Expected Constant-Factor Optimal Multi-Robot Path Planning in Well-Connected Environments

Jingjin Yu

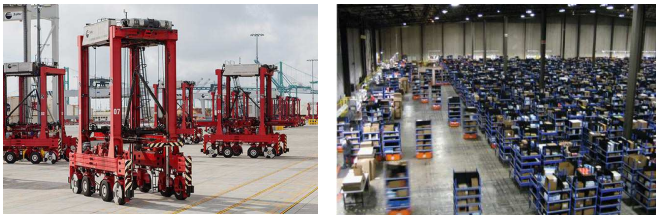
***Abstract*— Fast algorithms for optimal multi-robot path planning are sought after in both research and real-world applications. Known methods, however, generally do not simultaneously guarantee good solution optimality and fast run time for difficult instances. In this work, we develop a low-polynomial running time algorithm, called SPLITANDGROUP, that solves the multi-robot path planning problem on grids and grid-like environments, and produces constant factor time- and distance-optimal solutions, in expectation. In particular, SPLITANDGROUP computes solutions with sub-linear makespan. SPLITANDGROUP is capable of handling cases when the density of robot is extremely high - in a graph-theoretic setting, the algorithm supports cases where all vertices of the underlying graph are occupied by robots. SPLITANDGROUP attains its desirable properties through a careful combination of divide-and-conquer technique and network flow based methods for routing the robots.**

I. INTRODUCTION

Fast methods for multi-robot path planning have found many real-world applications including shipping container handling (Fig. 1(a)), order fulfillment (Fig. 1(b)), horticulture, among others, drastically improving the associated process efficiency. While commercial applications have been able to scale quite well, e.g., a single Amazon fulfillment center can operate over a thousand Kiva mobile robots, it remains unclear what level of optimality is achieved by the underlying scheduling algorithms in these applications. The same optimality-efficiency gap exists in the multi-robot research domain: known algorithms for multi-robot path planning do not simultaneous guarantee good solution optimality and fast running time. This is not entirely surprising as it is well known that optimal multi-robot path planning problems are generally NP-hard.

In this work, we narrow this optimality-efficiency gap in multi-robot path planning, focusing on a class of *grid-like well-connected* environments. Well-connected environments (to be formally defined) include the container shipping port scenario and the Amazon fulfillment center scenario. A key property of these environments is that sub-linear time-optimal solution is possible, which is not true for general environments. Using a careful combination of divide-and-conquer and network flow techniques, we show that *expected constant factor time- and distance-optimal solutions* can be computed in *low-polynomial running time* in such settings. We call the resulting algorithm SPLITANDGROUP. In other words, SPLITANDGROUP can efficiently compute $O(1)$ optimal solutions. The method readily generalizes to higher dimensions as well.

Jingjin Yu is with the Department of Computer Science, Rutgers University at New Brunswick. E-mails: jingjin.yu@cs.rutgers.edu. This work is supported by NSF awards IIS-1617744 and IIS-1734419.



(a) (b)

Fig. 1. (a) Automated straddle carriers at the port of Los Angeles. Each straddle carrier is capable of autonomously navigate to pick up or drop off a shipping container at a designated location. (b) Amazon's Kiva multi-robot system working at its order fulfillment centers.

Related work. In multi-robot path and motion planning, the goal is for the moving bodies, e.g., robots or vehicles, to reach their respective destinations, collision-free. Frequently, certain optimality measure (e.g., time, distance, communication) is also imposed. Variations of the multi-robot path and motion planning problem have been actively studied for decades [1]–[21]. As a fundamental problem, it finds applications in a diverse array of areas including assembly [22], [23], evacuation [24], formation [25]–[29], localization [30], micro droplet manipulation [31], [32], object transportation [33], [34], and search-rescue [35]. In industrial applications pertinent to the current work, *centralized* planners are generally employed to enforce global control to drive operational efficiency. This work also follows the same paradigm.

Similar to high dimensional single robot problems [36], [37], multi-robot path planning is well known to be strongly NP-hard for discs in simple polygons [38] and PSPACE-hard just for translating rectangles [39]. The hardness of the problem extends to unlabeled case [40] where it remains highly intractable [41], [42]. Nevertheless, under mild assumptions, the unlabeled case can be solved optimally or near optimally with guaranteed running time [13], [43]–[46].

Because general (labeled) optimal multi-robot path planning problems in continuous domains is extremely challenging, a common approach is to start with a discrete setting from the onset. Significant progress has been made on optimally solving the general (labeled) problem in discrete settings, in particular on grid-based environments. Multi-robot motion planning is less computationally expensive in discrete domains, with the feasibility problem readily solvable in $O(|V|^3)$ time, in which $|V|$ is the number of vertices of the discrete graph where the robots may reside [47]–[49]. Optimal versions remain computationally intractable under

the graph-theoretic setting [50]–[53], but the complexity has dropped from PSPACE-hard to NP-complete in many cases. Decoupling-based heuristics prove to be useful [54]–[56], allowing the effective minimization of certain accrued cost [9], [57]–[63]. Beyond decoupling, other ideas have also been explored, including casting the problem as other known NP-hard problems [64]–[66] for which high-performance solvers are available.

