Noname manuscript No.
(will be inserted by the editor)

Average Case Constant Factor Time and Distance Optimal Multi-Robot Path Planning in Well-Connected Environments

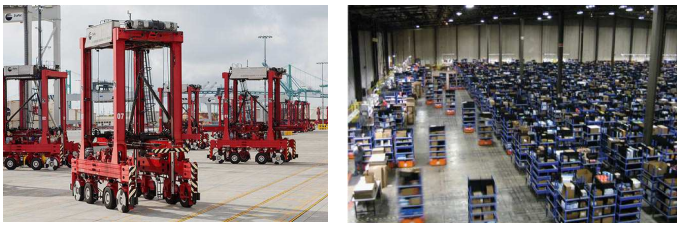
Jingjin Yu

Received: date / Accepted: date

Abstract Fast algorithms for optimal multi-robot path planning are sought after in real-world applications. Known methods, however, generally do not simultaneously guarantee good solution optimality and good (e.g., polynomial) running time. In this work, we develop a first low-polynomial running time algorithm, called SplitAndGroup (SAG), that solves the multi-robot path planning problem on grids and grid-like environments, and produces constant factor makespan optimal solutions on average over all problem instances. That is, SAG is an average case $O(1)$-approximation algorithm and computes solutions with sub-linear makespan. SAG is capable of handling cases when the density of robots is extremely high - in a graph-theoretic setting, the algorithm supports cases where all vertices of the underlying graph are occupied. SAG attains its desirable properties through a careful combination of a novel divide-and-conquer technique, which we denote as *global decoupling*, and network flow based methods for routing the robots. Solutions from SAG, in a weaker sense, are also a constant factor approximation on total distance optimality.

1 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 thousands of Kiva mobile robots, it remains unclear what level of optimality is achieved by the underlying planning and scheduling algorithms in these applications. As such, there remains the opportunity of uncovering structural insights and novel algorithmic solutions that substantially improve the throughput of current production systems that contain some multi-robot (or more generally, multi-body, where a single body may be actuated using some external mechanism) sub-systems.



(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.

The disconnection that exists in applications regarding multi-robot routing may be more formally characterized as an *optimality-efficiency gap* that has been outstanding in the multi-robot research domain for quite some time: 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 Goldreich (2011); Yu (2016); Banfi et al (2017). Nevertheless, whereas these negative results suggest that finding polynomial-time algorithms that compute exact optimal solutions for multi-robot path planning problems is impossible, they do not preclude the existence of polynomial-time algorithms that compute approximately optimal solutions.

Motivated by both practical relevance and theoretical significance, in this work, we narrow this optimality-efficiency gap in multi-robot path planning, focusing on a

Jingjin Yu
Computer Science, Rutgers University at New Brunswick, E-mail: jingjin.yu@rutgers.edu

class of *grid-like, well-connected* environments. Here, 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 *constant factor makespan optimal* solutions can be computed in *low-polynomial running time* in the *average case*, where the average is computed over all possible problem instances for an arbitrary fixed environment. We call the resulting algorithm SplitAndGroup (SAG). In other words, SAG can efficiently compute $O(1)$-approximate solutions on average. The current paper is devoted to establishing the construction, correctness, and key properties of SAG; we refer readers to Han et al (2018) for implementations and performance characteristics of SAG, where these issues are examined in detail.

Intuitively, when the density of the robots are high in a given environment, computing solutions for optimally routing these robots will be more difficult. With this line of research, our ultimate goal is to achieve a fine-grained structural understanding of the multi-robot path planning problem that allows the design of algorithms to gracefully balance between robot density and computational efficiency. As we know, when the density of robots are low, planning can be rather trivial: paths may be planned for individual robots first and because conflicts are rare, they can be resolved on the fly. In the current work, we attack the other end of the spectrum: we focus on the case of having a robot occupy each vertex of the underlying discrete graph, i.e., we work with the case of highest possible density under the given formulation. Beside the obvious theoretical challenge that is involved, we believe the study benefits algorithm design for lower density cases. Regarding this, a particularly interesting tool developed in this work is the *global decoupling* technique that enables the SAG algorithm.

Contributions. The main contribution brought forth by this work is a first low-polynomial time, deterministic algorithm, SAG, for solving the optimal multi-robot path planning problem on grids and grid-like, well-connected environments. Under the prescribed settings, SAG computes a solution with sub-linear makespan. Moreover, the solution is only a constant multiple of the optimal solution on average. In a weaker sense, SAG also computes solutions with total distance a constant multiple of the optimal for a typical instance on average. The results presented in this work expand over a conference publication Yu (2017). Most notably, this paper *(i)* provides a fuller account of the motivation and relevance that underlie the work, covering both practical and theoretical aspects, and *(ii)* includes complete proofs for all theorems; many of these proofs are much improved versions that are more clear than what appeared (as sketches) in Yu (2017).

Organization. The rest of the paper is organized as follows. Related works are discussed in Sec. 2. In Sec. 3, 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 SAG in Sec. 4. In Sec. 5, complexity and optimality properties of SAG are established. In Sec. 6, we show that SAG generalizes to higher dimensions and (grid-like) well-connected environments including continuous ones.

2 Related Work

In multi-robot path and motion planning, the main 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 Erdmann and Lozano-Pérez (1986); LaValle and Hutchinson (1998); Guo and Parker (2002); Silver (2005); Ryan (2008); Jansen and Sturtevant (2008); Luna and Bekris (2011); Standley and Korf (2011); van den Berg et al (2009); Solovey and Halperin (2012); Yu and LaValle (2013a); Turpin et al (2014); Choset et al (2005); van den Berg et al (2008); Branicky et al (2006); Khatib (1986); Earl and D'Andrea (2005); Bekris et al (2007); Knepper and Rus (2012); Alonso-Mora et al (2015). As a fundamental problem, it finds applications in a diverse array of areas including assembly Halperin et al (2000); Nnaji (1992), evacuation Rodriguez and Amato (2010), formation Balch and Arkin (1998); Poduri and Sukhatme (2004); Shucker et al (2007); Smith et al (2009); Tanner et al (2004), localization Fox et al (2000), micro droplet manipulation Griffith and Akella (2005), object transportation Matarić et al (1995); Rus et al (1995), and search-rescue Jennings et al (1997). In industrial applications pertinent to the current work, *centralized* planners are generally employed to enforce global control to drive operational efficiency. The algorithm proposed in this work also follows this paradigm.

Similar to single robot problems involving potentially many degrees of freedom Reif (1985); Canny (1988), multi-robot path planning is strongly NP-hard even for discs in simple polygons Spirakis and Yap (1984) and PSPACE-hard for translating rectangles Hopcroft et al (1984). The hardness of the problem extends to unlabeled case Kloder and Hutchinson (2006) where it remains highly intractable Hearn and Demaine (2005); Solovey and Halperin (2015). Nevertheless, under appropriate settings, the unlabeled case can be solved near optimally Katsev et al (2013); Turpin et al (2014); Adler et al (2015); Solovey et al (2015).

Because general (labeled) optimal multi-robot path planning problems in continuous domains are extremely chal-

lenging, a common approach is to start with a discrete setting from the onset. Significant progress has been made on solving the problem optimally 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 Auletta et al (1999); Goraly and Hassin (2010); Yu (2013); Yu and Rus (2015). In particular, Yu and Rus (2015) shows that the setting considered in this paper is always feasible except when the grid graph has only four vertices (which is a trivial case that can be safely ignored).

