Connected Coordinated Motion Planning with Bounded Stretch

Sándor P. Fekete[1], Phillip Keldenich[1], Ramin Kosfeld[1], Christian Rieck[1], and Christian Scheffer[1]

1 **Departement of Computer Science, TU Braunschweig, Germany**
{s.fekete, p.keldenich, r.kosfeld, c.rieck, c.scheffer}@tu-bs.de

Abstract

We consider the problem of coordinated motion planning for a swarm of simple, identical robots: From a given start grid configuration of robots, we need to reach a desired target configuration via a sequence of parallel, continuous, collision-free robot motions, such that the set of robots stays connected at all times. The objective is to minimize the *makespan* of the motion schedule, i.e., to reach the new configuration in a minimum amount of time. We show that this problem is NP-hard, even for deciding whether a makespan of 2 can be achieved, while it is possible to check in polynomial time whether a makespan of 1 can be achieved. We also provide a constant-factor approximation for *fat* configurations. Our algorithm achieves a *constant stretch factor*: If mapping the start configuration to the target configuration requires a maximum Manhattan distance of d, then the total duration of our overall schedule is $\mathcal{O}(d)$, which is optimal up to constant factors.

1 Introduction

Consider a connected configuration of objects, e.g., a swarm of mobile robots, which needs to be transformed into a desired target configuration by a sequence of parallel, continuous, collision-free motions that keeps the overall arrangement connected at all times. Such problems occur in many contexts requiring relocation of autonomous agents; the connectivity constraint arises naturally in many physical scenarios, e.g., for reconfigurable matter in space. How can we coordinate the corresponding motion, such that a desired target configuration is reached within a minimum amount of time, called *makespan*, without losing connectivity? As it turns out, this problem is provably hard, even in relatively simple cases. We present methods that, provided sufficient fatness of the start and target configuration, realize *constant stretch*: If mapping the start configuration C_s to the target configuration C_t requires a maximum Manhattan distance of d, then the total duration of our overall schedule is $\mathcal{O}(d)$.

Our Contributions. We provide new results for questions arising from efficiently reconfiguring a connected, unlabeled swarm of robots from a given start configuration C_s into a desired target configuration C_t, aiming for minimizing the overall makespan, i.e., the total time required for running the full schedule, and maintaining connectivity in each step.

- There exists a polynomial time algorithm to decide whether there is a schedule with a makespan of 1 that transforms C_s into C_t, see Theorem 1.
- It is NP-hard to decide whether there is a schedule with a makespan of 2 that transforms C_s into C_t, see Theorem 2. This implies NP-hardness of approximating the minimum makespan within a constant of $(\frac{3}{2} - \varepsilon)$, for any $\varepsilon > 0$, see Corollary 3.
- There is a constant c such that for any pair of start and target configurations with fatness of at least c, a schedule with constant stretch can be computed in polynomial time, see Theorem 4. This implies that there is a constant-factor approximation for the problem of computing schedules with minimal makespan restricted to pairs of start and target configuration with a fatness of at least c, see Corollary 8.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Related Work. Coordinating the motion of many agents plays a central role when dealing with large numbers of moving robots, vehicles, aircraft, or people. Basic questions arise in many applications, such as ground swarm robotics [8, 9], aerial swarm robotics [2, 12], air traffic control [3], and vehicular traffic networks [6, 10], and goes back to work by Schwartz and Sharir [11]. Most previous work has largely focused on sequential schedules, where one robot moves at a time, with objectives such as minimizing the number of moves. In practice, however, robots usually move simultaneously, so we desire a parallel motion schedule, with the objective of minimizing the makespan. In recent work [1, 4, 5], we provide several fundamental insights into these problems of coordinated motion planning for the scenario with labeled robots without a connectivity constraint.

2 Preliminaries

We consider n unlabeled *robots* at integer grid positions, inducing a grid graph. A configuration C is *c-fat*, if for any vertex v there is a $(c \times c)$-block $B \subset C$, such that $v \in B$.

A robot can move in discrete time steps by changing its location from a grid position v to an adjacent grid position w; this is denoted by $v \to w$. Two moves $v_1 \to w_1$ and $v_2 \to w_2$ are *collision-free* if $v_1 \neq v_2$ and $w_1 \neq w_2$. A *transformation* between two configurations $C_1 = \{v_1, \ldots, v_n\}$ and $C_2 = \{w_1, \ldots, w_n\}$ is a set of collision-free moves $\{v_i \to w_i \mid i = 1, \ldots, n\}$. For $M \in \mathbb{N}$, a *schedule* is a sequence $C_1 \rightrightarrows C_{M+1} := C_1 \to \cdots \to C_{M+1}$ of transformations, with a *makespan* of M. A *stable* schedule, $C_s \rightrightarrows_\chi C_t := C_s \to_\chi \cdots \to_\chi C_t$ between the start configuration C_s and the target configuration C_t uses only connected configurations. A *bottleneck matching* between the vertices of C_s and C_t minimizes the maximal Manhattan distance d, called the *diameter* of (C_s, C_t), between two matched vertices from C_s and C_t. The *stretch (factor)* of a (stable) schedule is the ratio between the makespan M and the diameter d.