Contributions. The main contribution brought forth by this work is a low-polynomial time deterministic algorithm, called SPLITANDGROUP, for solving the optimal multi-robot path planning problem in grids and grid-like well-connected environments. Under the prescribed settings, SPLITANDGROUP computes a solution with sub-linear makespan. Moreover, the solution is only a constant multiple of the true optimal solution in terms of both makespan and total distance, in expectation. SPLITANDGROUP applies to settings with extreme robot density. To the best of our knowledge, SPLITANDGROUP is the first such algorithm that achieves the combination of desirable properties.

Organization. The rest of the paper is organized as follows. In Sec. II, the discrete multi-robot path planning problem is formally defined, followed by analysis on connectivity for achieving good solution optimality. This leads us to the choice of grid-like environments. We describe the details of the main algorithm, SPLITANDGROUP, in Sec. III. In Sec. IV, complexity and optimality properties of SPLITANDGROUP are established. In Sec. V, we show that SPLITANDGROUP generalizes to higher dimensions and (grid-like) well-connected environments including certain continuous ones. Due to space limit, some proofs are omitted; the can be found in [67].

II. PRELIMINARIES

In this section, we state the multi-robot path planning problem and two important associated optimality objectives, under a graph-theoretic setting. Next, we establish that working with arbitrary graphs may lead to rather sub-optimal solutions (i.e., super-linear with respect to the number of robots). This necessitates the restriction of the graphs if desirable optimality results are to be achieved.

A. Graph-Theoretic Optimal Multi-Robot Path Planning

Let $G = (V, E)$ be a simple, undirected, and connected graph. On this graph G, a set R of labeled robots may move synchronously in a collision-free manner. At (integer) *time steps* starting from $t = 0$, each robot resides on a unique vertex of G, inducing a *configuration* X of the robots. Effectively, X is an injective map $X : R \to V$ specifying which robot occupies which vertex (see Fig. 2). From time step t to time step $t + 1$, a robot may *move* from its current vertex to an adjacent one under two (collision avoidance) conditions: *(i)* the new configuration at $t + 1$ remains an injective map, i.e., each robot occupies a single vertex, and *(ii)* no two robots travel along the same edge in opposite directions.

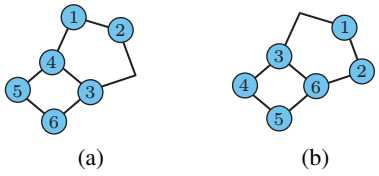


Fig. 2. Graph-theoretic formulation of the multi-robot path planning problem. (a) A configuration of six robots on a graph (roadmap) with seven vertices. (b) A configuration that is reachable from (a) in a single synchronous move.

A multi-robot path planning problem (MPP) is fully defined by a 3-tuple (G, X_I, X_G) in which G is a graph, and X_I and X_G are two configurations. In this work, we look at the *extreme* case of $|X_I| = |X_G| = |V|$. That is, all vertices of G are occupied. We are interested in two optimal MPP formulations. In what follows, *makespan* is the time span covering the start to the end of a task and all edges of G are assumed to have a length of 1 so that a robot traveling at unit speed can cross it in a single time step.

Problem 1 (Minimum Makespan (TMPP)): Given G, X_I, and X_G, compute a sequence of moves that takes X_I to X_G while minimizing the makespan.

Problem 2 (Minimum Total Distance (DMPP)): Given G, X_I, and X_G, compute a sequence of moves that takes X_I to X_G while minimizing the total distance traveled.

These two problems are known to be NP-hard and cannot always be solved simultaneously [52], [53]. In this paper, we assume that X_I and X_G are randomly distributed.

B. Effects of Environment Connectivity

The well-known *pebble motion* problems, which are highly similar to MPP, may require $\Omega(|V|^3)$ individual moves to solve [68]. Since each pebble (robot) may only move once per step, at most $|V|$ individual moves can happen in a time step. This implies that pebble motion problems, even with synchronous moves, can have an optimal makespan of $\Omega(|V|^2)$, which is super linear (i.e. $\omega(|V|)$). The same is true for TMPP under certain graph topologies. We first prove a simple but useful lemma for a class of graphs we call *figure-8* graphs. In such a graph, there are $|V| = 7n + 6$ vertices for some integer $n \geq 0$. The graph is formed by three disjoint paths of lengths $n, 3n+2$, and $3n+2$, meeting at two common end vertices. Figure-8 graphs with $n = 1$ are illustrated in Fig. 3.

An interesting and very useful property of figure-8 graphs is that an arbitrary MPP instance on such a graph is feasible.

Lemma 1: An arbitrary MPP instance (G, X_I, X_G) is feasible when G is a figure-8 graph.

