be readily used as a subroutine that quickly computes candidate solutions for coordinating a large number of robots in a planar workspace.

Paper organization. The rest of the paper is organized as follows. We formulate the MPP problem in Section II. In Section III we describe the first component of the algorithmic framework on roadmap-based problem construction. In Section IV we detail the second component of the framework on effective computation of near-optimal solutions to the roadmap-based discrete problem. In Section V we demonstrate the effectiveness of our framework. We conclude and discuss future work in Section VI.
In addition to planning collision-free paths, we are interested in optimizing path quality. Our particular focus in this paper is minimizing the global task completion time, also commonly known as makespan. Let $P = \{p_1, \ldots, p_n\}$ denote a feasible path set with each $p_i$ a continuous function, defined as

$$p_i : [0, t_f] \rightarrow C_f, p_i(0) = s_i, p_i(t_f) = g_i.$$  

The makespan objective seeks solutions that minimize $t_f$. In other words, let $P$ denote the set of all solution path sets, the task is to find a path set with $t_f$ close to

$$t_{min} := \min_{P \in P} t_f(P).$$  \hspace{1cm} (1)

We emphasize that the aim of this work is a method for quickly solving “typical” problem instances with many robots and high robot density (i.e., the ratio between robot footprint and the free space is high) with optimality assurance. By typical, we mean that: (i) the start location and goal locations are reasonably separated, (ii) a start or goal location is not too close to static obstacles in the environment, and (iii) there are no narrow passages in the environment that cause the discretized roadmap structure to have poor connectivity.

More formally, we assume that assumptions (i) and (ii), respectively, take the forms

$$\forall 1 \leq i, j \leq n, \quad |s_i - s_j| \geq 4, \quad |g_i - g_j| \geq 4$$  \hspace{1cm} (2)

and

$$\forall p \in \{S \cup G\}, \quad |p - q| \leq \sqrt{5} \Rightarrow q \in W.$$  \hspace{1cm} (3)

For (iii), the discretized roadmap should capture the topology of the continuous environment well. To be more concrete, see Figure 2. a) An environment with a discretization that does not capture its original topology. b) The roadmap/graph after restore connectivity (the operations are performed automatically from our code), which then captures the topology of the original environment.

Remark. We provide these assumptions only to suggest situations in which our framework is expected to perform well. In our evaluation, these assumptions are not enforced. We in fact greatly relax $4$ (from 4 to 2.5) and do not enforce $5$ at all. We also give an efficient subroutine for restoring connectivity when assumption (iii) is not satisfied. For example, the routine, when applied to the example in Fig. 2(a), yields the result in Fig. 2(b), which is a screen capture from our program. We also emphasize that, given that optimal MPP is an extremely challenging task computationally [Hopcroft et al. (1984)] and our focus on method effectiveness, we do not consider the problem from the angle of solution completeness.

III. CONSTRUCTION OF ROADMAP-BASED DISCRETE PROBLEM

We solve the proposed problem with two key algorithmic components–discretization followed by resolution of the roadmap-based problem. In this section, we describe the first key component of our algorithmic framework–the construction of the roadmap-based, discrete problem. This phase is carried out in three steps: (i) selecting and overlaying a regular lattice structure over the configuration space, (ii) restore environment connectivity that is lost in the discretization process, and (iii) “snapping” start and goal locations to roadmap nodes.

2Note that our algorithmic framework also applies to other time- and distance-based optimality objectives through the use of an appropriate discrete planning algorithm.
A. Lattice Selection and Imposition

1) Selection of appropriate lattice structures.: In selecting the appropriate lattice structure, we hope to allow the packing of more robots simultaneously on the resulting roadmap and obtain the structure fast. Clearly, if an insufficient number of nodes exists in the roadmap, the resulting discrete problem can be crowded with robots, which is difficult to solve and may not even have a solution. On the other hand, to allow a clean separation between the roadmap building phase and the discrete planning phase of the framework, the nodes cannot be too close to each other, e.g., two robots occupying two different nodes should not be in collision. Moreover, it is desirable that two robots moving on different edges in parallel will not collide with each other.