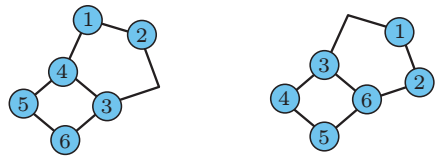
Optimal versions of the problem remain computationally intractable in a graph-theoretic setting Goldreich (2011); Ratner and Warmuth (1990); Yu and LaValle (2013b); Yu (2016); Banfi et al (2017), but the complexity has dropped from PSPACE-hard to NP-complete in many cases. This has allowed the application of the intuitive decoupling-based heuristics Alami et al (1995); Qutub et al (1997); Saha and Isto (2006) to address several different costs. In Standley and Korf (2011), individual paths are planned first. Then, interacting paths are grouped together for which collision-free paths are scheduled using Operator Decomposition (OD). The resulting algorithm can also be made complete (i.e., an anytime algorithm). Sub-dimensional expansion techniques (M*) were used in Wagner and Choset (2011); Ferner et al (2013) that actively restrict the search domain for groups of robots. Conflict Based Search (CBS) Sharon et al (2012, 2013); Boyarski et al (2015) maintains a constraint tree (CT) for facilitating its search to resolve potential conflicts. With Cohen et al (2016), efficient algorithms are supplied that compute solutions with bounded optimality guarantees. Robots with kinematic constraints are dealt with in Hönig et al (2016). Beyond decoupling, other ideas have also been explored, including casting the problem as other known NP-hard problems Surynek (2012); Erdem et al (2013); Yu and LaValle (2016) for which high-performance solvers are available. More recently, robustness, longer horizon, and other related issues have been studied in detail Ma et al (2017); Atzmon et al (2018).

3 Preliminaries

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

3.1 Graph-Theoretic Optimal Multi-Robot Path Planning

Let $G = (V, E)$ be a simple, undirected, and connected graph. A set of N labeled robots may move synchronously in a collision-free manner on G. 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 $X : \{1, \ldots, N\} \to V$ specifying which robot occupies which vertex (see Fig. 2). From step t to 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 unique vertex, and *(ii)* no two robots may travel along the same edge in opposite directions.



(a) (b)

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 X_I and X_G are two configurations. In this work, we look at the most constraining 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. 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 NP-hard and cannot always be solved simultaneously Yu and LaValle (2013b).

3.2 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 Kornhauser (1984). Since each pebble (robot) may only move once per step, at most $|V|$ individual moves can

happen in a 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 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 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* (more formally known as a *transposition* in group theory). 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 consists of two adjacent robots after the previous 2-switch is completed. Because solving the MPP instance (G, X_I, X_G) can be always decomposed into a sequence of two-robot exchanges, arbitrary MPP instances are solvable on figure-8 graphs. □

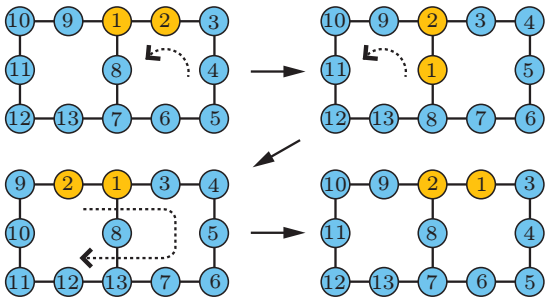


Fig. 3 A three-step plan for exchanging robots 1 and 2 on a figure-8 graph with $7n + 6$ vertices ($n = 1$ in this case).

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

Proposition 1 *There exists an infinite family of TMPP instances on figure-8 graphs with $\omega(|V|)$ minimum makespan.*

Proof We will establish the claim on the family of figure-8 graphs. By Lemma 1, there exists a sequence of moves that takes arbitrary configuration X_I to arbitrary configuration X_G. For a figure-8 graph with $|V|$ vertices, there are $|V|!$ possible configurations. Starting from an arbitrary configuration X_I, let us build a tree of adjacent configurations (two configurations are adjacent if a single move changes one configuration to the other) with X_I as the root and estimate its height h_T, which bounds the minimum possible makespan. In each move, only one of the three cycles on the figure-8 graph may be used to move the robots and each cycle may be moved in clockwise or counterclockwise direction; no two cycles may be rotated simultaneously. Therefore, the tree has a branching factor of at most 6. Assume the best case in which the tree is balanced and has no duplicate nodes (i.e., configuration), we can bound h_T as $6^{h_T+1} \geq |V|!$. That is, the tree must have at least $|V|!$ unique configuration nodes derived from the root X_I, because all $|V|!$ configurations are reachable from X_I. With Stirling's approximation Bollobás (2013),

$$|V|! \geq \sqrt{2\pi|V|}\left(\frac{|V|}{e}\right)^{|V|},$$

which yields

$$h_T = \Omega(|V| \log |V|).$$

This shows that solving some instances on figure-8 graphs requires $\Omega(|V| \log |V|)$ steps, establishing that TMPP could require a minimum makespan of $\omega(|V|)$.

Because n in the figure-8 graph is an arbitrary non-negative integer, $|V|$ has an infinite number of values. Hence, there is an infinite family of such graphs. □

Proposition 1 implies that if the classes of graphs are not restricted, we cannot always hope for the existence of solutions with linear or better makespan with respect to the number of vertices of the graphs, i.e.,

Corollary 1 *TMPP does not admit solutions with linear or sub-linear makespan on an arbitrary graph.*

Corollary 1 suggests that seeking general algorithms for providing linear or sub-linear makespan that apply to all environments will be a fruitless attempt. With this in mind, the paper mainly focuses on a restricted but very practical class of discrete environments: grid graphs.

4 Routing Robots on Rectangular Grids with a Sub-Linear Makespan

4.1 Main Result

We first outline the main algorithmic result of this work and the key enabling idea behind it, a divide-and-conquer scheme which we denote as *global decoupling*.

Assuming unit edge lengths, a rectangular 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 grid, respectively. Without loss of generality, assume that $m_\ell \geq m_s$ (see Fig. 4 for an 8×4 grid). We further assume that $m_\ell \geq 3$ and $m_s \geq 2$ since an MPP on a smaller grid is trivial. These assumptions are implicitly assumed in this paper whenever *grid* is mentioned, unless otherwise stated. We note that an MPP problem on such a grid is always feasible, as established formally in Yu and Rus (2015). The main result to be proven in this section is the following.

Theorem 1 *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 $O(m_\ell)$ makespan.*

Note that the $O(m_\ell)$ bound is *sub-linear* with respect to the number of vertices, which is $\Omega(m_s m_\ell)$ and $\Omega(m_\ell^2)$ for square grids. We name the algorithm, to be constructed, as SplitAndGroup (SaG) and first sketch how the divide-and-conquer algorithm works at a high level. In this section we focus on the makespan property of SaG. We delay the establishment of polynomial-time complexity and additional properties of the algorithm to Section 5.

Assume without loss of generality that $m_\ell = 2^{k_1}$ and $m_s = 2^{k_2}$ for some integers k_1 and k_2 (we note that our algorithm does not depend on m_ℓ and m_s being powers of 2 at all; the assumption only serves to simplify this high-level explanation). In the first iteration of SaG, it *splits* the grid into two smaller rectangular grids, G_1 and G_2, of size $2^{k_1-1} \times 2^{k_2}$ each. Then, robots are moved so that at the end of the iteration, if a robot has its goal in G_1 (resp., G_2) in X_G, it should be on some arbitrary vertex of G_1 (resp., G_2). This is the *group* operation. An example of a single SaG iteration is shown in Fig. 4. We will show that such an iteration can be completed in $O(m_\ell) = O(2^{k_1})$ steps (makespan). In the second iteration, the same process is carried out on both G_1 and G_2 in parallel, which again requires $O(m_\ell) = O(2^{k_1})$ steps. In the third iteration, we start with four $2^{k_1-1} \times 2^{k_2-1}$ grids and the iteration can be completed in $O(2^{k_1-1}) = O(\frac{m_\ell}{2})$ steps. After $2k_1$ iterations, the problem is solved with a makespan of

$$2O(m_\ell) + 2O(\frac{m_\ell}{2}) + 2O(\frac{m_\ell}{4}) + \ldots + 2O(1) = O(m_\ell).$$

The divide-and-conquer approach that we use share similarities with other decoupling techniques in that it seeks to break down the overall problem into independent subproblems. On the other hand, it significantly differs from previous decoupling schemes in that the decoupling in our case is *global*. Therefore, we denote the scheme as the *global decoupling* technique.