Proof: Using the three-step plan provided in Fig. 3, we may exchange the locations of robots 1 and 2 without collision. This three-step plan is scale invariant and applies to any n. With the three-step plan, the locations of any two adjacent robots (e.g., robots 4 and 5 in the top left figure of Fig. 3) can be exchanged. To do so, we may first rotate the two adjacent robots of interest to

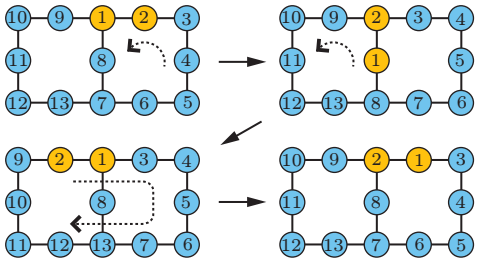


Fig. 3. A three-step plan for exchanging the locations of robots 1 and 2 on a figure-8 graph with $7n + 6$ vertices ($n = 1$ in this case).

the locations of robots 1 and 2, do the exchange using the three-step plan, and then reverse the initial rotation. Let us denote such a sequence of moves as a *2-switch*. Because the exchange of any two robots on the figure-8 graph can be decomposed into a sequence of 2-switches, such exchanges are always feasible. As an example, the exchange of robots 4 and 9 can be carried out using a 2-switch sequence $\langle(3,4),(2,4),(1,4),(4,9),(1,9),(2,9),(3,9)\rangle$, of which each individual pair is an adjacent one after the previous 2-switch is completed. Because solving the MPP instance (G, X_I, X_G) can be decomposed into a sequence of two-robot exchanges, arbitrary MPP instances are solvable on figure-8 graphs. ■

The introduction of figure-8 graphs allows us to formally establish that sub-linear optimal solutions are not possible on an arbitrary connected graph.

Theorem 2: There exists an infinite family of TMPP instances on figure-8 graphs with $\omega(|V|)$ minimum makespans.

Theorem 2 implies that if the classes of graphs are not restricted, we cannot always hope for the existence of solutions with linear (and therefore sub-linear) makespans with respect to the number of vertices of the graphs. That is,

Corollary 3: TMPP does not admit solutions with linear or sub-linear makespans on an arbitrary graph.

Corollary 3 suggests that seeking general algorithms for providing linear or sub-linear makespans that apply to all environments will be a fruitless attempt. On the other hand, sub-linear makespans are highly desirable in practice. With this in mind, we first focus our attention on a restricted but very practical class of discrete environments: grid graphs.

III. Routing Robots on Rectangular Grids with a Sub-Linear Makespan

We begin the analysis with rectangular grids. Assuming unit edge lengths, such a grid is fully specified by two integers m_ℓ and m_s, representing the number of vertices on the long and short sides of the rectangular grid, respectively. Without loss of generality, assume that $m_\ell \geq m_s$ (see Fig. 4 for a 8×4 grid). We further assume that $m_\ell \geq 3$ and $m_s \geq 2$ since a 2×2 grid does not admit non-trivial solutions. These assumptions on grid dimensions are implicitly assumed in this paper whenever "$m_\ell \times m_s$ grid" is used. The main result to be established in this section is the following.

Theorem 4: Let (G, X_I, X_G) be an arbitrary TMPP instance in which G is an $m_\ell \times m_s$ grid. The instance admits a solution with a makespan of $O(m_\ell)$.

Note that the $O(m_\ell)$ bound is *sub-linear* with respect to the number of robots, which is $\Omega(m_s m_\ell)$ and $\Omega(m_\ell^2)$ for square grids. We name the algorithm, to be constructed, as SplitAndGroup and explain how the divide-and-conquer algorithm works at a high level. In this section we focus on the makespan optimality aspects of SplitAndGroup. The establishment of polynomial-time complexity and additional properties of the algorithm is delayed until Sec. IV.

To simplify the explanation, assume that $m_\ell = m_s = 2^k$ for some integer k. In the first iteration of SplitAndGroup, it *splits* the grid into two smaller rectangular grids, G_1 and G_2, of size $2^k \times 2^{k-1}$ each. Then, robots are moved so that at the end of the iteration, if a robot belongs to G_1 (resp., G_2) in X_G, it should be on some arbitrary vertex of G_1 (resp., G_2). This is the *grouping* operation. An example of a single SplitAndGroup iteration is shown in Fig. 4. We will show that such an iteration can be completed in $O(m_\ell) = O(2^k)$ time steps (makespan). In the second iteration, the same process is carried out on both G_1 and G_2 in parallel, which again requires $O(2^k)$ time steps. In the third iteration, we start with four $2^{k-1} \times 2^{k-1}$ grids and the iteration can be completed in $O(2^{k-1})$ time steps. After $2k$ iterations, the problem is solved with a desired makespan of

$$2O(2^k) + 2O(2^{k-1}) + \ldots + 2O(1) = O(2^k) = O(m_\ell).$$

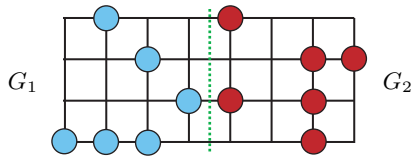


Fig. 4. Illustration of a single iteration of SplitAndGroup on a 8×4 grid. Note that the grid is fully populated with robots and some are not drawn. The overall grid is split in the middle by the dotted line to give two 4×4 grids G_1 and G_2. The robots shown on G_1 (resp., G_2) have goal locations on G_2 (resp., G_1). In the grouping operation, these robots must move across the split line after the grouping operation is complete. Other robots (not shown) on the grid must remain where they are after the grouping operation is complete. In the next iteration, the same procedure is applied to G_1 and G_2 in parallel.