Considering all these factors together, we resort to adopting uniform tilings of the plane [Robert (1978)]. A uniform tiling of the plane is a regular network structure that can be repeated infinitely to cover the entire two-dimensional plane. Due to the regularity of uniform tilings, it is computationally easy to overlay a tiling pattern over $C_f$. Choosing such a tiling then relieves us from selecting each node for the roadmap individually. Over the 11 uniform tilings of the plane [Robert (1978)], we computed the density of robots supported by each. To allow concurrent moves of robots on nearby edges, take square tiling as an example, a square must have a side length of $4/\sqrt{2}$ to avoid potential collision incurred by such moves (see, e.g., Fig. 3(a)). Indeed, it is straightforward to show that the closest inter-robot distance is reached when two robots are in the middle of two edges connecting to the same node. For hexagonal tilings, this results in a minimum side length of $4/\sqrt{3}$ (Fig. 3(b)).

![Fig. 3. Minimum distance between robots. To ensure no collision when executing a discrete plan, the distance between two lattice nodes must be $4/\sqrt{2} + \epsilon$ for square tilings (a) and $4/\sqrt{3} + \epsilon$ for hexagonal tilings (b). At exactly $4/\sqrt{2}$ (resp. $4/\sqrt{3}$) the robots will touch when reaching the midpoint of the edge. The contact point is shown as a red circle in both figures.](image)

After obtaining the required side length parameters for all 11 tilings, the maximum robot density allowed by these tilings can then be computed. We compute the density by assuming that all nodes of the regular tiling patterns are occupied by robots and compute the ratio between the area occupied by robots and the unoccupied free space. For an infinite lattice with no obstacles, the hexagonal tiling is the best with about 45% density, followed by the square tiling with roughly 39% density. Triangular tilings have a density of only 23%. This leads us to choose hexagonal lattices as the base structure of the discrete roadmap.

![Fig. 4. Efficient computation of the hexagonal lattice that falls inside $C_f$.](image)

2) Imposing the lattice structure.: After deciding on using a hexagonal lattice, we need an effective method of imposing the structure on $C_f$. We start by making an arbitrary alignment between a sufficiently large piece of the infinite hexagonal lattice and the continuous environment (Fig. 4). Then, we look at one C-space obstacle (including the outer boundary) at a time. For each obstacle, we pick an arbitrary vertex on the boundary (red dot in Fig. 4) and locate the hexagon from the lattice it belongs to (in this case, the shaded hexagonal with label “1” in Fig. 4). We then follow the obstacle boundary and find all (green) edges of the lattice that intersect the boundary. Using these edges, we can then compute the part of the lattice that falls inside $C_f$ (Fig. 4(b)). The computation takes time linear with respect to the complexity of the environment, which is trivial in comparison to the time it takes in doing discrete planning.

3These tilings are: triangular, trihexagonal, square, elongated triangular, hexagonal, truncated square, truncated trihexagonal, truncated hexagonal, snub square, rhombitrihexagonal, snub hexagonal.
B. Restore configuration space connectivity

We now address how we may ensure that the topology of $C_f$ is preserved in the discrete roadmap. For the problem given in Fig. 3(a), for each C-space obstacle, it is easy to obtain the smallest cycle on the lattice enclosing the obstacle (e.g., the green and red cycles in Fig. 5). Then, for each pair of obstacles, we check whether the corresponding enclosing cycles share non-trivial interior and if so, locate a minimum segment on the overlapping section (e.g., the red segments between the two orange nodes in Fig. 5). Using visibility graph [Lozano-Pérez and Wesley (1979), we may then restore the lost connectivity and obtain the roadmap shown in Fig. 3(b).

Fig. 5. Smallest cycles fully surrounding the two $C_f$ obstacles.

Remark. In the process of restoring connectivity, it is possible that the resulting roadmap cannot guarantee that simultaneous movements of disc robots are collision-free. Without getting into details, we mention that this issue can be fully addressed by sacrificing some time optimality.