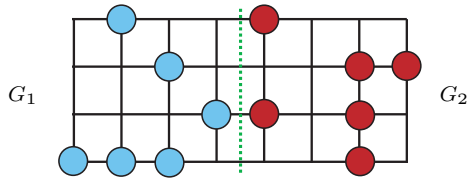


Fig. 4 Illustration of a single iteration of SaG on an 8×4 grid. Note that the grid is fully populated with robots and some are not shown in the figure. The overall grid is split in the middle by the dotted line to give two 4×4 grids, G_1 and G_2. The robots shown on G_1 (resp., G_2) have goal locations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

We proceed to describe an iteration of the SaG algorithm in detail, which depends on following sub-routines, in a sequential manner (i.e., a later sub-routine makes use of the earlier ones):

- Concurrent exchange of multiple pairs robots embedded in a grid in a constant number of steps (Lemma 2).
- Exchange of two groups of robots on a tree embedded in a grid in time steps linear with respect to the diameter (i.e., length of the longest path) of the tree (Lemma 3, Lemma 4, and Theorem 2).
- Partitioning a split problem into multiple exchange problems on trees and solving them concurrently.

Each of these steps is covered in a sub-section that follows.

4.2 Pairwise Exchanges In A Constant Number of Steps

To achieve $O(m_\ell)$ makespan, SaG needs to enable concurrent robot movements. This is 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 would like to simulate the exchange of the two robots on v_1 and v_2 without incurring collision. Let us call this operation $\textsc{flip}(E')$. We use $\textsc{flip}(\cdot)$ to mean the operation is applied to some unspecified set of edges, which is to be determined for the particular situation.

Lemma 2 *Let $G = (V, E)$ be an $m_\ell \times m_s$ grid. Let $E' \subset E$ be a set of vertex disjoint edges. Then the* FLIP(E') *operation can be completed in a constant number of steps.*

Proof A 3×2 rectangular grid may be viewed as a figure-8 graph with $|V| = 6$ vertices. Applying Lemma 1 to the 3×2 grid tells us that any two robots on such a graph can be exchanged without collision. Furthermore, all such exchanges can be pre-computed and performed in $O(1)$ (i.e., a constant number of) steps.

To perform FLIP(E') on an $m_\ell \times m_s$ grid G, we partition the grid into multiple disjoint 3×2 blocks. Using up to 4 different such partitions, it is always possible to cover all edges of G. Therefore, the FLIP(E') operation can be broken down into *parallel* two-robot exchanges on the 3×2 blocks of these partitions. Because of the parallel nature of the two-robot exchanges, the overall FLIP(E') operation can be completed $O(1)$ steps. As an example, Fig. 5 illustrates how a FLIP(E') operations can be carried out on a 7×5 grid. Note that two partitions (the top two in Fig. 5) are sufficient to cover all edges below the second row (including the second row). Then, two more partitions (the bottom two in Fig. 5) can cover all edges above the second row. In the figure, the solid edges represent the edge set E'. After each partition starting from the top left one, two-robot exchanges can be performed which allow the removal of the edges covered by the partition, as shown in the subsequent picture. □

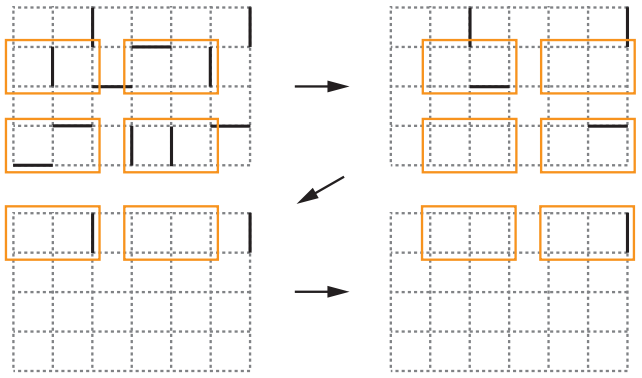


Fig. 5 Illustration of how the FLIP(E') operation can be completed in a constant number of steps on an $m_\ell \times m_s$ grid, which requires up to 4 partitions of the grid into 3×2 blocks.

4.3 Exchange of Groups Robots on an Embedded Tree

Lemma 2, in a nutshell, allows the concurrent exchange of adjacent robots to be performed in $O(1)$ steps. With Lemma 2, to prove Theorem 1, we are left to show that on an $m_\ell \times m_s$ grid, after splitting, the group operation in the first SAG iteration can be decomposed into $O(m_\ell)$ FLIP$(\cdot)$ operations. Because each FLIP$(\cdot)$ can be carried out in $O(1)$ steps, the overall makespan of the group operation is $O(m_\ell)$. To obtain the desired decomposition, we need to maximize parallelism along the split line. 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 group operation will then be carried out on these trees. Before detailing the tree-partitioning step, we show that grouping robots on trees can be done efficiently. We start by showing that we can effectively "herd" a group of robots to the end of a path.[1] 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.

Lemma 3 *Let P be a path of length ℓ embedded in a grid. An arbitrary group of up to $\lfloor \ell/2 \rfloor$ robots on P can be relocated to one end of P in $O(\ell)$ steps. Furthermore, the relocation may be performed using* FLIP$(\cdot)$ *on P.*

Proof Because we are to do the relocation using parallel two-robot exchanges on disjoint edges based on the FLIP$(\cdot)$ operation, without loss of generality, we may assume that the path is straight and we are to move the robots to the right end of the path. An example illustrating the scenario is given in Fig. 6. For a robot in the group, let its initial location on the path be of distance k from the right end. We inductively prove the claim that it takes $O(k)$ steps from the beginning of all moves to "shift" such a robot to its desired goal location.

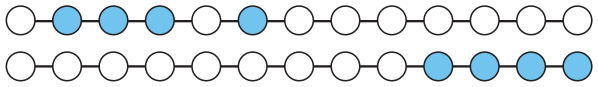


Fig. 6 The initial and goal configurations of a group of 4 robots on a path, before and after a *herding* operation.

At the beginning (i.e., $t = 0$), let the robot on P that is of distance k to the right end be denoted as r_k. The hypothesis trivially holds for $k = 0$. Suppose it holds for $k - 1$ and we need to show that the claim extends to k. If r_k does not belong to the group of robots to be moved, then there is nothing to do. Otherwise, there are two cases.

In the first case, robot r_{k-1} does not belong to the group of robots to be moved. Then at $t = 0$, r_k and r_{k-1} may be exchanged in $O(1)$ steps. Now r_k is of distance $k-1$ to the right and the inductive hypothesis yields that the rest of the moves for r_k can be completed in $O(k-1)$ steps. The total number of steps is then $O(k)$.

[1] We emphasize that the group operation and groups of robots are related but bear different meanings.

In the second case, robot r_{k-1} also belongs to the group of robots to be moved. By the inductive hypothesis, r_{k-1} can be moved to its desired goal in $O(k-1)$ steps. However, once r_{k-1} is moved to the right, it will allow r_k to follow it with a gap between them of at most 2. Once r_{k-1} reaches its goal, r_k, whose goal is on the right of r_{k-1}, can reach its goal in $O(1)$ additional steps. The total number of steps from the beginning is again $O(k)$.

