Sub-1.5 Time-Optimal Multi-Robot Path Planning on Grids in Polynomial Time

Teng Guo   Jingjin Yu

Abstract—**It is well-known that graph-based multi-robot path planning (MRPP) is NP-hard to optimally solve. In this work, we propose the first low polynomial-time algorithm for MRPP achieving 1–1.5 asymptotic optimality guarantees on solution makespan (i.e., the time it takes to complete a reconfiguration of the robots) for random instances under very high robot density, with high probability. The dual guarantee on computational efficiency and solution optimality suggests our proposed general method is promising in significantly scaling up multi-robot applications for logistics, e.g., at large robotic warehouses.**

Specifically, on an $m_1 \times m_2$ gird, $m_1 \geq m_2$, our RTH (Rubik Table with Highways) algorithm computes solutions for routing up to $\frac{m_1 m_2}{3}$ robots with uniformly randomly distributed start and goal configurations with a makespan of $m_1 + 2m_2 + o(m_1)$, with high probability. Because the minimum makespan for such instances is $m_1 + m_2 - o(m_1)$, also with high probability, RTH guarantees $\frac{m_1 + 2m_2}{m_1 + m_2}$ optimality as $m_1 \to \infty$ for random instances with up to $\frac{1}{3}$ robot density, with high probability. $\frac{m_1 + 2m_2}{m_1 + m_2} \in (1, 1.5]$. Alongside this key result, we also establish a series of related results supporting even higher robot densities and environments with regularly distributed obstacles, which directly map to real-world parcel sorting scenarios. Building on the baseline methods with provable guarantees, we have developed effective, principled heuristics that further improve the computed optimality of the RTH algorithms. In extensive numerical evaluations, RTH and its variants demonstrate exceptional scalability as compared with methods including ECBS and DDM, scaling to over 450×300 grids with $45,000$ robots, and consistently achieves makespan around 1.5 optimal or better, as predicted by our theoretical analysis.

I. INTRODUCTION

We examine multi-robot path planning (MRPP, also known as multi-agent path finding or MAPF [45]) on two-dimensional grids, with potentially regularly distributed obstacles (see Fig. 1). The main objective of MRPP is to find a set of collision-free paths for routing many robots from a start configuration to a goal configuration. In practice, solution optimality is also of key importance; yet optimally solving MRPP is generally NP-hard [46, 55], even in planar [52] and grid settings [10]. MRPP algorithms find many important large-scale applications, including, e.g., in warehouse automation for general order fulfillment [51], grocery order fulfillment [35], and parcel sorting [50]. Other application scenarios include formation reconfiguration [38], agriculture [6], object transportation [40], swarm robotics [39, 24], to list a few.

Motivated by applications including grocery fulfillment and parcel sorting, we focus on MRPP in which the underlying

The authors are with the Department of Computer Science, Rutgers, the State University of New Jersey, Piscataway, NJ, USA. E-Mails: {teng.guo, jingjin.yu}@rutgers.edu.
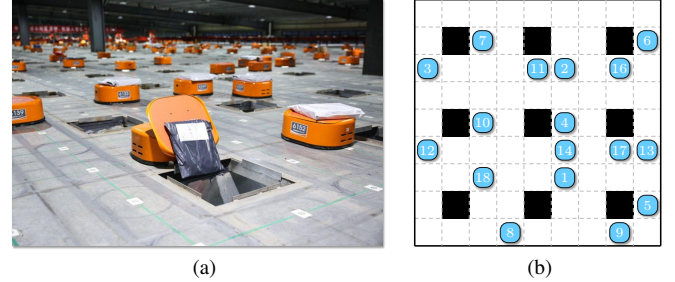
Fig. 1: (a) Real-world parcel sorting system (by JD.com) using many robots on a large grid-like environment with holes for dropping parcels; (b) A snapshot of a similar MRPP instance we can solve in polynomial-time with provable optimality guarantees. In practice, our algorithms scale to maps of size 450×300, supporting over 50K robots, and achieves $1.x$-optimality (see, e.g., Fig. 11).

graph is an $m_1 \times m_2$ grid, $m_1 \geq m_2$, with extremely high robot density. Whereas recent studies [53, 10] have shown that such problems can be solved in polynomial time with $O(1)$ optimality guarantees, the constant factor associated with the guarantee is generally prohibitively high ($\gg 1$) for these methods to be practical. In this research, we break this barrier by showing that, we can achieve $(1 + \delta)$-makespan optimality for MRPP on large grids in polynomial-time in which $\delta \in (0, 0.5 + \varepsilon]$, $\varepsilon \to 0$ as $m_1 \to \infty$. Through the judicious application of a novel global object rearrangement method called Rubik Tables [48] together with many algorithmic techniques, and combined with careful analysis, we establish that in polynomial time:

- For $m_1 m_2$ robots, i.e., at maximum robot density, RTM (Rubik Table for MRPP) computes a solution for an arbitrary MRPP instance under a makespan of $7m_1 + 14m_2$;
- For $\frac{m_1 m_2}{3}$ robots and uniformly randomly distributed start/goal configurations, RTH (Rubik Tables with Highways) computes a solution with a makespan of $m_1 + 2m_2 + o(m_1)$, with high probability. In contrast, such an instance has a minimum makespan of $m_1 + m_2 - o(m_1)$ with high probability. This implies that, as $m_1 \to \infty$, an optimality guarantee of $\frac{m_1 + 2m_2}{m_1 + m_2} \in (1, 1.5]$ is achieved, with high probability;
- For $\frac{m_1 m_2}{3}$ robots, for an arbitrary (i.e., not necessarily random) instance, a solution can be computed with a makespan of $3m_1 + 4m_2 + o(m_1)$ using RTH;
- For $\frac{m_1 m_2}{2}$ robots, the same $\frac{m_1 + 2m_2}{m_1 + m_2}$ optimality guarantee can be achieved with a slightly larger overhead using RTLM (Rubik Tables with Line Merge);
- The same $\frac{m_1 + 2m_2}{m_1 + m_2}$ optimality guarantee may be achieved on grids with up to $\frac{m_1 m_2}{9}$ regularly distributed obstacles

together with $\frac{2m_1m_2}{9}$ robots using RTH (e.g., Fig. 1(b)).

Moreover, we have developed effective and principled heuristics to work together with RTH that further reduce the computed makespan by a large margin, i.e., for $\frac{m_1m_2}{3}$ robots, a makespan smaller than $m_1 + 2m_2$ can often be achieved. Demonstrated through extensive numerical evaluations, our methods are highly scalable, capable of solving instances with tens of thousands of robots in dense settings under two minutes. Simultaneously, the solution optimality approaches the 1–1.5 range as predicted theoretically. This level of scalability far exceeds what was possible. With the sub-1.5 optimality guarantee, our approach unveils a promising direction toward the development of practical, provably optimal multi-robot routing algorithms that runs in low polynomial time.

Related work. Literature on multi-robot path and motion planning [25, 12] is expansive; here, we mainly focus on graph-theoretic (i.e., the state space is discrete) studies [56, 45]. As such, in this paper, MRPP refers explicitly to graph-based multi-robot path planning. Whereas the feasibility question has long been positively answered for MRPP [26], the same cannot be said when it comes to securing optimal solutions, as computing time- or distance-optimal solutions are shown to be NP-hard in many settings, including for general graphs [16, 46, 55], planar graphs [52, 2], and even regular grids [10], similar to the setting addressed in this study.

Nevertheless, given its high utility, especially in e-commerce applications [51, 35, 50] that are expected to grow significantly [9, 1], many algorithmic solutions have been proposed for optimally solving MRPP. Among these, combinatorial-search based solvers [27] have been demonstrated to be fairly effective. MRPP solvers may be classified as being optimal or suboptimal. Reduction-based optimal solvers solve the problem through reducing the MRPP problem to other problem, e.g., SAT [47], answer set programming [11], integer linear programming (ILP) [56]. Search-based optimal MRPP solvers include EPEA* [15], ICTS [42], CBS [43], M* [49], and many others. Due to the inherent intractability of optimal MRPP, optimal solvers usually exhibit limited scalability, leading to considerable interests in suboptimal solvers. Unbounded solvers like push-and-swap [31], push-and-rotate [8], windowed hierarchical cooperative A* [44], all return feasible solutions very quickly, but at the cost of solution quality. Balancing the running-time and optimality is one of the most attractive topics in the study of MRPP/MAPF. Some algorithms emphasize the scalability without sacrificing as much optimality, e.g., ECBS [3], DDM [22], EECBS [30], PIBT [36], PBS [34]. There are also learning-based solvers [7, 41] that scales well in sparse environments. Effective orthogonal heuristics have also been proposed [18]. Recently, $O(1)-$approximate or constant factor time-optimal algorithms have been proposed, e.g. [53, 10], that tackle highly dense instances. However, these algorithms only achieve low-polynomial time guarantees at the expense of very large constant factors, rendering them theoretically interesting but impractical.