We also note that the preservation of the connectivity or topology of the continuous environment can be crucially important. A better connected environment has a more diverse set of candidate paths, making the resulting problem easier to solve. Perhaps more importantly, the preservation of the connectivity of $C_f$ is essential to preserving path optimality. For a roadmap built from an overlaid square lattice, given a shortest path $p \subset C_f$ between two points, due to the strong equivalence between the Euclidean metric and the Manhattan metric, the shortest path $p$ and the corresponding shortest path $p'$ on the square lattice-based roadmap are within a constant factor multiple of each other for any reasonably long path $p$ (that is, $\text{length}(p) \ll 1$ does not hold). The same argument applies to the roadmap-based hexagonal lattices. Without obstacles, the ratio $\text{length}(p')/\text{length}(p)$ over a long path $p$ is bounded by $\sqrt{2}$ for square lattices and roughly the same for hexagonal lattices. The ratio is largely the same when obstacles are present. On the other hand, if the connectivity of $C_f$ is not preserved, then it becomes possible that $\text{length}(p')/\text{length}(p)$ is arbitrarily large. An example is given in Fig. 6.

Fig. 6. Suppose that the start and goal locations are at the center of the blue and the red discs, respectively. If the robot does not find the narrow passage on the left, it then needs to travel through a winding path on the right. By extending the width of the environment, we can make the winding path arbitrarily long when compared to the shortest path.

Once we establish that the roadmap preserves the near-optimality on path length, the same applies to time optimality. Given the preservation of near-optimality of individual paths, it does not directly imply that an optimal solution to the abstracted discrete problem also preserves optimality with respect to the original continuous problem, in terms of time or distance. However, our computational experiments show that this is generally the case when $C_f$ has good connectivity.

C. Snapping Start and Goal Locations to Roadmap Nodes

After the full roadmap is built, each start or goal location in $S \cup G$ must be associated with a nearby roadmap node. We call this process snapping. For the snapping step, for each $s_i \in S$, we simply associate $s_i$ with the closest roadmap node that $s_i$ can reach without colliding with another $s_j \in S$. The same process is performed for all $g_i \in G$ With the separation assumptions (2) and (3), this is almost always possible. In particular, (2) implies that each hexagon from the lattice contains (roughly) at most one start and one goal location. Therefore, the number of nodes on the roadmap is at least twice the number of robots. In rare cases when conflicts do happen, we may apply locally rearrangement algorithms (e.g., Solovey et al. 2015) to perform the snapping step without incurring much penalty on time optimality.
With the snapping process complete, a discrete abstraction of the original continuous problem is obtained. For our example, this leads to the scenario captured in Fig. 1(c). If we are not interested in optimality, the discrete problem may be attempted using a non-optimal but polynomial time algorithm \cite{Kornhauser:1984:RGT:500094.502191, Yu:2014:ISP:2598692.2598732}. As stated in the individual subsections, the computation required in this section can be carried out using low-degree polynomial time algorithms. The relative time used for this portion is trivial as compared to the time required for solving the roadmap-based discrete problem.

IV. Fast, Near-Optimal Discrete Path Planning

After a high quality roadmap is obtained with near-optimality guarantees on time and distance (e.g., an optimality-preserving reduction from continuous space to discrete space), one may then freely choose an algorithm for finding solutions to the discrete abstraction (Fig. 1(c) in our example). For minimizing task completion time, the algorithm from Yu and LaValle \cite{Yu:2013:DSM:2495178.2495200} as a baseline, we detail heuristics that greatly improve the computational efficiency of such methods without incurring too much penalty on (time) optimality.

A. The Baseline, ILP Model-Based Algorithm

We first briefly describe how an ILP model is obtained in Yu and LaValle \cite{Yu:2013:DSM:2495178.2495200}. The essential idea is to perform time expansion over a given discrete roadmap and then build the ILP model over the resulting space-time graph. For a hexagonal roadmap, between subsequent time steps, the time expansion has an intuitive local structure illustrated in Fig. 7 which basically says that a robot at a node \( v \) and its neighbors can reach \( v \) in the next time step. Then, in the ILP constraint setup phase, two additional constraints are enforced:

1) At most a single edge leading to \( v(t+1) \) from time step \( t \) can be used (i.e., all but one such edge binary variable can be set to 1), this enforces that only a single robot may reach node \( v \) at time step \( t+1 \). This prevents collision on a node.
2) At most one edge from \((u(t), v(t+1))\) and \((v(t), u(t+1))\) can be used. This prevents collision on an edge.

\[ \begin{align*} 
&v(t) \quad \text{node at time step } t, \quad \text{its neighbors can reach } v \text{ in the next time step.} \\
&v(t+1) \quad \text{node at time step } t+1. 
\end{align*} \]

Fig. 7. In a single time expansion step, a node's neighbors (including the node itself) at time step \( t \) are connected to the node at time step \( t+1 \).