We now proceed to describe the SplitAndGroup algorithm in more detail. To achieve the stated $O(m_\ell)$ makespan, SplitAndGroup needs to enable as much concurrent robot movement as possible. This is rather challenging because of our worst case assumption that there are as many robots as the number of vertices. This is where the grid graph assumption becomes critical: it enables the concurrent "flipping" or "bubbling" of robots. Let $G = (V, E)$ be an $m_\ell \times m_s$ grid graph whose vertices are fully occupied by robots. Let $E' \subset E$ be a set of vertex disjoint edges of G. Suppose for each edge $e = (v_1, v_2) \in E'$, we wish to exchange the two robots on v_1 and v_2 without collision. Let us call this

operation FLIP(E'). Then, the following holds.

Lemma 5: Let $G = (V, E)$ be an $m_\ell \times m_s$ grid. Let $E' \subset E$ be a set of vertex disjoint edges. Then FLIP(E') can be completed in a constant number of time steps.

Lemma 5, in a nutshell, allows the exchange of two adjacent robots to be performed in $O(1)$ time steps. Moreover, it allows such exchanges to happen in parallel on disjoint edges. With Lemma 5, to prove Theorem 4, we are left to show that on an $m_\ell \times m_s$ grid, after the split operation the grouping operation in the first SPLITANDGROUP iteration can be decomposed into $O(m_\ell)$ FLIP($\cdot$) operations. Because each FLIP($\cdot$) can be carried out in $O(1)$ time steps, the overall makespan cost of the grouping operation is then $O(m_\ell)$. To obtain the desired decomposition, we need to maximize parallelism along the split line used for the split operation. We achieve the desired parallelism by partitioning the grid into trees with limited overlap. Each such tree has a limited diameter and crosses the split line. The grouping operation will then be carried out on these trees. As an example, Fig. 5 illustrates such a tree and the two groups of robots to be exchanged. We want to show that that grouping robots on trees can be done efficiently. Note that we do not require a robot in the group to go to a specific goal vertex; we do not distinguish robots within the group. In what follows, by *non-path grid*, we mean a grid that is not a path. A robot does not have *net movement* if it start and goal locations coincide.

Theorem 6: Let T be a tree of diameter d embedded in a non-path grid whose vertices are fully occupied by robots. Let P be a length ℓ path branch of T. Then, a group of robots on P can be exchanged with robots on T outside P in $O(d)$ time steps without net movement of other robots. The relocation may be performed using FLIP($\cdot$) on T.

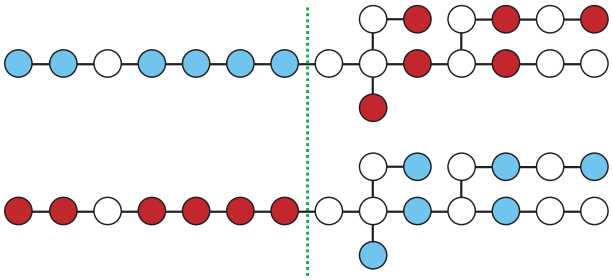


Fig. 5. Illustration of a tree-based subproblem in the grouping operation. The first picture illustrates the initial configuration and the second picture the goal configuration. The unshaded robots must retain their locations.

Implicitly, Theorem 6 says that the same $O(d)$ makespan can be achieved for multiple disjoint trees embedded in the same grid graph. The $O(d)$ makespan in fact continues to hold even when the trees have some minor overlaps. We proceed to sketch the proof of Theorem 4.

Proof: (Sketch of proof of Theorem 4) We sketch how to carry out a single iteration of SPLITANDGROUP with the help of an example. A split is always done along a longer side of the current grid. The split of a 9×7 grid G into smaller grids G_1 and G_2 is illustrated in Fig. 6(a). To move all the (red) robots to G_2, we hope to find routing paths for each red robot in G_1 to form a tree (e.g., Fig. 6(b)), after which Theorem 6 can be applied.

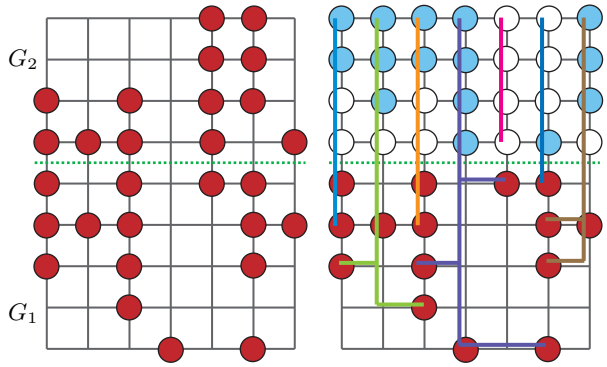


Fig. 6. (a) A 9×7 grid is split into two grids G_1 and G_2 of sizes 4×7 and 5×7, respectively. The dark-shaded robots' final goals are in G_2. (b) The grid is partitioned into (possibly non-disjoint) trees to allow the dark-shaded robots that are not already in G_2 to exchange with robots (lightly-shaded ones) that should be moved to G_1.

Toward this, for each robot in G_1 that needs to be moved, we compute paths from it to all possible targets in G_2, with the limitation that each such path has a single turn that must happen in G_1. We can then apply the Hungarian algorithm [69] to compute an initial path set with minimum total path lengths. Merging these paths then produces a set of up to m_s trees. The issue with these initial trees is that there might be *crossovers*. We define a crossover as an intersection between two paths, e.g., the dotted paths in Fig. 7(a). We do not consider the scenario in Fig. 7(b) as crossovers. Note that the scenario in Fig. 7(c) cannot happen due to the path set having the minimum total path lengths. That is, since updating these two paths Fig. 7(c) will reduce total path length, the original paths cannot be part of a minimum total distance solution.