In contrast, with high probability, our methods run in low polynomial time with provable 1–1.5 asymptotic optimality. To our knowledge, this paper presents the first MRPP algorithms to simultaneously guarantee polynomial running time and $1.x$ solution optimality.

Organization. The rest of the paper is organized as follows. In Sec. II, we provide a formal definition of graph-based MRPP, and introduce the Rubik Table problem and the associated algorithm. RTM, a basic adaptation of the Rubik Table results for MRPP at maximum robot density which ensures a makespan upper bound of $7m_1 + 14m_2$, is described in Sec. III. An accompanying lower bound of $m_1 + m_2 - o(m_1)$ for random MRPP instances is also established. In Sec. IV we introduce RTH for one third robot density achieving a makespan of $m_1 + 2m_2 + o(m_1)$. Obstacle support is also discussed. In Sec. V, we show how one-half robot density may be supported with optimality guarantees similar to that of RTH. We thoroughly evaluate the performance of our methods in Sec. VI and conclude with Sec. VII. Given the amount of material included in the work, to provide a concentrated discussion, we refer the readers to [17] for proofs to theorems.

II. PRELIMINARIES

A. Multi-Robot Path Planning on Graphs

Graph-based multi-robot path planning (MRPP) seeks collision-free paths that efficiently route robots. Consider an undirected graph $\mathcal{G}(V, E)$ and n robots with start configuration $S = \{s_1, \ldots, s_n\} \subseteq V$ and goal configuration $G = \{g_1, \ldots, g_n\} \subseteq V$. Each robot has start and goal vertices s_i, g_i. We define a *path* for robot i as a map $P_i : \mathbb{N} \to V$ where $\mathbb{N}$ is the set of non-negative integers. A feasible P_i must be a sequence of vertices that connects s_i and g_i: 1) $P_i(0) = s_i$; 2) $\exists T_i \in \mathbb{N}$, s.t. $\forall t \geq T_i, P_i(t) = g_i$; 3) $\forall t > 0, P_i(t) = P_i(t-1)$ or $(P_i(t), P_i(t-1)) \in E$.

With warehouse automation-like applications in mind, we work with $\mathcal{G}$ being 4-connected grids, aiming to minimize the *makespan*, i.e., $\max_i\{|P_i|\}$. Unless stated otherwise, $\mathcal{G}$ is assumed to be an $m_1 \times m_2$ grid with $m_1 \geq m_2$. Also, "randomness" in this paper always refers to uniform randomness. The version of MRPP we study is sometimes referred to as the *one-shot* MAPF problem [45]. We mention that our results also translate to guarantees on the life-long setting [45], which is briefly discussed in Sec. VII.

B. The Rubik Table Problem (RTP)

The Rubik Table problem (RTP) [48] formalizes the task of carrying out globally coordinated token swapping operations on lattices, with many interesting applications. The problem has many variations; in our study, we use the basic 2D form and the associated algorithms, which are summarized below.

Problem 1 (**Rubik Table Problem (RTP)** [48]). Let M be an $m_1(row) \times m_2(column)$ table, $m_1 \geq m_2$, containing m_1m_2 items, one in each table cell. The m_1m_2 items are of m_2 colors with each color having a multiplicity of m_1. In a *shuffle* operation, the items in a single column or a single row of M may be permuted in an arbitrary manner. Given an arbitrary configuration X_I of the items, find a sequence of shuffles that

take M from X_I to the configuration where row i, $1 \leq i \leq m_1$, contains only items of color i. The problem may also be *labeled*, i.e., each item has a unique label in $1, \ldots, m_1 m_2$.

A key result from [48], which we denote as the Rubik Table Algorithm (RTA), establishes that a colored RTP can be solved using m_2 column shuffles followed by m_1 row shuffles. Additional m_1 row shuffles then solve the labeled RTP.

Theorem 1 (**Rubik Table Theorem [48]**)**.** *An arbitrary Rubik Table problem on an $m_1 \times m_2$ table can be solved using $m_1 + m_2$ shuffles. The labeled Rubik Table problem can be solved using $2m_1 + m_2$ shuffles.*

We briefly illustrate how RTA works on an $m_1 \times m_2$ table with $m_1 = 4$ and $m_2 = 3$ (in Fig. 2); we refer readers to [48] for more details. RTA operates in two phases. In the first phase, a bipartite graph $B(T, R)$ is constructed based on the initial table configuration where the partite set T are the colors/types of items, and the set R are the rows of the table (Fig. 2(b)). An edge is added to B between $t \in T$ and $r \in R$ for every item of color t in row r. From $B(T, R)$, a set of m_2 *perfect matchings* can be computed, as guaranteed by [21]. Each matching, containing m_1 edges, connects all of T to all of R, and dictates how a column should look like after the first phase. For example, the first set of matching in solid lines in Fig. 2(b) says that the first column should be ordered as yellow, cyan, red, and green as shown in Fig. 2(c). After all matchings are processed, we get an intermediate table, Fig. 2(c). Notice that each row of Fig. 2(a) can be shuffled to yield the corresponding row of Fig. 2(c); this is the key novelty of the RTA. After the first phase of m_1 row shuffles, the intermediate table (Fig. 2(c)) can then be rearranged with m_2 column shuffles to solve the colored RTP (Fig. 2(d)). Another m_1 row shuffles can then solve the labeled RTP (Fig. 2(e)). We note that it is also possible to perform labeled rearrangement using m_2 column shuffles followed by m_1 row shuffles and then followed by another m_2 column shuffles.
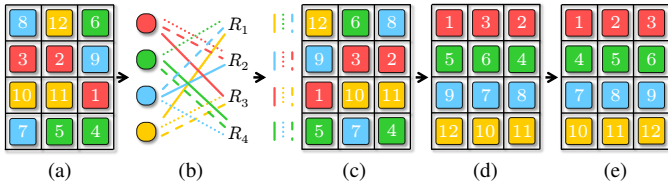

Fig. 2: Illustration of applying the 11 *shuffles*. (a) The initial 4×3 table with a random arrangement of 12 items that are colored and labeled. The labels are consistent with the colors. (b) The constructed bipartite graph. It contains 3 perfect matchings, determining the 3 columns in (c); only color matters in this phase. (c) Applying 4 row shuffles to (a), according to the matching results, leads to an intermediate table where each column has one color appearing exactly once. (d) Applying 3 column shuffles to (c) solves a colored RTP. (e) 4 additional row shuffles fully sort the labeled items.

RTA runs in $O(m_1 m_2 \log m_1)$ (notice that this is nearly linear with respect to $n = m_1 m_2$, the total number of items) expected time or $O(m_1^2 m_2)$ deterministic time. If $m_1 = m_2 = m$, then the times become $O(m^2 \log m)$ expected and $O(m^3)$ deterministic, respectively.

III. Solving MRPP up to Maximum Density w/ RTA

The built-in global coordination capability of RTA naturally applies to solving makespan-optimal MRPP. Since RTA only requires *three rounds* of shuffles and each round involves either parallel row shuffles or parallel column shuffles, if each round of shuffles can be realized with makespan proportional to the size of the row/column, then a makespan upper bound of $O(m_1 + m_2)$ can be guaranteed. This is in fact achievable even when all of $\mathcal{G}$'s vertices are occupied by robots, by recursively applying a *labeled line shuffle algorithm* [53], which can arbitrarily rearrange a line of m robots embedded in a grid using $O(m)$ makespan.

Lemma 2 (Basic Simulated Labeled Line Shuffle [53])**.** *For m labeled robots on a straight path of length m, embedded in a 2D grid, they may be arbitrarily ordered in $O(m)$ steps. Moreover, multiple such reconfigurations can be performed on parallel paths within the grid.*

The key operation is based on a localized, 3-step pair swapping routine, shown in Fig. 3. For more details on the line shuffle routine, see [53].
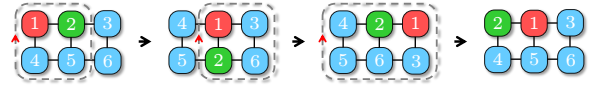

Fig. 3: On a 2×3 grid, swapping two robots may be performed in three steps with three cyclic rotations.

The basic simulated labeled line-shuffle algorithm, however, has a large constant factor. Borrowing ideas from *parallel odd-even sort* [4], we can greatly reduce the constant factor in Lemma 2. First, we need the following lemma.