The above constraints can be encoded easily using linear inequalities. To optimize for minimum time, an underestimate \( T \) of the minimum required time is used as the number of time steps in the time expansion. This underestimate can be easily obtained by computing the minimum path length for each robot ignoring the rest of the robots and then taking the maximum of the path lengths. To complete the model setup, for a robot starting at node \( u \) that has node \( v \) as its goal, an edge \((v(T), u(0))\) is added to the model and is forced to be used. This forces a solution to find a path through the space-time graph connecting \( u(0) \) and \( v(T) \). If the model is infeasible, \( T \) is then increased and the model is rerun until a solution is found. The number of time steps required for finding the first feasible solution is then the optimal time (for the discrete problem). When the initial roadmap has good connectivity, which is the case targeted by our work, this method appears to work reasonably well for instances with 50 robots, taking only minutes to solve such problems (see Yu and LaValle \cite{Yu:2013:DSM:2495178.2495200} for details).

B. Heuristic: Divide-and-Conquer Over Time Domain

In exploring the ILP model-based algorithm, we observe the general trend that the model solution time grows exponentially with respect to the size of the model. This prevents the baseline algorithm from being very useful as it does not work very well beyond 10-20 robots when the robot density is also high, even without the presence of static obstacles.

The same observation, though limiting the performance of the (exact) baseline algorithm, turns out to offer an useful insight toward a highly efficient divide-and-conquer heuristic. We notice that by limiting the size of the ILP model, we generally get fairly good performance from an ILP solver (we used Gurobi Inc. \cite{Gurobi:2015} in this paper). To apply the method to more challenging problems (e.g., solving problems with hundreds of robots quickly), we simply limit the individual ILP model that is fed to the solver. One way to achieve this is through divide-and-conquer over the time domain. We use a simple example (see Fig. 8) to illustrate this idea.

In Fig. 8(a), we have a simple planning problem for two robots on a 3 \times 3 grid. To carry out the heuristic, we first compute a shortest path for every pair of start and goal locations. In this case, we get the orange and green paths for robots 1 and 2, respectively. Then, if we decide to split the problem into two smaller problems, for each of the paths, it is split into two

\[ A \text{ preliminary poster version discussing these heuristics appeared in Yu and LaValle 2013b.} \]
(generally) equal length pieces and the middle node is set as the intermediate goal. In our example, we may do this for robot 1 easily and set the intermediate goal location at $(1, 1)$ from the top-left corner (the brown disc labeled $1$ in Fig. 8(b)). For robot 2, because the middle location coincides with that of robot 1, we pick an alternative location that is not already occupied as the intermediate goal for robot 2, in this case $(2, 2)$ from the top-left corner. The intermediate goals for the first instance will also serve as the start locations of the second instance. This yields two child instances with both requiring a time expansion with 2 steps each, whereas the original problem also requires a time expansion with 4 steps. In general, we may divide a problem into arbitrarily many smaller instances in the time domain.

If a problem is divided in this manner to $k$ sub problems, we call the resulting heuristic a $k$-way split. Because the division is over time, there is in fact no interaction between the individual, smaller instances. Once we obtain the solution for each child instance, the solutions can be glued together by simple concatenation. In practice, it turns out that this simple heuristic dramatically improves the performance without heavy negative impact on path optimality. In computational experiments, we observe a consistent speedup.

C. Heuristic: Reachability Analysis

Another method to effectively reduce ILP model size (without losing any guarantee) is through reachability analysis. Again using the example from Fig. 8(a) and focusing on robot 1, if the time expansion uses 4 time steps, then the reachable nodes (from both the start and the goal) of the graph at $t = 1, 2, 3$ is illustrated in Fig. 9. Constructing the time-expanded graph from these then greatly reduces the resulting ILP model size.

Remark. Because the problem we are to solve in this section is NP-complete [Yu and LaValle (2013c)] and we are aiming to solve it exactly, no meaningful analysis on computational complexity can be provided; we only note that the computational time required by this part of the framework dominates all other parts.