It is clear that all operations can be performed using FLIP$(\cdot)$ on edges of P when embedded in a grid. □

Using the herding operation, the locations of two disjoint groups of robots, equal in number, can also be exchanged efficiently.

Lemma 4 *Let P be a path of length ℓ embedded in a grid. Let two groups, equal in number, reside on two segments of P that do not intersect. Then positions of the two groups of robots may be exchanged in $O(\ell)$ steps without net movements of other robots. The relocation may be performed using* FLIP$(\cdot)$ *on P.*

Proof We may again assume that P is straight. An implicit assumption is that each group contains at most $\lfloor \ell/2 \rfloor$ robots. Fig. 7 illustrates an example in which two groups of 4 robots each need to switch locations on such a path.

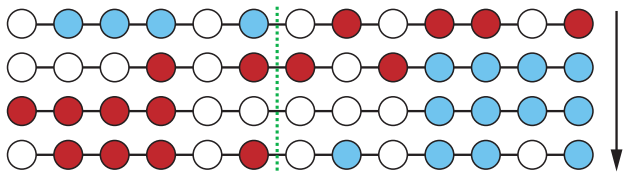


Fig. 7 The initial (first row), goal (last row) and intermediate configurations of two groups of 4 robots to be exchanged. Each group is marked with a different color/shade. The unshaded discs do not belong to either of the two group.

To do the grouping, we first apply a herding operation that moves one group of robots to one end of P. In Fig. 7, this is done to the group of lightly-shaded robots to move them to the right side (the second row of Fig. 7). Then, another herding operation is performed to move the other group to the other end of P (the third row of Fig. 7). In the third and last step, two parallel "reversed" herding operations are carried out on two disjoint segments of P to move them to their desired goal locations. This is best understood by viewing the process as applying the herding operation to the goal configuration. As an example, in Fig. 7, from the goal configuration (last row), we may readily apply two herding operations to move two groups of robots to the two ends of P as shown in the third row of the figure. Because each herding operation takes $O(\ell)$ steps, the overall operation takes $O(\ell)$ steps as well. It is clear that in the end, a robot not in the two groups will not have any net movement on P because the relative orders of these robots (unshaded ones in Fig. 7) never change. □

Next, we generalize Lemma 4 to a tree embedded in a grid. On a tree graph T, we call a subgraph a *path branch* of T if the subgraph is a path with no other attached branches. That is, all vertices of the subgraph have degrees one or two in T.

Theorem 2 *Let T be a tree of diameter d embedded in a grid. 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 $O(d)$ steps without net movement of other robots. The relocation may be performed using* FLIP$(\cdot)$ *on T.*

Proof We temporarily limit the tree T such that, after picking a proper *main path* that contains P and deleting this main path, there are only paths left. That is, we assume all vertices with degree three or four are on a single path containing P. An example of such a tree T and the exchange problem is given in the top row of Fig. 8. In the figure, the main path is the long horizontal path and P is the path on the left of the dotted split line. We call other paths off the main path *side branches*. Once this version is proven, the general version readily follows because all possible tree structures are considered in this special example, i.e., there may be either one or two branches coming out of a node on the main branch. The rest of the paper will only use the less general version. For ease of reference, for the two groups of robots, we denote the group fully on P as g^1 and the other group as g^2. In the example, g^1 has a light shade and g^2 has a darker shade.

To start, we first solve part of the relocation problem on the main path, which can be done in $O(d)$ steps by Lemma 4. After the step, the robots involved in the first step are no longer relevant. In the example, this is to exchange the robots marked with small arrows in the first row of Fig. 8. After the relocation of these robots is completed, we remove their shades.

In the second step, the relevant robots in g^2 on the side branch are moved so that they are just off the main path. We also assign priorities to these robots based on their closeness to P and break ties randomly. For a robot labeled i in a group g^j, we denote the robot as r_i^j. For our example, this current step yields the third row of Fig. 8 with the priorities marked. Since the moves are done in parallel and each branch is of length at most d, only $O(d)$ steps are needed.

In the third step, robots from g^2 will move out of the side branches in the order given, one immediately after the other (when possible). For the example (third row of Fig. 8), r_1^2 will move first. r_2^2 will follow. Then r_3^2, followed by r_4^2. Using the same inductive argument from the proof of Lemma 3, we observe that all robots from g^2 on the

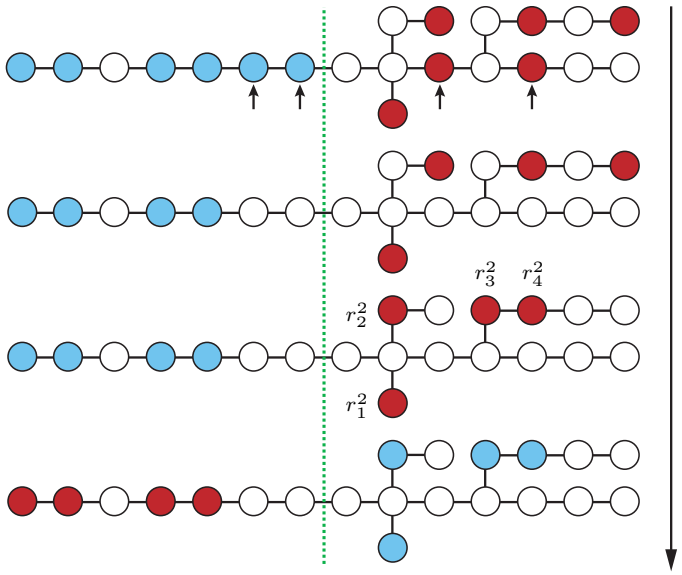


Fig. 8 The initial (first row) and three intermediate configurations in solving the problem of switching the location of these robots on a tree.

side branch can be moved off the side branches (and reach their goals on the main path) in $O(d)$ time. As the relevant robots from g^1 also move across the split line, they will fill in side branches in opposite order to when the robots from g_2 are moved out of the branches. In the example, this means that the branch where r_3^2 and r_4^2 were on will be populated with robots from g^1 first, followed by the branch where r_2^2 was, and finally the branch where r_1^2 was. This ensures that at the end of this step, any robot not in g^1 and g^2 will have no net movement. The number of steps for this is again $O(d)$.

In the last step, we simply reverse the second step, which takes another $O(d)$ steps. Putting everything together, $O(d)$ steps are sufficient for completing the task.

Combining all steps, only $O(d)$ steps are required to complete the desired exchange. To see that the same conclusion holds for more general trees with side branches that are not simple paths, we simply need to do the second step and third step more carefully. But, because we are only moving at most $O(d)$ robots, using an amortization argument, it is straightforward to see that the $O(d)$ bound does not change. □

We note that many of the operations used to prove Lemma 3, Lemma 4, and Theorem 2 can be combined without changing the outcome. However, doing so will make the proofs less modular. Given the focus of the current paper which is to construct a polynomial time algorithm with constant factor optimality guarantee, we opt for clarity instead of pursuing a smaller asymptotic constant.

4.4 Tree Forming and Robot Routing

We proceed to prove Theorem 1 by showing in detail how to carry out a single iteration of SAG, which boils down to partitioning the robot exchanges into robot exchanges on trees, to which Theorem 2 can then be applied. The proof itself can be subdivided into three steps:

- **Splitting and initial tree forming**, where a grid is partitioned into two roughly equal halves and trees are initially formed across the partition line for facilitating exchanging of two groups of robots.
- **Tree post-processing**, which addresses the issue where two initial trees might have "+" like crossovers.
- **Final robot routing**, which actually carries through the robot routing process and resolve some final issues.