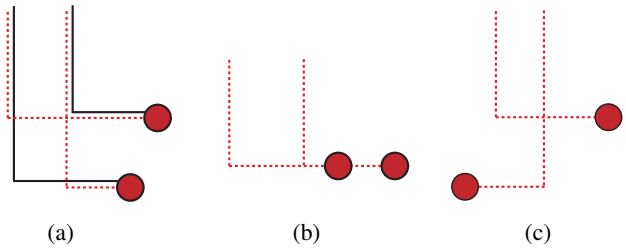


Fig. 7. (a) Example of a tree crossover (dotted paths) and its removal (solid paths) without increasing the total distance. Note that only the relevant paths of the two trees are shown. (b) An intersection that is not considered a crossover. (c) An impossible crossover scenario.

If a crossover is detected, it can be removed with simple updates (e.g., the solid paths in Fig. 7(a)) and the process will end a finite number of iterations. After all crossovers are resolved, Theorem 6 then applies to group the robots as desired. ■

IV. Complexity and Solution Optimality Properties of the SplitAndGroup Algorithm

In this section, we establish two key properties of SplitAndGroup, namely, its polynomial running-time and asymptotic solution optimality.

A. Time Complexity of SplitAndGroup

The SplitAndGroup algorithm is outlined in Algorithm 1. At Lines 1-2, a partition of the current grid G is made, over which initial path planning and scheduling is performed to generate the trees for grouping the robots into the proper subgraph. Then, at Line 3, crossovers are resolved. At Line 4, the final paths are scheduled, from which the robot moves can be extracted. This step also yields where each robot will end up at in the end of the iteration, which becomes the initial configuration for the next iteration (if there is one). After the main iteration steps are complete, at Lines 5-10, the algorithm recursively calls itself on smaller problem instances. The special case here is when a sub-problem is small enough (Line 7), in which case it is directly solved without further splitting operations.

We now proceed to bound the running time of SplitAndGroup, with the main goal to establish it polynomial running time. With this goal in mind, here, we will not push for the best bound on running time. It is straightforward to see that the Split routine takes $O(|V|) = O(m_\ell m_s)$ time to complete because we are basically scanning the input without much additional logic. MatchAndPlanPath can be implemented using the standard Hungarian algorithm [69], which runs in $O(|V|^3)$ time.

For ResolveCrossovers, we may implement it by starting with an arbitrary robot that needs to be moved across the split line and check whether the path it is on has crossovers that need to be resolved. Checking one path with another can be done in constant time because each path can be represented using a constant number of parameters even though it may have length $O(|V|)$; detecting a crossover then takes up to $O(|V|)$ running time because there are at $O(|V|)$ other paths to check against. Resolving a crossover can be completed in $O(|V|)$ running time as well. We note that, as a crossover is resolved, one of the two paths will end up being shorter (see, e.g., Fig. 7). We then repeat the process with this shorter path until no more crossover exists. Naively, because the path keeps getting shorter, this process will end in at most $O(|V|)$ steps, taking a total of $O(|V|^2)$ running time. Therefore, all together, ResolveCrossovers can be completed in $O(|V|^3)$ running time.

The ScheduleMoves routine simply extracts information from the already planned path set P and can be completed in $O(|V|)$ running time. The Solve routine takes constant time. Adding everything up, an iteration of SplitAndGroup can be carried out in $O(|V|^3)$ time using a naive implementation. Summing over all iterations, the total running time is

$$O(|V|^3) + 2O((\frac{|V|}{2})^3) + 4O((\frac{|V|}{4})^3) + \ldots = O(|V|^3),$$

which is low-polynomial with respect to the input size.

Algorithm 1: SplitAndGroup (G, X_I, X_G)

Input : $G = (V, E)$: an $m_\ell \times m_s$ grid graph
X_I: initial configuration
X_G: goal configuration
Output: $M = \langle M_1, M_2, \ldots \rangle$: a sequence of *moves*

```
%Run matching and construct initial
 trees
```
1 $(G_1, G_2) \leftarrow$ Split(G)
2 $P \leftarrow$ MatchAndPlanPath(G, X_I, X_G)

```
%Remove crossovers
```
3 $P \leftarrow$ ResolveCrossovers(P)

```
%Schedule the sequence of moves
```
4 $(M, X_I') \leftarrow$ ScheduleMoves(P')

```
%Recursively solve smaller sub-problems
```
5 **foreach** $G_i, i = 1, 2$ **do**
6 **if** $row(G_i) \leq 3$ *and* $col(G_i) \leq 3$ **then**
7 $M = M +$ Solve($G_i, X_I'|_{G_i}, X_G|_{G_i}$)
8 **else**
9 $M =$ $M +$ SplitAndGroup($G_i, X_I'|_{G_i}, X_G|_{G_i}$)
10 **end**
11 **end**
12 **return** M

B. Constant Factor Time- and Distance-Optimality

Having established that SplitAndGroup is a polynomial time algorithm that solves MPP with sub-linear makespan, we now show that a solution produced by SplitAndGroup is in fact only a constant factor away from the best possible, in expectation, for both TMPP and DMPP. That is, SplitAndGroup in fact computes an asymptotically optimal solution simultaneously for time- and distance-based objectives.