V. COMPUTATIONAL EVALUATION

We implemented the roadmap building phase in C++ using CGAL. The discrete path planning module, written in Java, uses Gurobi (2015) as the ILP solver. The experiments were carried out on an Intel i5-2450M laptop PC. Selected runs of the program are captured in a video, available at http://youtu.be/eg3tKlV2eI0.

For evaluation, we tested of our algorithmic framework over five distinct environments. The first one is a simple square with a side length of 35 (recall that the robots are unit discs), with no internal obstacles. The rest of the environments have the same bounding square but contain different obstacle setups. We randomly select start and goal locations for all our tests. These environments, along with a typical 50-robot problem instance, are illustrated in Fig. 10.

A. Performance in Bounded, Obstacle-Free Environment

We first characterize how our framework performs in terms computation speed and solution optimality, as $k$-way split heuristic is used with different values of $k$. For this task, we carry out two sets of computations. The first set, covered in this subsection, focuses on bounded, obstacle-free environment. For this environment, we let the number of robots vary between
10 to 100 and evaluate the performance of the framework with the baseline algorithm (i.e., a single sub-problem), 2-way split (i.e., two sub-problems), 4-way split, and 8-way split. For each choice of the number of robots and the heuristic, 10 test cases are randomly generated sequentially and solved. The average running time and optimality ratio is plotted in Fig. 11. Note that our computation of the optimality ratio is conservative. To compute this ratio, we find the shortest distance between each pair of start and goal locations and use the maximum of these distances as the estimate of optimal time (since the robot has maximum speed of 1). We then obtain the optimality ratio by dividing the actual task completion time by the estimated value.

From the experiments, we observe that the baseline algorithm actually performs quite well for up to 40 robots in the absence of obstacles. With that said, both 2-way and 4-way splits do much better without losing much optimality—all three achieves optimality ratio between 1.2 to 1.6 in our experiments. With the 8-way split, sacrificing some optimality, we were able to consistently solve problems with 100 robots in 10 seconds on average. Such settings correspond to robots occupying over 25% of the free space, a setting that has never been attempted before in optimal multi-robot path planning. With 8-way split, problems with 125 robots in the same environment, which corresponds to a robot density over 31.4%, can be comfortably solved in about 15 minutes.

**B. Performance in Bounded Environment with Obstacles**

The second set of experiments shifts the focus to an environment with obstacles. For this we use the “Jack” environment. We choose this environment because it is in fact a relatively difficult setting as many shortest paths may have to pass through the middle, causing conflicts. The experimental result, for 5 to 50 robots, is plotted in Fig. 12 which is consistent with our first set of experiments. We note that obstacles, though increases computation time, do not heavily impact the optimality of the result.

**C. Evaluation of Overall Framework Performance**

Our last set of experiments is aimed at showing the overall effectiveness of our framework. For this purpose we select the splitting heuristic automatically. Roughly, we do this by increasing $k$ (in a $k$-way split) to keep each time expansion with 10 time steps, which we have found to strike a good balance between speed and optimality. For the set of environments illustrated in Fig. 10 the experimental result is plotted in Fig. 13. Note that our method is able to consistently solve all instances with an average solution time from 0.5 to 10 seconds while provide good time-optimality assurance. The two spikes in Fig. 13(a) at 40 robots are due to the switching to 8-way split at 45 robot for these environments.
VI. CONCLUSION

In this paper, we present an algorithmic framework for tackling the multi-robot path planning problem in continuous, multiply-connected environments. Our framework partitions the planning task into two phases. In the first phase, the configuration space is tiled with a carefully selected regular lattice pattern, taking into account robot-robot collision avoidance. The imposed lattice is then processed to yield a roadmap that preserves the connectivity of the continuous configuration space, which is essential for achieving near optimality in the final solution. Snapping the robots and their goal locations to the roadmap then transforms the initial continuous planning problem to a discrete planning problem. In the second phase, the discrete planning problem can be solved using any graph-based multi-robot path planning algorithms, after which the solution can be readily used in continuous domains. With an improved ILP-based solver, our planner can consistently compute large problem instances for tens to hundreds of robots in seconds to minutes.