Proof (Proof of Theorem 1) **Splitting and initial tree forming**. In a split, we always split along the longer side of the current grid. Since $m_\ell \geq m_s$, the $m_\ell \times m_s$ grid is split into two grids of dimensions $\lceil m_\ell/2 \rceil \times m_s$ and $\lfloor m_\ell/2 \rfloor \times m_s$, respectively. For convenience, we denote the two split grids as G_1 and G_2, respectively. Recall that in the group operation, we want to exchange robots so that a robot with goal in G_1 (resp., G_2) resides in G_1 (resp., G_2) at the end of the operation. To do this efficiently, we need to maximize the parallelism. This is achieved through the computation of a set of m_s trees with which we can apply Theorem 2. We will use the example from Fig. 9 to facilitate the higher level explanation.

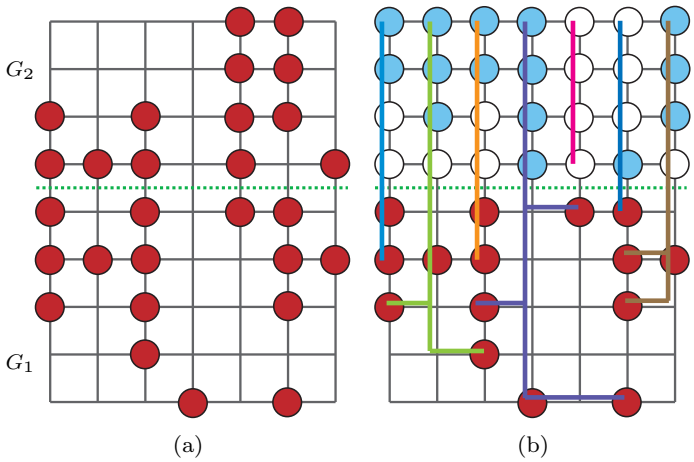


Fig. 9 (a) A 9×7 grid is split into two grids G_1 and G_2 of sizes 4×7 and 5×7, respectively. The dark-shaded robots' final goals are in G_2. (b) The grid is partitioned into (possibly non-disjoint) trees to allow the dark-shaded robots that are not already in G_2 to exchange with robots (lightly-shaded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

Assume that the grid is oriented so that m_s is the number of columns and m_ℓ is the number of rows (see Fig. 9). The trees that will be built will be based on the columns of

one of the split graphs, say G_2. A column i of G_2 is a path of length $\lfloor m_\ell/2 \rfloor - 1$ with $\lfloor m_\ell/2 \rfloor$ robots on it. Suppose k_i of these robots have goals outside G_2 (the lightly-shaded ones in Fig. 9 (b)), then it is always possible to find k_i robots (the dark-shaded ones in Fig. 9(b)) on G_1 that must go to G_2. A tree T_i is built to allow the exchange of these $2k_i$ robots such that the part of T_i in G_2 is simply column i. That is, the m_s trees to be built do not overlap in G_2.

For a column i in G_2 with k_i robots to be moved to G_1, it is not always possible to find exactly k_i robots on column i of G_1. This makes the construction of the trees in G_1 more complex. The construction is done in two steps. In the first step, robots to be moved to G_2 are grouped in a distance optimal manner, which induces a preliminary tree structure. Focusing on G_1, we know the number of robots that must be moved across the split line in each column (see Fig. 10). For each robot to be moved across the split line, the distance between the robot and all the possible exits of G_1 is readily computed. Once these distances are computed, a standard matching procedure can be run to assign each robot an exit point that minimizes the total distance traveled by these robots Kuhn (1955); Solovey et al (2015). The assignment has a powerful property that we will use later. For each robot, either a straight or an L shaped path can be obtained based on the assignment. Merging these paths for robots exiting from the same column then yields a tree for each column (see Fig. 9(b)). Note that each tree has a single vertical segment.

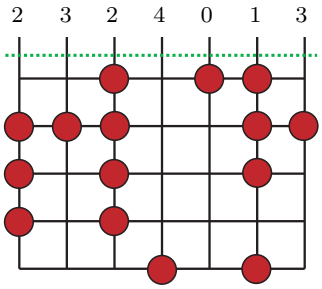


Fig. 10 For the example given in Fig. 9, this figure highlights G_1, the robots that must be moved to G_2, and how many robot need to be moved through the top of G_1 along each column. Regarding distance, the bottom left robot needs to travel 4 edges to exit G_1 through the left most column. It needs to travel $4 + 7 - 1 = 10$ edges to exit from the right most column.

Tree post-processing. In the second step, the trees are post-processed to remove crossings between them. Example of such a crossing we refer to is illustrated in Fig. 11(a) (dotted lines). Formally, we say two trees T_1 and T_2 has a *crossover* if a horizontal path of T_1 intersects with a vertical path of T_2, with the additional requirement that one of the involved horizontal path from one tree forms a + with the vertical segment of the other tree. For example, Fig. 11(b) is not considered a crossover.

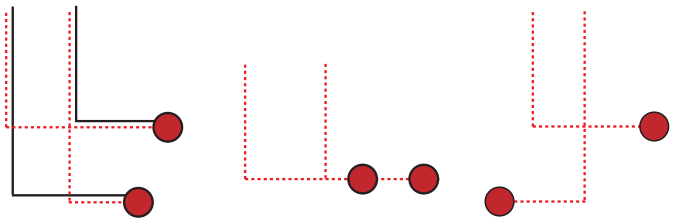


Fig. 11 (a) Example of a tree crossover (dotted paths) and its removal (solid paths) without increasing the total distance. Note that only the relevant paths of the two trees are shown. (b) An intersection that is not considered a crossover. (c) An impossible crossover scenario.

For each crossover, we update the two trees to remove the crossover, as illustrated in Fig. 11(a). The removal will not change the total distance traveled by the two (or more) affected robots but will change the path for these robots. To see that the process will end, note that one of the two involved paths is shortened. Since there are finite number of such paths and each path can only be shortened a finite number of times, the crossover removal process can get rid of all crossovers. We will show later this can be done in polynomial computation time when we perform algorithmic analysis. We note here that the crossover scenario in Fig. 11(c) cannot happen because a removal would shorten the overall length, which contradicts the assumption that these paths have the shortest total distance.

Final robot routing. At the end of the crossover removal process, we may first route all robots on a tree branch that do not have overlaps with other trees. However, this does not route all robots because it is possible for the tree structures for different columns to overlap horizontally (see Fig. 12). For two trees that partially overlap with each other (e.g., the left and middle two trees in Fig. 12), one of the trees does not extend lower (row wise) than the row where the overlap occurs. Otherwise, this yields a crossover, which should have already been removed. For two overlapping trees T_1 and T_2, we say T_1 is a *follower* of T_2 if a robot going to T_2 must pass through the vertical path of T_1. In the example from Fig. 12, T_1 is a follower of T_2. Similarly, the right (green) tree is a follower of T_1.

We state some readily observable properties of overlapping trees: *(i)* two trees may have at most one overlapping horizontal branch (otherwise, there must be a path crossover), *(ii)* because of *(i)*, any three trees cannot pair wise overlap at different rows, and *(iii)* there must be at least one tree that is not a follower, e.g., the left (purple) tree in Fig. 12. We call this tree a *leader*. From a leader tree, we can recursively collect its followers, and the fol-

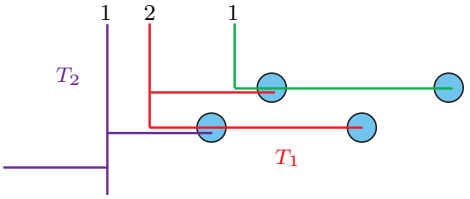


Fig. 12 Illustration of overlaps between tree structures for different columns. For the four shaded robots to be moved, one must be moved through the left most vertical path, two must be moved through the middle vertical path, and the last must be moved through the right most vertical path.

lowers of these followers, and so on and so forth. We call such a collection an *interacting bundle* (e.g., Fig. 12).