Lemma 3. *It takes at most seven steps and six steps to perform arbitrary combinations of pairwise horizontal swaps on 3×2 grids and 4×2 grids, respectively.*

Proof of Lemma 3. Using integer programming [56], we exhaustively compute makespan-optimal solutions for arbitrary horizontal reconfiguration on 3×2 (8 possible cases) and 4×2 grids (16 possible cases), which confirms the claim. $\square$

As an example, it takes seven steps to horizontally "swap" all three pairs of robots on a 3×2 grid, as shown in Fig. 4.
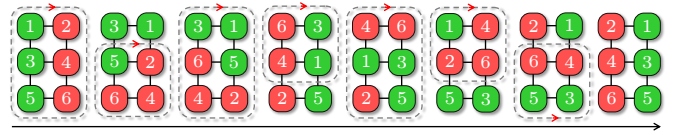

Fig. 4: An example of a horizontal "swap" on a 3×2 grid that takes seven steps, in which all three pairs are swapped. It takes at most seven steps to horizontally swap robots arbitrarily on a 3×2 grid.

Lemma 4 (Faster Line Shuffle)**.** *For m robots on a straight path of length m, embedded in a 2D grid, they may be arbitrarily ordered in $7m$ steps. Moreover, multiple such reconfigurations can be performed simultaneously on parallel straight paths within the grid.*

Proof. "Sorting" of m robots on a straight path of length m may be realized using parallel odd-even sort [4] in $m-1$ rounds, which only requires the ability to simulate potential pairwise "swaps" interleaving odd phases (swapping robots located at positions $2k+1$ and $2k+2$ on the path for some k) and even phases (swapping robots located at positions $2k+2$ and $2k+3$ on the path for some k). Here, it does not matter whether m is odd or even. To simulate these swaps, we can partition the grid embedding the path into 3×2 grids in two ways for the two phases, as illustrated in Fig. 5.
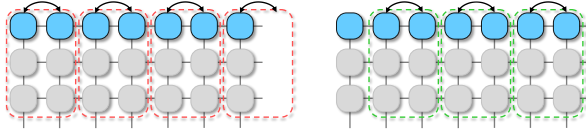

Fig. 5: Partitioning a grid into disjoint 3×2 grids in two ways for simulating odd-even sort. The highlighted pairs of robots maybe independently "swapped" within each 3×2 grid as needed.

A perfect partition requires that the second dimension of the grid, perpendicular to the straight path, be a multiple of 3. If this is not the case, some partitions at the bottom can use 4×2 grids. By Lemma 3, each odd-even sorting phase can be simulated using at most $7m$ steps. Clearly, shuffling on parallel paths is directly supported. $\square$

Combining RTA and fast line shuffle (Lemma 4) yields a polynomial time MRPP algorithm for fully occupied grids with a makepsan of $7m_1 + 14m_2$.

Theorem 5 (MRPP on Grids under Maximum Robot Density, Upper Bound). MRPP *on an $m_1 \times m_2$ grid, $m_1 \geq m_2 \geq 3$, with each grid vertex occupied by a robot, can be solved in polynomial time in a makespan of $7m_1 + 14m_2$.*

We note that the case of $m_2 = 2$ can also be solved similarly except when $m_1 = 2$, with a slightly altered procedure since we can only use partitions of 2×3 grids. We omit the details for this minor case which readers can readily fill in.

The straightforward pseudo-code for RTM, the RTA based algorithm for MRPP on grids supporting the maximum possible robot density, is given in Alg. 1. The comments in the main RTM routine indicate the corresponding RTA phases. For MRPP on an $m_1 \times m_2$ grid with row-column coordinates (x, y), we say robot i belongs to color $1 \leq j \leq m_1$ if $g_i.y = j$. Function `Prepare()` in the first phase finds intermediate states $\{\tau_i\}$ for each robot through perfect matchings and routes them towards the intermediate states by (simulated) column shuffles. If the robot density is smaller than required, we may fill the table with "virtual" robots [23, 53]. For each robot i we have $\tau_i.y = s_i.y$. Function `ColumnFitting()` in the second phase routes the robots to their second intermediate states $\{\mu_i\}$ through row shuffles where $\mu_i.x = \tau_i.x$ and $\mu_i.y = g_i.y$. In the last phase, function `RowFitting()` routes the robots to their final goal positions using additional column shuffles.

We now establish the optimality guarantee of RTM, assuming MRPP instances are randomly generated. For rearranging

---

Algorithm 1: Rubik Table Based MRPP Solver

Input: Start and goal vertices $S = \{s_i\}$ and $G = \{g_i\}$

1 **Function** RTM(S, G)**:**
2   `Prepare`(S, G) $\triangleright$ Computing Fig. 2(b)
3   `ColumnFitting`(S, G) $\triangleright$ Fig. 2(a) $\rightarrow$ Fig. 2(c)
4   `RowFitting`(S, G) $\triangleright$ Fig. 2(c) $\rightarrow$ Fig. 2(d)

5 **Function** `Prepare`(S, G)**:**
6    $A \leftarrow [1, ..., m_1m_2]$
7   **for** $(t, r) \in [1, ..., m_1] \times [1, ..., m_1]$ **do**
8   **if** $\exists i \in A$ *where* $s_i.x = r \wedge g_i.y = t$ **then**
9   add edge (t, r) to $B(T, R)$
10   remove i from A
11   compute matchings $\mathcal{M}_1, ..., \mathcal{M}_{m_2}$ of $B(T, R)$
12    $A \leftarrow [1, ..., m_1m_2]$
13   **foreach** $\mathcal{M}_r$ *and* $(t, r) \in \mathcal{M}_r$ **do**
14   **if** $\exists i \in A$ *where* $s_i.x = r \wedge g_i.y = t$ **then**
15        $\tau_i \leftarrow (r, s_i.y)$ and remove i from A
16   mark robot i to go to τ_i
17   perform simulated column shuffles in parallel

18 **Function** `ColumnFitting`(S, G)**:**
19   **foreach** $i \in [1, ..., m_1m_2]$ **do**
20      $\mu_i \leftarrow (\tau_i.x, g_i.y)$ and mark robot i to go to μ_i
21   perform simulated row shuffles in parallel

22 **Function** `RowFitting`(S, G)**:**
23   **foreach** $i \in [1, ..., m_1m_2]$ **do**
24   mark robot i to go to g_i
25   perform simulated column shuffles in parallel

---

robots on an $m_1 \times m_2$ grid, the expected makespan lower bound on random instances is $\Omega(m_1 + m_2)$ [53]. To obtain a finer optimality ratio, however, a finer lower bound is needed, which is established in the following.

Proposition 6 (Precise Makespan Lower Bound of MRPP on Grids). *The minimum makespan of random* MRPP *instances on an $m_1 \times m_2$ grid with $\Theta(m_1m_2)$ robots is $m_1 + m_2 - o(m_1)$ with arbitrarily high probability as $m_1 \to \infty$.*

Proof. Without loss of generality, let the constant in $\Theta(m_1m_2)$ be some $c > 0$, i.e., there are cm_1m_2 robots. We examine the top left and bottom right corners of the $m_1 \times m_2$ grid $\mathcal{G}$. In particular, let $\mathcal{G}_{tl}$ (resp., $\mathcal{G}_{br}$) be the top left (resp., bottom right) $\alpha m_1 \times \alpha m_2$ sub-grid of $\mathcal{G}$, for some positive constant $\alpha \ll 1$. For $u \in V(\mathcal{G}_{tl})$ and $v \in V(\mathcal{G}_{br})$, assuming each grid edge has unit distance, then the Manhattan distance between u and v is at least $(1 - 2\alpha)(m_1 + m_2)$. Now, the probability that some $u \in V(\mathcal{G}_{tl})$ and $v \in V(\mathcal{G}_{br})$ are the start and goal, respectively, for a single robot, is α^4. For cm_1m_2 robots, the probability that at least one robot's start and goal fall into $\mathcal{G}_{tl}$ and $\mathcal{G}_{br}$, respectively, is $p = 1 - (1 - \alpha^4)^{cm_1m_2}$.

Because $(1 - x)^y < e^{-xy}$ for $0 < x < 1$ and $y > 0$ [1], $p > 1 - e^{-\alpha^4 cm_1m_2}$. Therefore, for arbitrarily small α, we may choose m_1 such that p is arbitrarily close to 1. For example,

---

[1]This is because $\log(1 - x) < -x$ for $0 < x < 1$; multiplying both sides by a positive y and exponentiate with base e then yield the inequality.

we may let $\alpha = m_1^{-\frac{1}{8}}$, which decays to zero as $m_1 \to \infty$, then it holds that the makespan is $(1 - 2\alpha)(m_1 + m_2) = m_1 + m_2 - 2m_1^{-\frac{1}{8}}(m_1 + m_2) = m_1 + m_2 - o(m_1)$ with probability $p > 1 - e^{-c\sqrt{m_1 m_2}}$. $\qquad\square$

Comparing the upper bound established in Theorem 5 and the lower bound from Proposition 6 immediately yields

Theorem 7 (Optimality Guarantee of RTM). *For random MRPP instances on an $m_1 \times m_2$ grid with $\Omega(m_1 m_2)$ robots, $m_1 \geq m_2 \geq 3$, as $m_1 \to \infty$, RTM computes in polynomial time solutions that are $7(1 + \frac{m_2}{m_1 + m_2})$-makespan optimal, with high probability.*

We emphasize that RTM always runs in polynomial time and is not limited by any probabilistic guarantee; the high probability guarantee is only for solution optimality. The same is true for other algorithms' high probability guarantees proposed in this paper. We also note that high probability guarantees are stronger than and imply guarantees in expectation.