Theorem 7: Let (G, X_I, X_G) be an MPP instance in which G is an $m_\ell \times m_s$ grid, and X_I and X_G are selected uniformly randomly. Then SplitAndGroup computes constant factor optimal solutions, in expectation, with respect to the makespan and the total distance objectives.

Proof: Let i be a specific robot with $s_i \in X_I$ and $g_i \in X_G$ be its start and goal locations on G, respectively. Since X_I and X_G are randomly distributed, the expected distance between s_i and g_i on G can be readily seen to be $\Omega(m_\ell)$. This implies that the minimum possible makespan is $\Omega(m_\ell)$. Since SplitAndGroup produces solutions with an $O(m_\ell)$ makespan, it computes a constant factor approximation of the optimal makespan.

Similarly, each robot incurs an expected travel distance of $\Omega(m_\ell)$; therefore, the minimum total distance for all robots, in expectation, is $\Omega(m_s m_\ell^2)$ because there are $m_s m_l$ robots. On the other hand, because SplitAndGroup produces a solution with an $O(m_\ell)$ makespan, each robot can only travel a distance of $O(m_\ell)$. Summing this over all robots,

the solution from SPLITANDGROUP has a total distance of $O(m_s m_\ell^2)$. This matches the lower bound $\Omega(m_s m_\ell^2)$. ■

V. GENERALIZATIONS

In this section, we establish that SPLITANDGROUP readily generalizes to environments other than 2D rectangular grids, including high dimensional grids and certain continuous environments.

A. High Dimensions

SPLITANDGROUP can be readily extended to work for grids of arbitrary dimensions. For dimensions $d \geq 2$, two updates to SPLITANDGROUP are needed. First, the split line should be updated to a split hyperplane of dimension $d-1$. Second, the crossover check now takes $O(d)$ time instead of $O(1)$ time because each extra dimension may require the path to turn one more time; two paths can potentially intersect d times. Other than these changes, the rest of SPLITANDGROUP continue to work without major change. The updated SPLITANDGROUP algorithm for dimension d therefore runs in $O(d|V|^3)$ time.

B. Well-Connected Environments

The selection of G as a grid plays a critical role in proving the desirable properties of SPLITANDGROUP. In particular, two features of grid graphs are used. First, grids are composed of small cycles, which allow the 2-switch operation to be carried out locally. This in turn allows multiple 2-switch operations to be carried out in parallel. Second, restricting to two adjacent rows (or columns) of a rectangular grid (e.g., row 4 and row 5 in Fig. 6(a)), multiple 2-switches can be completed between these two rows in constant number of steps, allowing more parallelism in routing the robots. As long as the environment possesses these two features, SPLITANDGROUP works. We call such environments *well-connected*.

More precisely, a well-connected environment, $\mathcal{E}$, is one with the following properties. Let G be an $m_\ell \times m_s$ rectangular grid that contains $\mathcal{E}$. Unlike earlier grids, here, G is not required to have unit edge lengths; a cell of G is only required to be of rectangular shape with $O(1)$ side lengths. Let r_1 and r_2 be two arbitrary adjacent rows of G, and let $c_1 \in r_1$, $c_2 \in r_2$ be two neighboring cells (see, e.g., Fig. 8). The only requirement over $\mathcal{E}$ is that a robot in c_1 and a robot in c_2 may exchange residing cells locally, without affecting the configuration of other robots. In terms of the example in Fig. 8, the two shaded robots (other robots are not drawn) must be able to exchange locations in constant makespan within a region of constant radius. The requirement then implies that parallel exchanges of robots between r_1 and r_2 can be performed with a constant makespan. The same requirement applies to two adjacent columns of G. Subsequently, given an arbitrary well-connected environment $\mathcal{E}$ and an initial robot configuration X_I, the steps from SPLITANDGROUP can be readily applied to reach an arbitrary X_G that is a permutation of X_I. As long as pairwise robot exchanges can be computed efficiently,

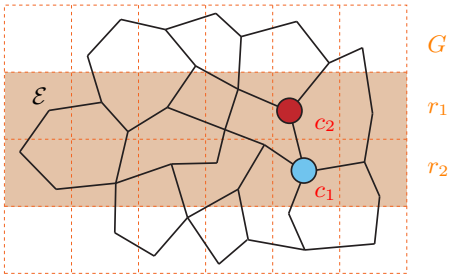


Fig. 8. Illustration of a well-connected non-grid graph environment.

the overall generalized SPLITANDGROUP algorithm also runs efficiently while maintaining the optimality guarantees. We note that the definition of well-connectedness can be further generalized to certain continuous settings. Fig. 9 provides a discrete example and a continuous example of well-connected settings, which include both the environment and the robots.

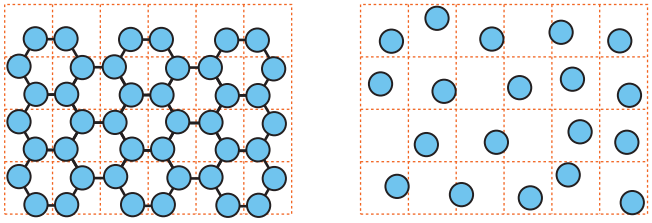


Fig. 9. Examples of two well-connected settings, with both environments and robots. An arbitrary permutation of the (labeled) robots can be reached using SPLITANDGROUP with optimality guarantees.