With these properties in mind, the group operation in a SAG iteration is carried out as follows. Because robots to be moved from G_2 to G_1 are on straight vertical paths, there are no interactions among them between different trees. Therefore, we only need to consider interactions of robots on G_1. For trees that have no overlap with other trees, Theorem 2 directly applies to complete the robot exchange on these trees in $O(m_\ell)$ steps because each tree has a diameter of at most $2m_\ell$. In parallel, we can also complete the movement of all robots that should go from G_1 to G_2 which are not residing on a horizontal tree branch that overlaps with other trees, also in $O(m_\ell)$ steps. After these robots are exchanged, we can effectively forget about them.

After the previous step, we are left to deal with robots on overlapping horizontal tree branches that must be moved (e.g., the shaded robots in Fig. 12). It is clear that different interacting bundles do not have any interactions; we only need to focus on a single bundle. This is actually straightforward; we use the example from Fig. 13 to facilitate the proof explanation. The routing of robots in this case follows a greedy approach starting from the left most tree, i.e., we essentially try to "flush" the shaded robots in the left-up direction, which can always be realized in two phases, each of which using at most $O(m_\ell)$ makespan.

Observe that the problem can be solved for the leader tree (left most tree in Fig. 13). At the same time, for each successive follower tree, the movement of robots can be partially solved for these follower trees. The middle row of Fig. 13 shows how this can be done for each tree. Formally, if a horizontal branch is shared by two trees, say T_2 and its follower T_1, then we obtain a simple exchange problem of moving a few robots through a path on T_2. In the figure, these are the first and fourth trees from the left, with the dotted lines marking the path. If a horizontal branch is shared by three or more trees, we get an exchange problem on a tree. In the figure, the middle three trees create such a problem. For this, Theorem 2 applies with minor modifications. All the exchange problems can be carried out

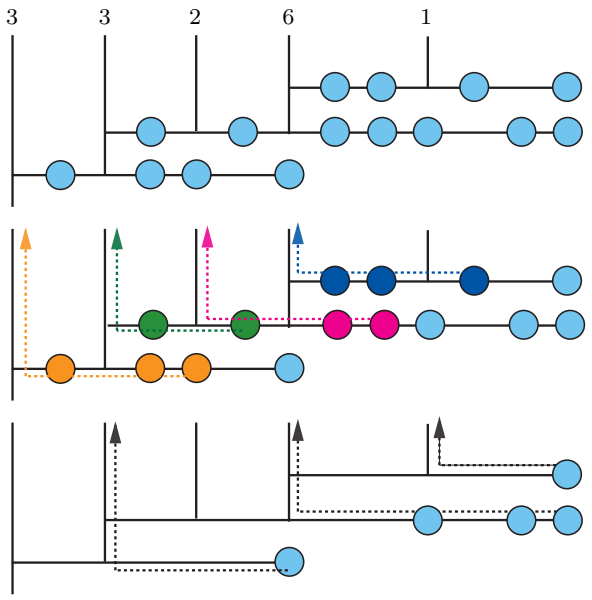


Fig. 13 An example interacting bundle in detail. The top row is the initial configuration of the robots on the overlapping horizontal tree branches to be moved through the vertical paths. The numbers on the top of the vertical paths mark how many robots should be moved through that path. For example, 3 of the shaded robots must be routed through the left most vertical path. The middle row and the bottom rows mark how the exchanges can be completed in two steps or phases.

in parallel because there is no further interaction between them. It takes $O(m_\ell)$ steps to complete, after which we are left with another set of exchange problems, each of which is on a path (e.g., the three problems in the last row of Fig. 13). Lemma 4 applies to yield $O(m_\ell)$ required steps.

Stitching everything together, the first iteration of SAG on an $m_\ell \times m_s$ grid can be completed in $O(m_\ell)$ steps. In the next iteration, we are working with grids of sizes $\lceil m_\ell/2 \rceil \times m_s$, which requires $O(\max\{\lceil m_\ell/2 \rceil, m_s\})$ steps. Following the simple recursion, which terminates after $O(\log m_\ell)$ iterations, we readily obtain that $O(m_\ell)$ steps are sufficient for solving the entire problem. □

5 Complexity and Solution Optimality Properties of the SaG Algorithm

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

5.1 Time Complexity of SAG

The SAG algorithm is outlined in Algorithm 1, which summarizes the results from Section 4 in the form of an algorithm. At Lines 1-2, a partition of the current grid G is

made, over which initial path planning 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 the problem is small enough (Line 7), in which case the problem is directly solved without further splitting.

Algorithm 1: SplitAndGroup (G, X_I, x_G)

Input : $G = (V, E)$: an $m_\ell \times m_s$ grid graph
X_I: initial robot configurations
X_G: goal robot configurations
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 $\mathcal{P} \leftarrow$ MatchAndPlanPath(G, X_I, X_G)

%Remove crossovers
3 $\mathcal{P}' \leftarrow$ ResolveCrossovers($\mathcal{P}$)

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

%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 $M = M +$ SAG$(G_i, X_I'|_{G_i}, X_G|_{G_i})$
10 **end**
11 **end**
12 **return** M

We now proceed to bound the running time of SAG. It is straightforward to see that the Split routine takes $O(|V|) = O(m_\ell m_s)$ running time. MatchAndPlanPath can be implemented using the standard Hungarian algorithm Kuhn (1955), 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 has only two straight segments. Detecting a crossover then takes up to $O(|V|)$ running time. We note that, as a crossover is resolved, one of the two paths will end up being shorter (see, e.g., Fig. 11). 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. Therefore, all together, ResolveCrossovers can be completed in $O(|V|^3)$ time.

The ScheduleMoves routine simply extracts information from the already planned path set $\mathcal{P}'$ and can be completed in $O(|V|)$ running time. The Solve routine takes constant time.

Adding everything up, an iteration of SAG 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.

5.2 Optimality Guarantees

Having established that SAG is a polynomial time algorithm that solves MPP with sub-linear makespan, we now show that SAG is an $O(1)$-approximate makespan optimal algorithm for MPP in the average case. Moreover, SAG also computes a constant factor distance optimal solution in a weaker sense. To establish these, we first show that for fixed $m_\ell m_s$ that is large, the number of MPP instances with $o(m_\ell + m_s)$ makespan is negligible.

Lemma 5 *The fraction of MPP instances on a fixed graph G as an $m_\ell \times m_s$ grid with $o(m_\ell + m_s)$ makespan is no more than $(\frac{1}{2})^{\frac{m_\ell}{4}}$ for sufficiently large $m_\ell m_s$.*

Proof Let (G, X_I, X_G) be an MPP instance wth G being a $m_\ell \times m_s$ grid. Without loss of generality, we may assume that X_I is arbitrary and X_G is a row-major ordering of the robots, i.e., with the i-th row containing the robots labeled $(i-1)m_\ell + 1, (i-1)m_\ell + 2, \ldots, im_\ell$, in that order. Then, over all possible instances, $\frac{1}{4}$ of instances have X_I with robot 1 having a distance of $\frac{1}{4}(m_\ell + m_s)$ or less from robot 1's location in X_G (note that the number is $\frac{1}{4}$ instead of $\frac{1}{16}$ because m_s maybe as small as 2). Let these $\frac{1}{4}$ instances be $\mathcal{P}_1$. Among $\mathcal{P}_1$, again about $\frac{1}{4}$ of instances have have X_I with robot 2 having a distances of $\frac{1}{4}(m_\ell + m_s)$ or less from robot 2's location in X_G. Following this reasoning and limiting to the first $\frac{m_\ell}{4}$ robots, we may conclude that the fraction of instances with $o(m_\ell + m_s)$ makespan is no more than $(\frac{1}{2})^{\frac{m_\ell}{4}}$ for sufficiently large $m_\ell m_s$. □

Lemma 5 is conservative but sufficient for establishing that SAG delivers an average case $O(1)$-approximation for makespan. That is, since only an exponentially small number of instances have small makespan, the average makespan ratio is clearly constant, that is,

Theorem 3 *On average and in polynomial time,* SAG *computes $O(1)$-approximate makespan optimal solutions for MPP.*

Proof For a fixed G as an $m_\ell \times m_s$ grid, Lemma 5 says that no more than a $(\frac{1}{2})^{\frac{m_\ell}{4}}$ fraction of instances have sub-linear makespan (the minimum possible makespan is 1).