IV. NEAR-OPTIMALLY SOLVING MRPP WITH UP TO ONE THIRD ROBOT DENSITY

Though RTM runs in polynomial time and provides constant factor makespan optimality in expectation, the constant factor is still relatively large due to the extreme density. In practice, a robot density of around $\frac{1}{3}$ (i.e., $n = \frac{m_1 m_2}{3}$) is already very high. As it turns out, with $n = cm_1 m_2$ for some constant $c > 0$ and $n \leq \frac{m_1 m_2}{3}$, which is assumed throughout this section, the constant factor can be dropped significantly by employing a "highway" heuristic to simulate the row/column shuffle operations.

A. Rubik Table for MRPP with "Highway" Shuffle Primitive

Random MRPP instances. For the highway heuristics, we first work with random MRPP instances. Let us assume for the moment that m_1 and m_2 are multiples of three; we partition $\mathcal{G}$ into 3×3 cells (see, e.g., Fig. 1(b) and Fig. 6). We use Fig. 6, where Fig. 6(a) is a random start configuration and Fig. 6(f) is a random goal configuration, as an example to illustrate RTH– Rubik Table (for MRPP) with Highways, targeting robot density up to $\frac{1}{3}$. RTH involves two phases: *anonymous reconfiguration* and MRPP *resolution with Rubik Table and highway heuristics*.

In the anonymous reconfiguration phase, in which robots are treated as being indistinguishable or unlabeled, arbitrary start and goal configurations (under $\frac{1}{3}$ robot density) are converted to intermediate configurations where each 3×3 cell contains no more than 3 robots. We call such configurations *balanced configurations*. With high probability, random MRPP instances are not far from being balanced. To establish this result (Proposition 9), we need the following.

Theorem 8 (Minimax Grid Matching [28]). *Consider an $m \times m$ square containing m^2 points following the uniform distribution. Let ℓ be the minimum length such that there exists a perfect matching of the m^2 points to the grid points in the*
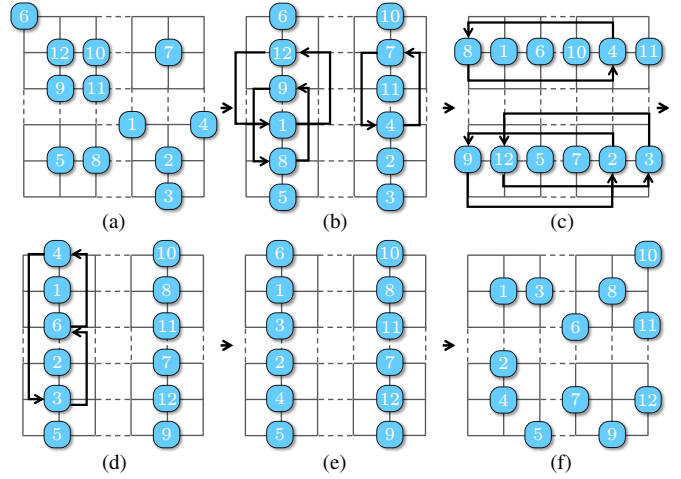

Fig. 6: An example of applying RTH to solve an MRPP instance. (a) The start configuration; (b) The start balanced configuration obtained from (a); (c) The intermediate configuration obtained from the Rubik Table preparation phase; (d) The intermediate configuration obtained from the column fitting phase. Apply additional column shuffles for labeled items; (e) The goal balanced configuration obtained from the goal configuration; (f) The goal configuration.

square for which the distance between every pair of matched points is at most ℓ. Then $\ell = O(\log^{\frac{3}{4}} m)$ with high probability.

Theorem 8 applies to rectangles with the longer side being m as well (Theorem 3 in [28]).

Proposition 9. *On an $m_1 \times m_2$ grid, with high probability, a random configuration of $n = \frac{m_1 m_2}{3}$ robots is of distance $o(m_1)$ to a balanced configuration.*

Proof. We prove for the case of $m_1 = m_2 = 3m$ using the minimax grid matching theorem (Theorem 8); generalization to $m_1 \geq m_2$ can be then seen to hold using the generalized version of Theorem 8 that applies to rectangles (Theorem 3 of [28], which in fact applies to arbitrarily simply connected region within a square region).

Now let $m_1 = m_2 = 3m$. We may view a random configuration of m^2 robots on a $3m \times 3m$ grid as randomly placing m^2 continuous points in an $m \times m$ square with scaling (by three in each dimension) and rounding. By Theorem 8, a random configuration of m^2 continuous points in an $m \times m$ square can be moved to the m^2 grid points at the center of the m^2 disjoint unit squares within the $m \times m$ square, where each point is moved by a distance no more than $O(\log^{\frac{3}{4}} m)$, with high probability. Translating this back to a $3m \times 3m$ gird, we have that m^2 randomly distributed robots on the grid can be moved so that each 3×3 cell contains exactly one robot and the maximum distance moved for any robot is no more than $O(\log^{\frac{3}{4}} m)$, with high probability. Applying this argument three times yields that a random configuration of $\frac{m_1^2}{3}$ robots on an $m_1 \times m_1$ gird can be moved so that each 3×3 cell contains exactly three robots and no robot needs to move more than a $O(\log^{\frac{3}{4}} m_1)$ steps, with high probability. We note that, because the robots are indistinguishable, overlaying

three sets of reconfiguration paths will not cause an increase in the distance traveled by any robot (and will reduce it). $\square$

In the example, unlabeled reconfiguration corresponds to Fig. 6(a)→Fig. 6(b) and Fig. 6(f)→Fig. 6(e) (note that MRPP solutions are time-reversible). We simulated the process of anonymous reconfiguration for $m_1 = m_2 = 300$, i.e., on a 300×300 grids. For $\frac{1}{3}$ robot density, the actual number of steps, averaged over 100 random instances, is less than 5. We call configurations like Fig. 6(b)-(e), which have all robots concentrated vertically or horizontally in the middle of the 3×3 cells, *centered balanced configurations* or simply *centered configurations*. Completing the first phase requires solving two unlabeled MRPP problems [54, 32], easily doable in polynomial time.

In the second phase, RTA is applied with a highway heuristic to get us from Fig. 6(b) to Fig. 6(e), transforming between vertical centered configurations and horizontal centered configurations. To do so, RTA is applied (e.g., to Fig. 6(b) and (e)) to obtain two intermediate configurations (e.g., Fig. 6(c) and (d)). To go between these configurations, e.g., Fig. 6(b)→Fig. 6(c), we apply a heuristic by moving robots that need to be moved out of a 3×3 cell to the two sides of the middle columns of Fig. 6(b), depending on their target direction. If we do this consistently, after moving robots out of the middle columns, we can move all robots to their desired goal 3×3 cell without stopping nor collision. Once all robots are in the correct 3×3 cells, we can convert the balanced configuration to a centered configuration in at most 3 steps, which is necessary for carrying out the next simulated row/column shuffle. Adding things up, we can simulate a shuffle operation using no more than $m + 5$ steps where $m = m_1$ or m_2. The efficient simulated shuffle leads to low makespan MRPP routing algorithms. It is clear that all operations take polynomial time; a precise running time is given at the end of this subsection.

Theorem 10 (Makespan Upper Bound for Random MRPP, $\leq \frac{1}{3}$ Density). *For random MRPP instances on an $m_1 \times m_2$ grid, where $m_1 \geq m_2$ are multiples of three, for $n \leq \frac{m_1 m_2}{3}$ robots, an $m_1 + 2m_2 + o(m_1)$ makespan solution can be computed in polynomial time, with high probability.*

Proof. By Proposition 9, anonymous reconfiguration requires distance $o(m_1)$ with high probability. By Theorem 1 from [53], this implies that a plan can be obtained for anonymous reconfiguration that requires $o(m_1)$ makespan. For the second phase of MRPP resolution with Rubik Table and highway heuristics, by Theorem 1, we need to perform m_1 parallel row shuffles with row width of m_2, followed by m_2 parallel column shuffles with column width of m_1, followed by another m_1 parallel row shuffles with row width of m_2. Simulating these shuffles require $m_1 + 2m_2 + O(1)$ steps. All together, a makespan of $m_1 + 2m_2 + o(m_1)$ is required, with very high probability. $\square$

Contrasting Theorem 10 and Proposition 6 yields

Theorem 11 (Makespan Optimality for Random MRPP, $\leq \frac{1}{3}$ Density). *For random MRPP instances on an $m_1 \times m_2$ grid, where $m_1 \geq m_2$ are multiples of three, for $n = cm_1 m_2$ robots with $c \leq \frac{1}{3}$, as $m_1 \to \infty$, a $(1 + \frac{m_2}{m_1 + m_2})$ makespan optimal solution can be computed in polynomial time, with high probability.*

Since $m_1 \geq m_2$, $1 + \frac{m_2}{m_1 + m_2} \in (1, 1.5]$. In other words, in polynomial running time, RTH achieves $(1 + \delta)$ asymptotic makespan optimality for $\delta \in (0, 0.5]$, with high probability.