An important issue not addressed in this paper, partly due to limited space, is path planning for non-holonomic robots. We briefly touch upon this issue here. Our algorithmic framework supports quite naturally nonholonomic robots that are small-time locally controllable (STLC) with reasonable minimum turning radius. Essentially, to apply our method to a nonholonomic robot, the robot only need the capability to: (i) move from its start location to a nearby roadmap node with a given orientation, (ii) trace any path on the roadmap without incurring collision, and (iii) move from a roadmap node to a nearby goal location (with an arbitrary orientation). A car-like robot, or any robot that is STLC, possesses the first and the third capabilities. Then, as long as the robot has a minimum turning radius of 2, it can follow any path on a hexagonal lattice without violating its nonholonomic constraints (see Fig. 14). More importantly, multiple robots may move concurrently in such a manner without causing collisions. The introduction of nonholonomic constraints does not significantly affect optimality.

Fig. 14. A car-like robot with a minimum turning radius of 2 can trace any given path on a hexagonal lattice with side length $4/\sqrt{3}$ without violating its nonholonomic constraints or colliding with other robots.
REFERENCES

J. Alonso-Mora, “Collaborative Motion Planning for Multi-Agent Systems,” Ph.D. dissertation, ETH Zurich, Mar. 2014.

J. Snape, S. J. Guy, J. van den Berg, and D. Manocha, “Smooth coordination and navigation for multiple differential-drive robots,” in Experimental Robotics. Springer, 2014, pp. 601–613.

R. A. Knepper and D. Rus, “Pedestrian-inspired sampling-based multi-robot collision avoidance,” in International Symposium on Robot and Human Interactive Communication (RO-MAN), 2012, pp. 94–100.

P. R. Wurman, R. D’Andrea, and M. Moutz, “Coordinating hundreds of cooperative, autonomous vehicles in warehouses,” AI Magazine, vol. 29, no. 1, pp. 9–19, 2008.

J. R. Snape, “Smooth and collision-free navigation for multiple mobile robots and video game characters,” Ph.D. dissertation, University of North Carolina at Chapel Hill, 2012.

E. J. Griffith and S. Akella, “Coordinating multiple droplets in planar array digital microfluidic systems,” International Journal of Robotics Research, vol. 24, no. 11, pp. 933–949, 2005.

J. Schwartz and M. Sharir, “On the piano movers’ problem: III. coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers,” International Journal of Robotics Research, vol. 2, no. 3, pp. 46–75, 1983.

S. J. Buckley, “Fast planning for multiple moving robots,” in Proceedings IEEE International Conference on Robotics & Automation, 1989, pp. 322–326.

M. A. Erdmann and T. Lozano-Pérez, “On multiple moving objects,” in Proceedings IEEE International Conference on Robotics & Automation, 1986, pp. 1419–1424.

Z. Bien and J. Lee, “A minimum-time trajectory planning method for two robots,” IEEE Transactions on Robotics & Automation, vol. 8, no. 3, pp. 414–418, 1992.

P. A. O’Donnell and T. Lozano-Pérez, “Deadlock-free and collision-free coordination of two robot manipulators,” in Proceedings IEEE International Conference on Robotics & Automation, 1989, pp. 484–489.

R. Ghrist, J. M. O’Kane, and S. M. LaValle, “Computing Pareto Optimal Coordinations on Roadmaps,” International Journal of Robotics Research, vol. 24, no. 11, pp. 997–1010, 2005.

S. M. LaValle and S. A. Hutchinson, “Optimal planning for multiple robots having independent goals,” IEEE Transactions on Robotics & Automation, vol. 14, no. 6, pp. 912–925, Dec. 1998.

J. Peng and S. Akella, “Coordinating multiple robots with kinodynamic constraints along specified paths,” in Algorithmic Foundations of Robotics V. J.-D. Boissonat, J. Burdick, K. Goldberg, and S. Hutchinson, Eds. Berlin: Springer-Verlag, 2002, pp. 221–237.

J. van den Berg and M. Overmars, “Prioritized motion planning for multiple robots,” in Proceedings IEEE/RSJ International Conference on Intelligent Robots & Systems, 2005.

J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha, “Centralized path planning for multiple robots: Optimal decoupling into sequential plans,” in Robotics: Science and Systems, 2009.

P. Švestka and M. H. Overmars, “Coordinated path planning for multiple robots,” Robotics and Autonomous Systems, vol. 23, no. 3, pp. 125–152, 1998.