For the rest of the instances, their makespan is linear, i.e., $\Omega(m_\ell + m_s)$. On the other hand, Algorithm 1 guarantees a makespan of $O(m_\ell + m_s)$. We may then compute the expected (i.e., average) makespan ratio over all instances of MPP for the same G as no more than (for sufficiently large m_ℓ)

$$(\frac{1}{2})^{\frac{m_\ell}{4}} \frac{O(m_\ell + m_s)}{1} + (1 - (\frac{1}{2})^{\frac{m_\ell}{4}}) \frac{O(m_\ell + m_s)}{\Omega(m_\ell + m_s)} = O(1).$$

This establishes the claim of the theorem. □

SAG can also provide guarantees on total distance optimality. In this case, because every robot contributes to the total distance, a weaker guarantee is ensured (in the case of makespan, one robot's makespan dominates the makespan of all other robots). This leads us to work with a typical (i.e., average) MPP instance instead of working with averages of makespan optimality ratio over all instances. That is, for the makespan case, we first compute the optimality ratio for each instance, which is subsequently averaged. For total distance, we work with a typical random instance and compute the optimality ratio for such a typical instance.

Theorem 4 *For an average MPP instance,* SAG *computes an $O(1)$-approximate total distance optimal solution.*

Proof For an average MPP instance, each robot incurs a minimum 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 $m_s m_\ell$ robots. On the other hand, because SAG produces a solution with an $O(m_\ell)$ makespan, each robot travels a distance of $O(m_\ell)$. Summing this over all robots, the solution from SAG has a total distance of $O(m_s m_\ell^2)$. This matches the lower bound $\Omega(m_s m_\ell^2)$. □

6 Extensions

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

6.1 High Dimensions

SAG can be extended to work for grids of arbitrary dimensions. For dimensions $d \geq 2$, let the grid be $m_1 \times \ldots \times m_d$. Two updates to SAG are needed to make it work for higher dimensions. First, the split line should be updated to a split plane of dimension $d-1$. In the case of $d = 2$, two iterations will halve all dimensions. In the case of general d, d iterations are required. Thus, the approach produces a makespan of $O(d(m_1 + \ldots m_d))$ where m_i is the side length of the i-th dimension. Second, the crossover check becomes more complex; each check now takes $O(d)$ time instead of $O(1)$ time because each path, though still having up to two straight pieces, requires $O(d)$ coordinates to describe. Other than these changes, the rest of SAG continues to work with some minor modifications to the scheduling procedure (which can again be proven to be correct using inductive proofs). The updated SAG algorithm for dimension d therefore runs in $O(d^2|V|^3)$ time because d iterations of split and group are need to halve all dimensions and each iteration takes $O(d|V|^3)$ time. The optimality guarantees, e.g., Theorems 3 and 4, also carry over.

6.2 Well-Connected Environments

The selection of G as a grid plays a critical role in proving the desirable properties of SAG. 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. 9(a)), multiple 2-switches can be completed between these two rows in a constant number of steps. As long as the environment possesses these two features, SAG 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. 14). 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. 14, 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 SAG 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, the overall generalized SAG 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. 15 provides a discrete example and a continuous example of well-connected settings, which include both the environment and the robots.

As mentioned in Sec. 1, well-connected environments are frequently found in real-world applications, e.g., auto-

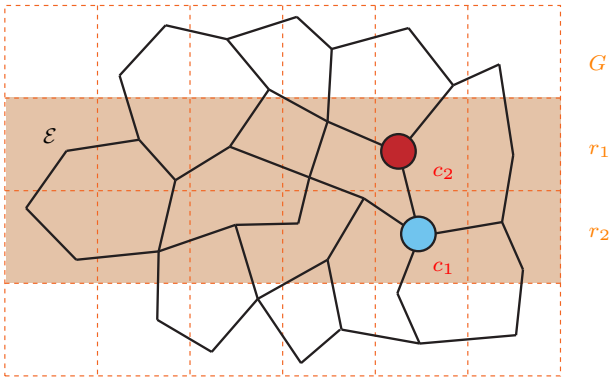


Fig. 14 Illustration of a well-connected non-grid graph.

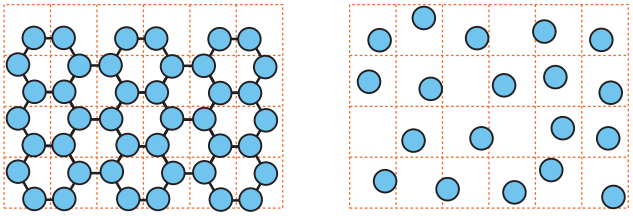


Fig. 15 Examples of well-connected settings, with both environments and robots.

mated 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.

7 Conclusion and Future Work

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

The development of SAG opens up many possibilities for promising future work. On the theoretical side, SAG 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 global decoupling and network flow techniques to seek more optimal algorithms for cases with lower robot density.

Acknowledgments

This work was supported by NSF grants IIS-1617744, IIS-1734419, and IIS-1845888. Opinions or findings expressed in this paper do not necessarily reflect the views of the sponsors.

References

Adler A, De Berg M, Halperin D, Solovey K (2015) Efficient multi-robot motion planning for unlabeled discs in simple polygons. In: Algorithmic Foundations of Robotics XI, Springer, pp 1–17

Alami R, Robert F, Ingrand F, Suzuki S (1995) Multi-robot cooperation through incremental plan-merging. In: Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on, IEEE, vol 3, pp 2573–2579

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

Atzmon D, Stern R, Felner A, Wagner G, Barták R, Zhou NF (2018) Robust multi-agent path finding. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp 1862–1864

Auletta V, Monti A, Parente M, Persiano P (1999) A linear-time algorithm for the feasbility of pebble motion on trees. Algorithmica 23:223–245

Balch T, Arkin RC (1998) Behavior-based formation control for multirobot teams. IEEE Transactions on Robotics & Automation 14(6):926–939

Banfi J, Basilico N, Amigoni F (2017) Intractability of time-optimal multirobot path planning on 2d grid graphs with holes. IEEE Robotics and Automation Letters 2(4):1941–1947

Bekris KE, Tsianos KI, Kavraki LE (2007) 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, pp 3784–3790

van den Berg J, Lin MC, Manocha D (2008) Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proceedings IEEE International Conference on Robotics & Automation, pp 1928–1935

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

Bollobás B (2013) Modern graph theory, vol 184. Springer Science & Business Media

Boyarski E, Felner A, Stern R, Sharon G, Betzalel O, Tolpin D, Shimony E (2015) Icbs: The improved conflict-based search algorithm for multi-agent pathfinding. In: Eighth Annual Symposium on Combinatorial Search

Branicky MS, Curtiss MM, Levine J, Morgan S (2006) Sampling-based planning, control and verification of hybrid systems. IEE Proceedings Control Theory and Applications 153(5):575

Canny JF (1988) The Complexity of Robot Motion Planning. MIT Press, Cambridge, MA

Choset H, Lynch KM, Hutchinson S, Kantor G, Burgard W, Kavraki LE, Thrun S (2005) Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge, MA