From the analysis so far, if m_1 and/or m_2 are not multiples of 3, it is clear that all results in this subsection continue to hold for robot density $\frac{1}{3} - \frac{(m_1 \mod 3)(m_2 \mod 3)}{m_1 m_2}$, which is arbitrarily close to $\frac{1}{3}$ for large m_1. It is also clear that the same can be said for grids with certain patterns of regularly distributed obstacles (Fig. 1(b)), i.e.,

Corollary 12 (Random MRPP, $\frac{1}{9}$ Obstacle and $\frac{2}{9}$ Robot Density). *For random MRPP instances on an $m_1 \times m_2$ grid, where $m_1 \geq m_2$ are multiples of three and there is an obstacle at coordinates $(3k_1 + 2, 3k_2 + 2)$ for all applicable k_1 and k_2, for $n = cm_1 m_2$ robots with $c \leq \frac{2}{9}$, a solution can be computed in polynomial time that has makespan $m_1 + 2m_2 + o(m_1)$ with high probability. As $m_1 \to \infty$, the solution approaches $1 + \frac{m_2}{m_1 + m_2}$ optimal, with high probability.*

Arbitrary MRPP instances. We now examine applying RTH to arbitrary MRPP instances under $\frac{1}{3}$ robot density. If an MRPP instance is arbitrary, all that changes to RTH is the makespan it takes to complete the anonymous reconfiguration phase. On an $m_1 \times m_2$ grid, by computing a matching, it is straightforward to show that it takes no more than $m_1 + m_2$ steps to complete the anonymous reconfiguration phase, starting from an arbitrary start configuration. Since two executions of anonymous reconfiguration are needed, this adds $2(m_1 + m_2)$ additional makespan. Therefore, we have

Theorem 13 (Arbitrary MRPP, $\leq \frac{1}{3}$ Density). *For arbitrary MRPP instances on an $m_1 \times m_2$ grid, $m_1 \geq m_2$, for $n \leq \frac{m_1 m_2}{3}$ robots, a $3m_1 + 4m_2 + o(m_1)$ makespan solution can be computed in polynomial time. This implies that, for $n = cm_1 m_2 \leq \frac{m_1 m_2}{3}$ robots, a polynomial time can compute an asymptotic $3 + \frac{m_2}{m_1 + m_2}$ makespan optimal solution, with high probability.*

We now give the running time of RTM and RTH.

Proposition 14 (Running Time, RTH). *For $n \leq \frac{m_1 m_2}{3}$ robots on an $m_1 \times m_2$ grid, RTH runs in $O(nm_1^2 m_2)$ time.*

Proof. The running time of RTM and RTH are dominated by the matching computation and solving anonymous MRPP. The matching part takes $O(m_1^2 m_2)$ in deterministic time or $O(m_1 m_2 \log m_1)$ in expected time [14]. Anonymous MRPP may be tackled using the max-flow algorithm [13] in $O(nm_1 m_2 T) = O(nm_1^2 m_2)$ time, where $T = O(m_1 + m_2)$ is the expansion time horizon of a time-expanded graph that allows a routing plan to complete. $\square$

B. Reducing Makespan via Optimizing Matching

Based on RTA, RTH has three simulated shuffle phases. Eventually, the makespan is dominated by the robot that needs the longest time to move, as a sum of moves for the robot in all three phases. As a result, the optimality of Rubik Table methods is determined by the first preparation phase. The matchings determine the intermediate states in all three phases. Finding arbitrary perfect matchings is fast but the process can be improved to reduce the overall makespan.

For improving matching, we propose two heuristics; the first is based on *integer programming* (IP). We create binary variables $\{x_{ri}\}$ where r represents the row number and i the robot. robot i is assigned to row r if $x_{ri} = 1$. Define single robot cost as $C_{ri}(\lambda) = \lambda|r - s_i.x| + (1 - \lambda)|r - g_i.x|$. We optimize the makespan lower bound of the first phase by letting $\lambda = 0$ or the third phase by letting $\lambda = 1$. The objective function and constraints are given by

$$\max_{r,i}\{C_{ri}(\lambda = 0)x_{ri}\} + \max_{r,i}\{C_{ri}(\lambda = 1)x_{ri}\} \quad (1)$$

$$\sum_r x_{ri} = 1, \text{for each robot } i \quad (2)$$

$$\sum_{g_i.y=t} x_{ri} \leq 1, \text{for each row } r \text{ and each color } t \quad (3)$$

$$\sum_{s_i.y=c} x_{ri} = 1, \text{for each column } c \text{ and each row } r \quad (4)$$

Eq. (1) is the summation of makespan lower bound of the first phase and the third phase. Note that the second phase cannot be improved through optimizing the matching. Eq. (2) requires that robot i be only present in one row. Eq. (3) specifies that each row should contain robots that have different goal columns. Eq. (4) specifies that each vertex (r, c) can only be assigned to one robot. The IP model represents a general assignment problem which is NP-hard in general. It has limited scalability but provides a way to evaluate how optimal the matching could be in the limit.

A second matching heuristic we developed is based on *linear bottleneck assignment (LBA)* [5], which takes polynomial time. LBA differs from the IP heuristic in that the bipartite graph is weighted. For the matching assigned to row r, the edge weight of the bipartite graph is computed greedily. If column c contains robots of color t, we add an edge (c, t) and its edge cost is

$$C_{ct} = \min_{g_i.y=t} C_{ri}(\lambda = 0) \quad (5)$$

We choose $\lambda = 0$ to optimize the first phase. Optimizing the third phase ($\lambda = 1$) would give similar results. After constructing the weighted bipartite graph, an $O(\frac{m_1^{2.5}}{\log m_1})$ LBA algorithm [5] is applied to get a minimum bottleneck cost matching for row r. Then we remove the assigned robots and compute the next minimum bottleneck cost matching for next row. After getting all the matchings $\mathcal{M}_r$, we can further use LBA to assign $\mathcal{M}_r$ to a different row r' to get a smaller makespan lower bound. The cost for assigning matching $\mathcal{M}_r$ to row r' is defined as

$$C_{\mathcal{M}_r r'} = \max_{i \in \mathcal{M}_r} C_{r'i}(\lambda = 0) \quad (6)$$

The total time-complexity of using LBA heuristic for matching is $O(\frac{m_1^{3.5}}{\log m_1})$.

We denote RTH with IP and LBA heuristics as RTH-IP and RTH-LBA, respectively. We mention that RTM, which uses the line swap motion primitive, can also benefit from these heuristics to re-assign the goals within each group. This can lower the bottleneck path length and improve the optimality.

V. NEAR-OPTIMALLY SOLVING MRPP AT HALF ROBOT DENSITY

The key design philosophy behind RTM and RTH is to effectively simulate row/column shuffles. With this in mind, we further explored the case of $\frac{1}{2}$ robot density. Using a more sophisticated shuffle routine, $\frac{1}{2}$ robot density can be supported while retaining most of the guarantees for the $\frac{1}{3}$ density setting; obstacles are no longer supported.

To best handle $\frac{1}{2}$ robot density, we employ a new shuffle routine called *linear merge*, based on merge sort, and denote the resulting algorithm as Rubik Table with Linear Merge heuristics or RTLM. The basic idea behind linear merge (shuffle) is straightforward: for m robots on a $2 \times m$ grid, we iteratively sort the robots first on 2×2 grids, then 2×4 grids, and so on, much like how merge sort works. An illustration of the process on a 2×8 grid is shown in Fig. 7.
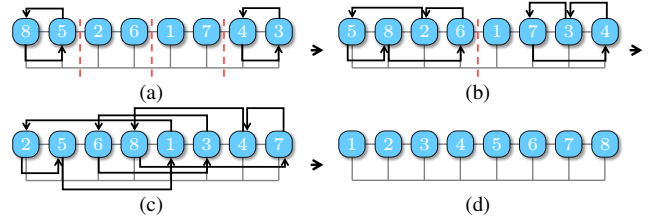

Fig. 7: A demonstration of the linear merge shuffle primitive on a 2×8 grid. Robots going to the left always use the upper channel while robots going to the right always use the lower channel.

We now show that linear merge is always feasible and has the desired properties.