As mentioned in Sec. I, well-connected environments are frequently found in real-world applications, e.g., automated warehouses at Amazon and road networks in cities like Manhattan. Our theoretical results imply that such environments are in fact quite optimal in their design, in terms of being able to efficiently route robots or vehicles.

VI. CONCLUSION AND FUTURE WORK

In this work, we developed a low-polynomial time algorithm, SPLITANDGROUP, for solving the multi-robot path planning problem in grids and grid-like, well-connected environments. In expectation, the solution produced by SPLITANDGROUP is within a constant factor of the best possible in terms of makespan and total distance optimality. SPLITANDGROUP applies to problems with the maximum possible density in graph-based settings and supports certain continuous problems as well.

The development of SPLITANDGROUP opens up many possibilities for promising future work. On the theoretical side, SPLITANDGROUP gets us closer to the goal of a finding a PTAS (polynomial time approximation scheme) for optimal multi-robot path planning. Also, it would be desirable to remove the probabilistic element (i.e., the "in expectation" part) from the guarantees. On the practical side, noting that we have only looked at the case with the highest

robot density, it is promising to exploit the combination of divide-and-conquer and network flow techniques to seek more optimal algorithms for cases with lower robot density.

REFERENCES

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

[2] A. Zelinsky, "A mobile robot exploration algorithm," *IEEE Transactions on Robotics & Automation*, vol. 8, no. 6, pp. 707–717, 1992.

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

[4] Y. Guo and L. E. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," in *Proceedings IEEE International Conference on Robotics & Automation*, 2002, pp. 2612–2619.

[5] D. Silver, "Cooperative pathfinding," in *The 1st Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2005, pp. 23–28.

[6] M. R. K. Ryan, "Exploiting subgraph structure in multi-robot path planning," *Journal of Artificial Intelligence Research*, vol. 31, pp. 497–542, 2008.

[7] R. Jansen and N. Sturtevant, "A new approach to cooperative pathfinding," in *In International Conference on Autonomous Agents and Multiagent Systems*, 2008, pp. 1401–1404.

[8] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative pathfinding with completeness guarantees," in *Proceedings International Joint Conference on Artificial Intelligence*, 2011, pp. 294–300.

[9] T. Standley and R. Korf, "Complete algorithms for cooperative pathfinding problems," in *Proceedings International Joint Conference on Artificial Intelligence*, 2011, pp. 668–673.

[10] 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.

[11] K. Solovey and D. Halperin, "k-color multi-robot motion planning," in *Proceedings Workshop on Algorithmic Foundations of Robotics*, 2012.

[12] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic Foundations of Robotics X, Springer Tracts in Advanced Robotics*. Springer Berlin/Heidelberg, 2013, vol. 86, pp. 157–173.

[13] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "CAPT: Concurrent assignment and planning of trajectories for multiple robots," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

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

[15] 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.

[16] M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan, "Sampling-based planning, control and verification of hybrid systems," *IEE Proceedings Control Theory and Applications*, vol. 153, no. 5, p. 575, 2006.

[17] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[18] M. G. Earl and R. D'Andrea, "Iterative milp methods for vehicle-control problems," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1158–1167, 2005.

[19] K. E. Bekris, K. I. Tsianos, and L. E. Kavraki, "A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 3784–3790.

[20] J. Alonso-Mora, R. Knepper, R. Siegwart, and D. Rus, "Local motion planning for collaborative multi-robot manipulation of deformable objects," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5495–5502.

[21] R. A. Knepper and D. Rus, "Pedestrian-inspired sampling-based multi-robot collision avoidance," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 94–100.

[22] D. Halperin, J.-C. Latombe, and R. Wilson, "A general framework for assembly planning: The motion space approach," *Algorithmica*, vol. 26, no. 3-4, pp. 577–601, 2000.

[23] B. Nnaji, *Theory of Automatic Robot Assembly and Programming*. Chapman & Hall, 1992.

[24] S. Rodriguez and N. M. Amato, "Behavior-based evacuation planning," in *Proceedings IEEE International Conference on Robotics & Automation*, 2010, pp. 350–355.

[25] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics & Automation*, vol. 14, no. 6, pp. 926–939, 1998.

[26] S. Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proceedings IEEE International Conference on Robotics & Automation*, 2004.

[27] B. Shucker, T. Murphey, and J. K. Bennett, "Switching rules for decentralized control with simple control laws," in *American Control Conference*, Jul 2007, pp. 1485–1492.

[28] B. Smith, M. Egerstedt, and A. Howard, "Automatic generation of persistent formations for multi-agent networks under range constraints," *ACM/Springer Mobile Networks and Applications Journal*, vol. 14, no. 3, pp. 322–335, Jun. 2009.

[29] H. Tanner, G. Pappas, and V. Kumar, "Leader-to-formation stability," *IEEE Transactions on Robotics & Automation*, vol. 20, no. 3, pp. 443–455, Jun 2004.

[30] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, Jun. 2000.

[31] J. Ding, K. Chakrabarty, and R. B. Fair, "Scheduling of microfluidic operations for reconfigurable two-dimensional electrowetting arrays," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 20, no. 12, pp. 1463–1468, 2001.

[32] 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.