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’,” International Journal of Robotics Research, vol. 3, no. 4, pp. 76–88, 1984.

J. van den Berg, M. C. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in Proceedings IEEE International Conference on Robotics & Automation, 2008, pp. 1928–1935.

J. van den Berg, J. Snape, S. J. Guy, and D. Manocha, “Reciprocal collision avoidance with acceleration-velocity obstacles,” in Proceedings IEEE International Conference on Robotics & Automation, 2011, pp. 3475–3482.

K. Kant and S. Zucker, “Towards efficient trajectory planning: The path velocity decomposition,” International Journal of Robotics Research, vol. 5, no. 3, pp. 72–89, 1986.

A. F. v. d. S. I. Karamouzas, R. Geraerts, “Space-time group motion planning,” in Proceedings Workshop on Algorithmic Foundations of Robotics, 2012.

M. Peasgood, C. Clark, and J. McPhee, “A complete and scalable strategy for coordinating multiple robots within roadmaps,” IEEE Transactions on Robotics, vol. 24, no. 2, pp. 283–292, 2008.

K. Solovey, O. Salzman, and D. Halperin, “Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning,” in Proceedings Workshop on Algorithmic Foundations of Robotics, 2014.

K. Solovey, J. Yu, O. Zamir, and D. Halperin, “Motion planning for unlabeled discs with optimality guarantees,” in Robotics: Science and Systems, 2015, to appear.

M. Turpin, N. Michael, and V. Kumar, “Concurrent assignment and planning of trajectories for large teams of interchangeable robots,” in Proceedings IEEE International Conference on Robotics & Automation, 2013.

D. Kornhauser, G. Miller, and P. Spirakis, “Coordinating pebble motion on graphs, the diameter of permutation groups, and applications,” in Proceedings IEEE Symposium on Foundations of Computer Science, 1984, pp. 241–250.

R. M. Wilson, “Graph puzzles, homotopy, and the alternating group,” Journal of Combinatorial Theory (B), vol. 16, pp. 86–96, 1974.

M. R. K. Ryan, “Exploiting subgraph structure in multi-robot path planning,” Journal of Artificial Intelligence Research, vol. 31, pp. 497–542, 2008.

T. Standley and R. Korf, “Complete algorithms for cooperative pathfinding problems,” in Proceedings International Joint Conference on Artificial Intelligence, 2011, pp. 668–673.

G. Wagner and H. Choset, “M*: A complete multirobot path planning algorithm with performance bounds,” in Proceedings IEEE/RSJ International Conference on Intelligent Robots & Systems, 2011, pp. 3260–3267.

J. Yu and S. M. LaValle, “Planning optimal paths for multiple robots on graphs,” in Proceedings IEEE International Conference on Robotics & Automation, 2013, pp. 3612–3617.

A. Krontiris, Q. Sajid, and K. Bekris, “Towards using discrete multiagent pathfinding to address continuous problems,” Workshop on Multi-agent Pathfinding (in conjunction with AAAI), 2012.

K. Solovey and D. Halperin, “k-color multi-robot motion planning,” in Proceedings Workshop on Algorithmic Foundations of Robotics, 2012.

W. Robert, The Geometrical Foundation of Natural Structure. A Source Book of Design. Dover Publications, Inc. New York, 1978.

T. Lozano-Pérez and M. A. Wesley, “An algorithm for planning collision-free paths among polygonal obstacles,” Communications of the ACM, vol. 22, no. 10, pp. 560–570, 1979.

J. Yu and D. Rus, “Pebble motion on graphs with rotations: Efficient feasibility tests and planning,” in Proceedings Workshop on Algorithmic Foundations of Robotics, 2014.

J. Yu and S. M. LaValle, “Fast, near-optimal computation for multi-robot path planning on graphs,” in Proceedings AAAI National Conference on Artificial Intelligence, 2013, late breaking papers.

G. O. Inc., “Gurobi optimizer reference manual,” 2015. [Online]. Available: [http://www.gurobi.com]

J. Yu and S. M. LaValle, “Structure and intractability of optimal multi-robot path planning on graphs,” in Proceedings AAAI National Conference on Artificial Intelligence, 2013, pp. 1444–1449.

“CGAL, Computational Geometry Algorithms Library,” http://www.cgal.org.