Lemma 15 (Properties of Linear Merge). *On a $2 \times m$ grid, m robots, starting on the first row, can be arbitrarily ordered using $m + o(m)$ steps. The motion plan can be computed in polynomial time.*

Proof. We first show *feasibility*. The procedure takes $\lceil \log m \rceil$ phases; in a phase, let us denote a section of the $2 \times m$ grid where robots are treated together as a *block*. For example, the left 2×4 grid in Fig. 7(b) is a block. It is clear that the first phase, involving up to two robots per block, is feasible (i.e., no collision). Assuming that phase k is feasible, we look at phase $k + 1$. We only need to show that the procedure is feasible on one block of length up to 2^{k+1}. For such a block, the left half block of length up to 2^k is already fully sorted as desired, e.g., in increasing order from left to right. For the $k + 1$ phase, all robots in the left half block may only stay in place or move to the right. For these robots that stay, they must be all at the leftmost positions of the half block and will not block motions

of any other robot. For the robots that do need to move to the right, their relative orders do not need to change, and therefore will not cause collisions among themselves. Because these robots that move in the left half block will move down on the grid by one edge, they will not interfere with any robot from left from the right half block. Because the same arguments hold for the right half block (except the direction change), the overall process of merging a block occurs without collision.

Next, we examine the *makespan*. For any single robot r, at phase k, suppose it belongs to block b and block b is to be merged with block b'. It is clear that the robot cannot move more than $len(b') + 2$ steps, where $len(b')$ is the number of columns of b' and the 2 extra steps may be incurred because the robot needs to move down and then up the grid by one edge. This is because any move that r needs to do is to allow robots from b' to move toward b. Because there are no collisions in any phase, adding up all the phases, no robot moves more than $m + 2(\log m + 1) = m + o(m)$ steps.

Finally, it is clear that the merge sort-like linear merge shuffle primitive runs in $O(m \log m)$ time since it is a standard divide-and-conquer routine with $\log m$ phases. $\square$

With linear merge, the asymptotic properties of RTH for $\frac{1}{3}$ robot density mostly carries over to RTLM.

Theorem 16 (Random MRPP, $\frac{1}{2}$ Robot Density). *For random* MRPP *instances on an $m_1 \times m_2$ grid, where $m_1 \geq m_2$ are multiples of two, for $\frac{m_1 m_2}{3} \leq n \leq \frac{m_1 m_2}{2}$ robots, a solution can be computed in polynomial time that has makespan $m_1 + 2m_2 + o(m_1)$ with high probability. As $m_1 \to \infty$, the solution approaches an optimality of $1 + \frac{m_2}{m_1 + m_2} \in (1, 1.5]$, with high probability.*

VI. SIMULATION EXPERIMENTS

In this section, we evaluate Rubik Table based algorithms and compare them with fast and near-optimal solvers, ECBS (w=1.5) [3] and DDM [22]. These two methods are, to our knowledge, two of the fastest near-optimal solvers for MRPP. We considered a state-of-the-art polynomial algorithm, push-and-swap [31], which gave fairly suboptimal results; the makespan optimality ratio is often above 100 for densities we examine. We also tested prioritized methods, e.g., [34, 36, 44], which faced significant difficulties in resolving deadlocks. Given the limited relevance and considering the amount of results we are presenting, we omit these methods in our comparison.

All experiments are performed on an Intel® Core™ i7-9700 CPU at 3.0GHz. Each data point is an average over 20 runs on randomly generated instances, unless otherwise stated. A running time limit of 300 seconds is imposed over all instances. The optimality ratio is estimated as compared to conservatively estimated makespan lower bounds. The Rubik Table based algorithms are implemented in Python. The compared solvers are C++ based. As such, one can expect additional significant running time reductions from our algorithms implemented in C++. We choose Gurobi [20] as the mixed integer programming solver and ORtools [37] as the

max-flow solver. The video of the simulations can be found at https://youtu.be/aphCjWFwfss.

A. Optimality of RTM, RTLM, and RTH

We first evaluate the optimality achieved by RTM, RTLM, and RTH over randomly generated instances on their maximum designed robot density. That is, for RTM, the grid is fully occupied; for RTLM and RTH, the robot density is $\frac{1}{2}$ and $\frac{1}{3}$, respectively. We test over three $m_1 : m_2$ ratios: $1 : 1$, $3 : 2$, and $5 : 1$. The result is plotted in Fig. 8. Computation time is not listed (because we list the computation time later for RTH; the running times of RTM, RTLM, and RTH are similar). The optimality ratio is computed as the ratio between the solution makespan and the longest Manhattan distance between any pair of start and goal. Therefore, the estimate is an overestimate (i.e., the actual ratio may be lower/better).
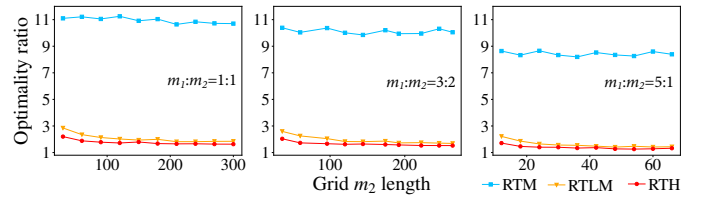

Fig. 8: Makespan optimality ratio for RTM, RTLM, and RTH for their maximum designed robot density, for different grid sizes and $m_1 : m_2$ ratios. We note that the largest problem has $90,000$ robots on a 300×300 grid.

We observe that RTM achieves 7–10.5+ makespan optimality ratio, which justifies the correctness of Theorem 7. Both RTLM and RTH achieve sub-2 optimality guarantee for most of the test cases, with result for RTH dropping below 1.5 on large grids. For all settings, as the grid size increases, there is a general trend of improvement of optimality across all methods/grid aspect ratios. This is due to two reasons: (1) the overhead in the shuffle operations becomes relatively smaller as grid size increases, and (2) with more robots, the makespan lower bound becomes closer to $m_1 + m_2$. Lastly, as $m_1 : m_2$ ratio increases, the optimality ratio improves as predicted. For many test cases, the optimality ratio for the RTH for $m_1 : m_2 = 5$ setting is around 1.3.

We note that the performance of RTLM and RTH on optimality can be further improved using the heuristics described in Sec. IV-B. For the rest of the evaluations, we focus on RTH and its variants with additional heuristics.

B. Evaluation and Comparative Study of RTH

1) Impact of grid size: For our first detailed comparative study of the performance of RTH, we set $m_1 : m_2 = 3 : 2$ and fix robot density at $\frac{1}{3}$. In Fig. 9, we compare the performance of ECBS[3], DDM[22], RTH, RTH-IP, and RTH-LBA, in terms of computation time and optimality ratio.

Despite the less efficient Python-based implementation, RTM, RTH, and RTH-LBA are faster than ECBS and DDM, due to their low polynomial running time guarantees. RTH and RTH-LBA can solve very large instances, e.g., on 450×300 grids with $45,000$ robots in about 100 seconds while neither
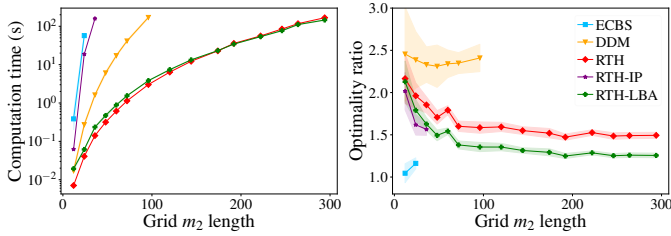
Fig. 9: Computation time and optimality ratios on $m_1 \times m_2$ grids of varying sizes with $m_1 : m_2 = 3 : 2$ and robot density at $\frac{1}{3}$. One standard deviation is shown as shaded regions.

Fig. 10: Computation time and optimality ratios on 180×120 grids with varying robot density.

ECBS nor DDM can. ECBS stopped working after $m_2 = 30$, though it shows better optimality on problems it can solve. DDM could handle up to $m_2 = 90$ but demonstrated poor optimality under the $\frac{1}{3}$ density setting. The optimality ratio of RTH and RTH-LBA improves as the graph size increases, as predicted by our theoretical analysis. The optimality ratios of RTH and RTH-LBA reach as low as 1.49 and 1.26, respectively, agreeing with the ratio predicted by Theorem 11; here, the high probability asymptotic ratio is $1 + \frac{m_2}{m_1 + m_2} = 1.4$. RTH-LBA is able to do better than 1.4 because of the LBA heuristic. RTH-IP does slightly better on optimality in comparison to RTH and RTH-LBA, but its scalability is limited. On the other hand, RTH-LBA does nearly as well on optimality and remains competitive in terms of running time in comparison to RTH. As a consequence, we do not include further evaluation of RTH-IP.

For each method, the one standard deviation range is also shown in the figure. Because RTH and RTH-LBA are mostly deterministic and there are many robots, the change in optimality across different instances is small. We omit the inclusion of standard deviations from other plots as they are mostly similar in other tested settings.