Cohen L, Uras T, Kumar T, Xu H, Ayanian N, Koenig S (2016) Improved bounded-suboptimal multi-agent path finding solvers. In: International Joint Conference on Artificial Intelligence

Earl MG, D'Andrea R (2005) Iterative milp methods for vehicle-control problems. IEEE Transactions on Robotics 21(6):1158–1167

Erdem E, Kisa DG, Öztok U, Schueller P (2013) A general formal framework for pathfinding problems with multiple agents. In: AAAI

Erdmann MA, Lozano-Pérez T (1986) On multiple moving objects. In: Proceedings IEEE International Conference on Robotics & Automation, pp 1419–1424

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

Fox D, Burgard W, Kruppa H, Thrun S (2000) A probabilistic approach to collaborative multi-robot localization. Autonomous Robots 8(3):325–344

Goldreich O (2011) Finding the shortest move-sequence in the graph-generalized 15-puzzle is np-hard. In: Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation, Springer, pp 1–5

Goraly G, Hassin R (2010) Multi-color pebble motion on graph. Algorithmica 58:610–636

Griffith EJ, Akella S (2005) Coordinating multiple droplets in planar array digital microfluidic systems. International Journal of Robotics Research 24(11):933–949

Guo Y, Parker LE (2002) A distributed and optimal motion planning approach for multiple mobile robots. In: Proceedings IEEE International Conference on Robotics & Automation, pp 2612–2619

Halperin D, Latombe JC, Wilson R (2000) A general framework for assembly planning: The motion space approach. Algorithmica 26(3-4):577–601

Han SD, Rodriguez EJ, Yu J (2018) Sear: A polynomial-time expected constant-factor optimal algorithmic framework for multi-robot path planning. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots & Systems

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

Hönig W, Kumar TS, Cohen L, Ma H, Xu H, Ayanian N, Koenig S (2016) Multi-agent path finding with kinematic constraints. In: ICAPS, pp 477–485

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

Jansen R, Sturtevant N (2008) A new approach to cooperative pathfinding. In: In International Conference on Autonomous Agents and Multiagent Systems, pp 1401–1404

Jennings JS, Whelan G, Evans WF (1997) Cooperative search and rescue with a team of mobile robots. In: Proceedings IEEE International Conference on Robotics & Automation

Katsev M, Yu J, LaValle SM (2013) Efficient formation path planning on large graphs. In: Proceedings IEEE International Conference on Robotics & Automation, pp 3606–3611

Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. The international journal of robotics research 5(1):90–98

Kloder S, Hutchinson S (2006) Path planning for permutation-invariant multirobot formations. IEEE Transactions on Robotics 22(4):650–665

Knepper RA, Rus D (2012) 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, pp 94–100

Kornhauser DM (1984) Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. PhD thesis, Massachusetts Institute of Technology

Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2:83–97

LaValle SM, Hutchinson SA (1998) Optimal motion planning for multiple robots having independent goals. IEEE Transactions on Robotics & Automation 14(6):912–925

Luna R, Bekris KE (2011) Push and swap: Fast cooperative path-finding with completeness guarantees. In: Proceedings International Joint Conference on Artificial Intelligence, pp 294–300

Ma H, Li J, Kumar T, Koenig S (2017) Lifelong multi-agent path finding for online pickup and delivery tasks. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp 837–845

Matarić MJ, Nilsson M, Simsarian KT (1995) Cooperative multi-robot box pushing. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots & Systems, pp 556–561

Nnaji B (1992) Theory of Automatic Robot Assembly and Programming. Chapman & Hall

Poduri S, Sukhatme GS (2004) Constrained coverage for mobile sensor networks. In: Proceedings IEEE International Conference on Robotics & Automation

Qutub S, Alami R, Ingrand F (1997) 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, IEEE, vol 3, pp 1610–1615

Ratner D, Warmuth M (1990) The $(n^2 - 1)$-puzzle and related relocation problems. Journal of Symbolic Computation 10:111–137

Reif JH (1985) Complexity of the generalized mover's problem. Tech. rep., DTIC Document

Rodriguez S, Amato NM (2010) Behavior-based evacuation planning. In: Proceedings IEEE International Conference on Robotics & Automation, pp 350–355

Rus D, Donald B, Jennings J (1995) Moving furniture with teams of autonomous robots. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots & Systems, pp 235–242

Ryan MRK (2008) Exploiting subgraph structure in multi-robot path planning. Journal of Artificial Intelligence Research 31:497–542

Saha M, Isto P (2006) Multi-robot motion planning by incremental coordination. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp 5960–5963

Sharon G, Stern R, Felner A, Sturtevant N (2012) Conflict-Based Search for Optimal Multi-Agent Path Finding. In: Proc of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Sharon G, Stern R, Goldenberg M, Felner A (2013) The increasing cost tree search for optimal multi-agent pathfinding. Artificial Intelligence 195:470–495

Shucker B, Murphey T, Bennett JK (2007) Switching rules for decentralized control with simple control laws. In: American Control Conference, pp 1485–1492

Silver D (2005) Cooperative pathfinding. In: The 1st Conference on Artificial Intelligence and Interactive Digital Entertainment, pp 23–28

Smith B, Egerstedt M, Howard A (2009) Automatic generation of persistent formations for multi-agent networks under range constraints. ACM/Springer Mobile Networks and Applications Journal 14(3):322–335

Solovey K, Halperin D (2012) k-color multi-robot motion planning. In: Proceedings Workshop on Algorithmic Foundations of Robotics

Solovey K, Halperin D (2015) On the hardness of unlabeled multi-robot motion planning. In: Robotics: Science and Systems (RSS)

Solovey K, Yu J, Zamir O, Halperin D (2015) Motion planning for unlabeled discs with optimality guarantees. In: Robotics: Science and Systems

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

Standley T, Korf R (2011) Complete algorithms for cooperative pathfinding problems. In: Proceedings International Joint Conference on Artificial Intelligence, pp 668–673

Surynek P (2012) Towards optimal cooperative path planning in hard setups through satisfiability solving. In: Proceedings 12th Pacific Rim International Conference on Artificial Intelligence

Tanner H, Pappas G, Kumar V (2004) Leader-to-formation stability. IEEE Transactions on Robotics & Automation 20(3):443–455

Turpin M, Mohta K, Michael N, Kumar V (2014) CAPT: Concurrent assignment and planning of trajectories for multiple robots. International Journal of Robotics Research 33(1):98–112

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

Yu J (2013) A linear time algorithm for the feasibility of pebble motion on graphs. arXiv:13012342

Yu J (2016) Intractability of optimal multi-robot path planning on planar graphs. IEEE Robotics and Automation Letters 1(1):33–40

Yu J (2017) Expected constant-factor optimal multi-robot path planning in well-connected environments. In: Multi-Robot and Multi-Agent Systems (MRS), 2017 International Symposium on, IEEE, pp 48–55

Yu J, LaValle SM (2013a) Multi-agent path planning and network flow. In: Algorithmic Foundations of Robotics X, Springer Tracts in Advanced Robotics, vol 86, Springer Berlin/Heidelberg, pp 157–173

Yu J, LaValle SM (2013b) Structure and intractability of optimal multi-robot path planning on graphs. In: Proceedings AAAI National Conference on Artificial Intelligence, pp 1444–1449

Yu J, LaValle SM (2016) Optimal multi-robot path planning on graphs: Complete algorithms and effective heuristics. IEEE Transactions on Robotics 32(5):1163–1177

Yu J, Rus D (2015) Pebble motion on graphs with rotations: Efficient feasibility tests and planning. In: Algorithmic Foundations of Robotics XI, Springer Tracts in Advanced Robotics, Springer Berlin/Heidelberg, vol 107, pp 729–746