[33] M. J. Matarić, M. Nilsson, and K. T. Simsarian, "Cooperative multi-robot box pushing," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots & Systems*, 1995, pp. 556–561.

[34] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomous robots," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots & Systems*, 1995, pp. 235–242.

[35] J. S. Jennings, G. Whelan, and W. F. Evans, "Cooperative search and rescue with a team of mobile robots," in *Proceedings IEEE International Conference on Robotics & Automation*, 1997.

[36] J. H. Reif, "Complexity of the generalized mover's problem." DTIC Document, Tech. Rep., 1985.

[37] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.

[38] P. Spirakis and C. K. Yap, "Strong NP-hardness of moving many discs," *Information Processing Letters*, vol. 19, no. 1, pp. 55–59, 1984.

[39] 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"," *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 76–88, 1984.

[40] S. Kloder and S. Hutchinson, "Path planning for permutation-invariant multirobot formations," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 650–665, 2006.

[41] R. A. Hearn and E. D. Demaine, "PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation," *Theoretical Computer Science*, vol. 343, no. 1, pp. 72–96, 2005.

[42] K. Solovey and D. Halperin, "On the hardness of unlabeled multi-robot motion planning," in *Robotics: Science and Systems (RSS)*, 2015.

[43] J. Yu and S. M. LaValle, "Distance optimal formation control on graphs with a tight convergence time guarantee," in *Proceedings IEEE Conference on Decision & Control*, 2012, pp. 4023–4028.

[44] M. Katsev, J. Yu, and S. M. LaValle, "Efficient formation path planning on large graphs," in *Proceedings IEEE International Conference on Robotics & Automation*, 2013, pp. 3606–3611.

[45] A. Adler, M. De Berg, D. Halperin, and K. Solovey, "Efficient multi-robot motion planning for unlabeled discs in simple polygons," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 1–17.

[46] K. Solovey, J. Yu, O. Zamir, and D. Halperin, "Motion planning for unlabeled discs with optimality guarantees," in *Robotics: Science and Systems*, 2015.

[47] V. Auletta, A. Monti, M. Parente, and P. Persiano, "A linear-time algorithm for the feasbility of pebble motion on trees," *Algorithmica*, vol. 23, pp. 223–245, 1999.

[48] G. Goraly and R. Hassin, "Multi-color pebble motion on graph," *Algorithmica*, vol. 58, pp. 610–636, 2010.

[49] J. Yu, "A linear time algorithm for the feasibility of pebble motion on graphs," *arXiv:1301.2342*, 2013.

[50] O. Goldreich, "Finding the shortest move-sequence in the graph-generalized 15-puzzle is NP-hard," 1984, laboratory for Computer Science, Massachusetts Institute of Technology, Unpublished manuscript.

[51] D. Ratner and M. Warmuth, "The $(n^2 - 1)$-puzzle and related relocation problems," *Journal of Symbolic Computation*, vol. 10, pp. 111–137, 1990.

[52] 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.

[53] J. Yu, "Intractability of optimal multi-robot path planning on planar graphs," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 33–40, 2016.

[54] R. Alami, F. Robert, F. Ingrand, and S. Suzuki, "Multi-robot cooperation through incremental plan-merging," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3. IEEE, 1995, pp. 2573–2579.

[55] S. Qutub, R. Alami, and F. Ingrand, "How to solve deadlock situations within the plan-merging paradigm for multi-robot cooperation," in *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1997, pp. 1610–1615.

[56] M. Saha and P. Isto, "Multi-robot motion planning by incremental coordination," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 5960–5963.

[57] G. Sharon, R. Stern, A. Felner, and N. Sturtevant, "Conflict-Based Search for Optimal Multi-Agent Path Finding," in *Proc of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[58] 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.

[59] C. Ferner, G. Wagner, and H. Choset, "Odrm* optimal multirobot path planning in low dimensional search spaces," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3854–3859.

[60] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 195, pp. 470–495, 2013.

[61] E. Boyarski, A. Felner, R. Stern, G. Sharon, O. Betzalel, D. Tolpin, and E. Shimony, "Icbs: The improved conflict-based search algorithm for multi-agent pathfinding," in *Eighth Annual Symposium on Combinatorial Search*, 2015.

[62] W. Hönig, T. S. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, "Multi-agent path finding with kinematic constraints." in *ICAPS*, 2016, pp. 477–485.

[63] L. Cohen, T. Uras, T. Kumar, H. Xu, N. Ayanian, and S. Koenig, "Improved bounded-suboptimal multi-agent path finding solvers," in *International Joint Conference on Artificial Intelligence*, 2016.

[64] P. Surynek, "Towards optimal cooperative path planning in hard setups through satisfiability solving," in *Proceedings 12th Pacific Rim International Conference on Artificial Intelligence*, 2012.

[65] E. Erdem, D. G. Kisa, U. Öztok, and P. Schueller, "A general formal framework for pathfinding problems with multiple agents." in *AAAI*, 2013.

[66] J. Yu and S. M. LaValle, "Optimal multi-robot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[67] J. Yu, "Constant factor optimal multi-robot path planning in well-connected environments," *arXiv:1706.07255*, 2017, available at: https://arxiv.org/abs/1706.07255.

[68] D. M. Kornhauser, "Coordinating pebble motion on graphs, the diameter of permutation groups, and applications," Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[69] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.