We mention that we also evaluated ECBS with temporal splitting heuristics [19], which did not show significant difference in comparison to ECBS at the density we tried; so we did not include it here. We also evaluated push-and-swap [31], which runs fast but yields very poor optimality ratios (> 100 for many instances). We further evaluated prioritized methods, e.g., [36], which faced significant difficulties in resolving deadlocks. Given these, we did not include results from these methods in our comparatively study.

2) Impact of robot density: Next, we experiment the impact of different robot density on a 180×120 grid (Fig. 10). At density $\frac{1}{3}$, there are up to $7,200$ robots. For densities below $\frac{1}{3}$, we add "virtual robots" when the matching step is performed. ECBS does not appear because it cannot solve problems at this scale within 300 seconds.

RTH and RTH-LBA both return solutions around 10s on this graph for all instances. Both computation time and optimality ratio of DDM grow as the robot density increases, while the robot density has little impact on RTH and RTH-LBA. At lower density, DDM demonstrates better optimality. In environments with high densities, however, robots are highly coupled which causes more conflicts for structurally-agnostic approaches like DDM (and ECBS). RTH and RTH-
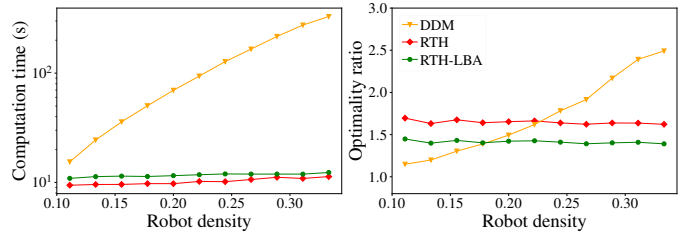
LBA show improved optimality as the density increases, reaching 1.4, mostly due to the makespan of the instances getting larger.

3) Handling obstacles: RTH can also handle scattered obstacles and are especially suitable for cases where obstacles are regularly distributed. For instance, problems with underlying graphs like that in Fig. 1(b), where each 3×3 cell has a hole in the middle, can be natively solved without performance degradation. Such settings find real-world applications in parcel sorting facilities in large warehouses [50, 29]. For this parcel sorting setup, we fix the robot density at $\frac{2}{9}$ and test ECBS, DDM, RTH and RTH-LBA on graphs with varying sizes. The results are shown in Fig 11. Note that DDM can only apply when there is no narrow passage. So we added additional "borders" to the map to make it solvable for DDM. The results are similar as earlier ones; RTH and RTH-LBA run very fast and produce high-quality solutions, with conservatively estimated optimality ratio approaching 1.27.
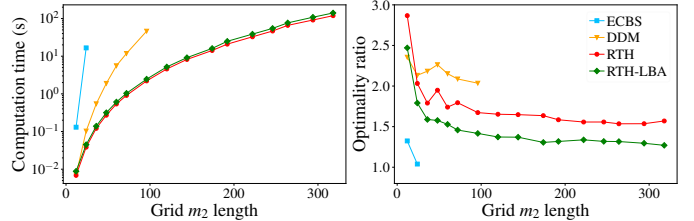

Fig. 11: Computation time and optimality ratios on environments of varying sizes with regularly distributed obstacles at $\frac{1}{9}$ density and robots at $\frac{2}{9}$ density. $m_1 : m_2 = 3 : 2$.

4) Impact of grid aspect ratios: In this section, we fix $m_1 m_2 = 90000$ and vary the $m_2 : m_1$ ratio between 0 (nearly one dimensional) and 1 (square grids). We evaluated four algorithms, two of which are RTH and RTH-LBA. Now recall that RTP on an $m_1 \times m_2$ table can also be solved using $2m_2$ column shuffles and m_1 row shuffles. Adapting RTH and RTH-LBA with $m_1 + 2m_2$ shuffles gives the other two variants which we denote as RTH-LL and RTH-LBA-LL respectively, with "LL" suggesting two sets of longer shuffles are performed (each set of column shuffle work with columns of length m_1). The result is summarized in Fig. 12.

Interestingly but not surprisingly, the result clearly demonstrates the trade-offs between computation effort and solution optimality. RTH and RTH-LBA achieve better optimality ratio in comparison to RTH-LL and RTH-LBA-LL but require more computation time. Notably, the optimality ratios
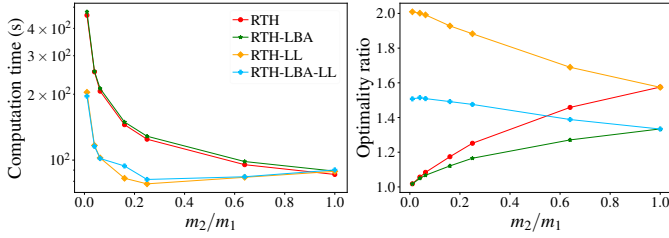
Fig. 12: Computation time and optimality ratios on rectangular grids of varying aspect ratio and $\frac{1}{3}$ robot density.

for RTH and RTH-LBA are very close to 1 when $m_2 : m_1$ is close to 0. Because the LBA heuristic aims to reduce the possible makespan of the first or third phase of RTA, as m_1/m_2 increases, the optimality gap between LBA and non-LBA variants of RTH increases, which clearly demonstrates the advantage of the LBA heuristic.

C. Special Patterns

Besides random start and goal settings, we also test RTH-LBA on many "special" instances; two are presented here (Fig. 13). For both settings, $m_1 = m_2$. In the first, the "squares" setting, robots form concentric square rings and each robot and its goal are centrosymmetric. In the second, the "blocks" setting, the grid is divided into smaller square blocks (not necessarily 3×3) containing the same number of robots. robots from one block need to move to another random chosen block. RTH-LBA achieves optimality that is fairly close to 1.0 in the square setting and 1.7 in the block setting. The computation time is similar to that of Fig. 11; ECBS does well on optimality but scales poorly (only works on 30×30 grids). For some reason, DDM does very poorly on optimality and is not included.
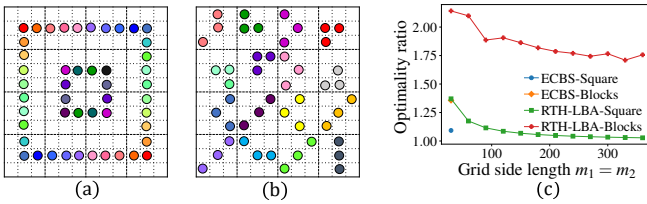

Fig. 13: (a) An illustration of the "squares" setting. (b) An illustration of the "blocks" setting. (c) Optimality ratios for the two settings for ECBS and RTH-LBA.

VII. CONCLUSION AND DISCUSSION

In this study, we propose to apply Rubik Tables [48] to solving MRPP. A basic adaptation of RTA, with a more efficient line shuffle routine, enables solving MRPP on grids at maximum robot density, in polynomial time, with previously unachievable optimality guarantee. Then, combining RTA, a highway heuristic, and additional matching heuristics, we obtain novel polynomial time algorithms that are provably asymptotically $1 + \frac{m_2}{m_1 + m_2}$ makespan-optimal on $m_1 \times m_2$ grids with up to $\frac{1}{3}$ robot density, with high probability. Similar guarantees are also achieved with the presence of obstacles and at robot density up to one half. In practice, our methods can

solve problems on graphs with over 10^5 number of vertices and 4.5×10^4 robots to 1.26 makespan-optimal (which can be better with larger $m_1 : m_2$ ratio). To our knowledge, no previous MRPP solvers provide dual guarantees on low-polynomial running time and practical optimality.

Our study opens the door for many follow up research directions; we discuss a few here.

New line shuffle routines. Currently, RTLM and RTH only use two/three rows to perform a simulated row shuffle. Among other restrictions, this requires that the sub-grids used for performing simulated shuffle be well-connected (i.e. obstacle-free or the obstacles are regularly spaced so that there are at least two rows that are not blocked by static obstacles in each motion primitive to simulate the shuffle). Using more rows or even irregular rows in a simulated row shuffle, it is potentially possible to accommodate larger obstacles and/or support density higher than one half.

Better optimality at lower robot density. It is interesting to examine whether further optimality gains can be realized at lower robot density settings, e.g., $\frac{1}{9}$ density or even lower, which are still highly practical. We hypothesize that this can be realized by somehow merging the different phases of RTA so that some unnecessary robot travel can be eliminated, after computing an initial plan.

Consideration of more realistic robot models. The current study assumes a unit-cost model in which a robot takes a unit amount of time to travel a unit distance and allow turning at every integer time step. In practice, robots will need to accelerate/decelerate and also need to make turns. Turning can be especially problematic and can cause significant increase in plan execution time, if the original plan is computed using the unit-cost model mentioned above. We note that RTH returns solutions where robots move in straight lines most of the time, which is advantageous in comparison to all existing MRPP algorithms, such as ECBS and DDM, which have large number of directional changes in their computed plans. it would be interesting to see whether the performance of RTA based MRPP algorithms will further improve as more realistic robot models are adapted.

Life-long MRPP settings. Currently, out RTA based MRPP solves are limited to a static setting whereas e-commerce applications of multi-robot motion planning often require solving life-long setting [33]. The metric for evaluating life-long MRPP is often the *throughput*, namely the number of goals reached per time step. We note that RTH also provides optimality guarantees for such settings, e.g., for the setting where $m_1 = m_2 = m$, we have

Proposition 17 (Bound for Random Life-Long MRPP). *The direct application of* RTH *to large-scale life-long* MRPP *on square grids yields an optimality ratio of* $\frac{2}{9}$ *on throughput.*

Proof. We may solve life-long MRPP using RTH in *batches*. For each batch with n robots, RTH takes about $3m$ steps; the throughput is then $\mathcal{T}_{RTH} = \frac{n}{3m}$. As for the lower bound estimation of the throughput, the expected Manhattan distance in an $m \times m$ square, ignoring inter-robot collisions, is $\frac{2m}{3}$.

Therefore, the lower bound throughput for each batch is $\mathcal{T}_{lb} = \frac{3n}{2m}$. The asymptotic optimality ratio is $\frac{\mathcal{T}_{RTH}}{\mathcal{T}_{lb}} = \frac{2}{9}$. $\qquad\square$

The $\frac{2}{9}$ estimate is fairly conservative because RTH supports much higher robot densities not supported by known life-long MRPP solvers. Therefore, it appears very promising to develop optimized Rubik Table inspired algorithms for solving life-long MRPP problems.

REFERENCES

[1] Warehouse automation market with post-pandemic (covid-19) impact by technology, by industry, by geography - forecast to 2026. https://www.researchandmarkets.com/r/s6basv. Accessed: 2022-01-05.

[2] Jacopo Banfi, Nicola Basilico, and Francesco Amigoni. Intractability of time-optimal multirobot path planning on 2d grid graphs with holes. *IEEE Robotics and Automation Letters*, 2(4):1941–1947, 2017.

[3] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Seventh Annual Symposium on Combinatorial Search*, 2014.

[4] Dina Bitton, David J DeWitt, David K Hsaio, and Jaishankar Menon. A taxonomy of parallel sorting. *ACM Computing Surveys (CSUR)*, 16(3):287–318, 1984.

[5] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment problems: revised reprint*. SIAM, 2012.

[6] Fernando Alfredo Auat Cheein and Ricardo Carelli. Agricultural robotics: Unmanned robotic service units in agricultural tasks. *IEEE industrial electronics magazine*, 7(3):48–58, 2013.

[7] Mehul Damani, Zhiyao Luo, Emerson Wenzel, and Guillaume Sartoretti. Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robotics and Automation Letters*, 6(2):2666–2673, 2021.

[8] Boris De Wilde, Adriaan W Ter Mors, and Cees Witteveen. Push and rotate: a complete multi-agent pathfinding algorithm. *Journal of Artificial Intelligence Research*, 51:443–492, 2014.

[9] Ashutosh Dekhne, Greg Hastings, John Murnane, and Florian Neuhaus. Automation in logistics: Big opportunity, bigger uncertainty. *McKinsey Q*, pages 1–12, 2019.

[10] Erik D Demaine, Sándor P Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM Journal on Computing*, 48(6):1727–1762, 2019.

[11] Esra Erdem, Doga Gizem Kisa, Umut Oztok, and Peter Schüller. A general formal framework for pathfinding problems with multiple agents. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[12] Michael Erdmann and Tomas Lozano-Perez. On multiple moving objects. *Algorithmica*, 2(1):477–521, 1987.

[13] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

[14] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings in o(n\logn) time in regular bipartite graphs. *SIAM Journal on Computing*, 42(3):1392–1404, 2013.

[15] Meir Goldenberg, Ariel Felner, Roni Stern, Guni Sharon, Nathan Sturtevant, Robert C Holte, and Jonathan Schaeffer. Enhanced partial expansion a. *Journal of Artificial Intelligence Research*, 50:141–187, 2014.

[16] Oded Goldreich. 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*, pages 1–5. Springer, 2011.

[17] Teng Guo and Jingjin Yu. Sub-1.5 time-optimal multi-robot path planning on grids in polynomial time. *arXiv preprint arXiv:2201.08976*, 2022.

[18] Teng Guo, Shuai D. Han, and Jingjin Yu. Spatial and temporal splitting heuristics for multi-robot motion planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8009–8015, 2021. doi: 10.1109/ICRA48506.2021.9561899.

[19] Teng Guo, Shuai D Han, and Jingjin Yu. Spatial and temporal splitting heuristics for multi-robot motion planning. In *IEEE International Conference on Robotics and Automation*, 2021.

[20] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL https://www.gurobi.com.

[21] Philip Hall. On representatives of subsets. In *Classic Papers in Combinatorics*, pages 58–62. Springer, 2009.

[22] Shuai D Han and Jingjin Yu. Ddm: Fast near-optimal multi-robot path planning using diversified-path and optimal sub-problem solution database heuristics. *IEEE Robotics and Automation Letters*, 5(2):1350–1357, 2020.

[23] Shuai D Han, Edgar J Rodriguez, and Jingjin Yu. Sear: A polynomial-time multi-robot path planning algorithm with expected constant-factor optimality guarantee. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[24] Wolfgang Hönig, James A Preiss, TK Satish Kumar, Gaurav S Sukhatme, and Nora Ayanian. Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics*, 34(4):856–869, 2018.

[25] John E Hopcroft, Jacob Theodore Schwartz, and Micha Sharir. 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, 1984.

[26] D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation

groups, and applications. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 241–250, 1984.

[27] Edward Lam, Pierre Le Bodic, Daniel Damir Harabor, and Peter J Stuckey. Branch-and-cut-and-price for multi-agent pathfinding. In *IJCAI*, pages 1289–1296, 2019.

[28] Tom Leighton and Peter Shor. Tight bounds for minimax grid matching with applications to the average case analysis of algorithms. *Combinatorica*, 9(2):161–187, 1989.

[29] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W Durham, TK Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *AAMAS*, pages 1898–1900, 2020.

[30] Jiaoyang Li, Wheeler Ruml, and Sven Koenig. Eecbs: A bounded-suboptimal search for multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[31] Ryan J Luna and Kostas E Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[32] Hang Ma and Sven Koenig. Optimal target assignment and path finding for teams of agents. In *AAMAS*, 2016.

[33] Hang Ma, Jiaoyang Li, T. K. S. Kumar, and Sven Koenig. Lifelong multi-agent path finding for online pickup and delivery tasks. In *AAMAS*, 2017.

[34] Hang Ma, Daniel Harabor, Peter J Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7643–7650, 2019.

[35] Robert Mason. Developing a profitable online grocery logistics business: Exploring innovations in ordering, fulfilment, and distribution at ocado. In *Contemporary Operations and Logistics*, pages 365–383. Springer, 2019.

[36] Keisuke Okumura, M. Machida, X. Défago, and Yasumasa Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. In *IJCAI*, 2019.

[37] Laurent Perron and Vincent Furnon. Or-tools. URL https://developers.google.com/optimization/.

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

[39] James A Preiss, Wolfgang Hönig, Gaurav S Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017.

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

[41] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

[42] Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195:470–495, 2013.

[43] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[44] David Silver. Cooperative pathfinding. *Aiide*, 1:117–122, 2005.

[45] Roni Stern, Nathan R Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Satish Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[46] Pavel Surynek. An optimization variant of multi-robot path planning is intractable. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

[47] Pavel Surynek. Towards optimal cooperative path planning in hard setups through satisfiability solving. In *Pacific Rim International Conference on Artificial Intelligence*, pages 564–576. Springer, 2012.

[48] Mario Szegedy and Jingjin Yu. On rearrangement of items stored in stacks. In *The 14th International Workshop on the Algorithmic Foundations of Robotics*, 2020.

[49] Glenn Wagner. Subdimensional expansion: A framework for computationally tractable multirobot path planning. 2015.

[50] Qian Wan, Chonglin Gu, Sankui Sun, Mengxia Chen, Hejiao Huang, and Xiaohua Jia. Lifelong multi-agent path finding in a dynamic environment. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 875–882. IEEE, 2018.

[51] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008.

[52] Jingjin Yu. Intractability of optimal multirobot path planning on planar graphs. *IEEE Robotics and Automation Letters*, 1(1):33–40, 2015.

[53] Jingjin Yu. Constant factor time optimal multi-robot routing on high-dimensional grids. *2018 Robotics: Science and Systems*, 2018.

[54] Jingjin Yu and M. LaValle. Distance optimal formation control on graphs with a tight convergence time guarantee. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 4023–4028, 2012. doi: 10.1109/CDC.2012.6426233.

[55] Jingjin Yu and Steven M LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[56] Jingjin Yu and Steven M LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32

(5):1163–1177, 2016.