3 Makespan 1 and 2

It can efficiently be decided whether a stable schedule $C_s \to_\chi C_t$ with a makespan of 1 exist.

▶ **Theorem 1.** *For two configurations C_s and C_t, each with n vertices, it can be decided in polynomial time whether there is a schedule with a makespan of 1 transforming C_s into C_t.*

To decide this, it suffices to compute a maximum matching in the bipartite graph $G = (V_s \cup V_t, E)$ consisting of vertices for all occupied positions in both configurations, and edges between vertices if their respective positions are adjacent or identical. It is easy to see that there is a schedule with a makespan of 1 if and only if G admits a perfect matching. This can be checked with the method of Hopcraft and Karp in $\mathcal{O}(n^{5/2})$ time [7].

However, even for a makespan of 2, the same problem becomes provably hard.

▶ **Theorem 2.** *For a pair of configurations C_s and C_t, each with n vertices, deciding whether there is a stable schedule with a makespan of 2 transforming C_s into C_t is NP-hard.*

The proof establishes a reduction from Planar Monotone 3SAT, which asks to decide whether we can satisfy a Boolean 3-CNF formula φ for which in each clause the literals are either all positive or all negative. For every instance φ of Planar Monotone 3SAT, we construct an instance I_φ, consisting of a start configuration C_s and a target configuration C_t, as indicated in Figure 2. In the figure, we use three differently colored squares to indicate occupied positions in the C_s (red), in C_t (dark cyan), and in both configurations (gray).

We can argue that there is a stable schedule transforming the start configuration into the target configuration with a makespan of 2, if and only if φ is satisfiable. In order to transform C_s into C_t, the separation gadgets (yellow) ensure that in the single intermediate configuration, all clause and helper gadgets (shades of blue) are disconnected from each other. Therefore, to satisfy the connectivity constraint, some robots of the variable gadget (light red) have to move in a very particular way, such that these robots ensure connections between the variable gadget and the clause gadgets. At the same time, we ensure that robots representing a variable can either connect this variable to their positive or to their negative literal containing clauses (otherwise the connectivity within the variable gadget would be broken); thus, these movements can be used to determine a valid assignment for φ.

Most of the gadgets are straightforward. Because we use the movements in the variable gadget to determine a variable assignment for φ, we briefly explain how this works.

Thus, consider the arrangement consisting of start and target configurations in Figure 1. Without loss of generality, we consider the situation in which in the single intermediate configuration robots representing the positive arm segment are connected to their respective bridges. Hence, at least one robot of this segment (i.e., $p_1, \ldots, p_\ell$) has to move up. Then the robots on X_i and A_i have unique target locations, i.e., Y_i and B_i, respectively. Given all these moves, the robot R_i has to move up to maintain connectivity. Because R_i cannot simultaneously maintain connectivity for $\overline{X_i}$ and $\overline{A_i}$, its movement can be used to determine the variable assignment for φ.

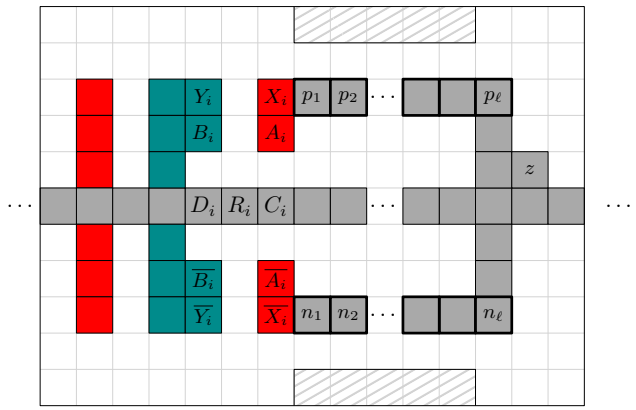

Figure 1 The variable gadget. The robot R_i is used to determine the variable assignment.

More technical details of the proof of Theorem 2 are omitted due to space constraints. As the proof of Theorem 2 shows that it is NP-hard to decide whether there is a stable schedule with a makespan of 2 transforming C_s into C_t, we obtain the following.

▶ **Corollary 3.** *It is NP-hard to compute for a pair of configurations C_s and C_t, each with n vertices, a stable schedule that transforms C_s into C_t within a constant of $(\frac{3}{2} - \varepsilon)$ (for any $\varepsilon > 0$) of the minimum makespan.*

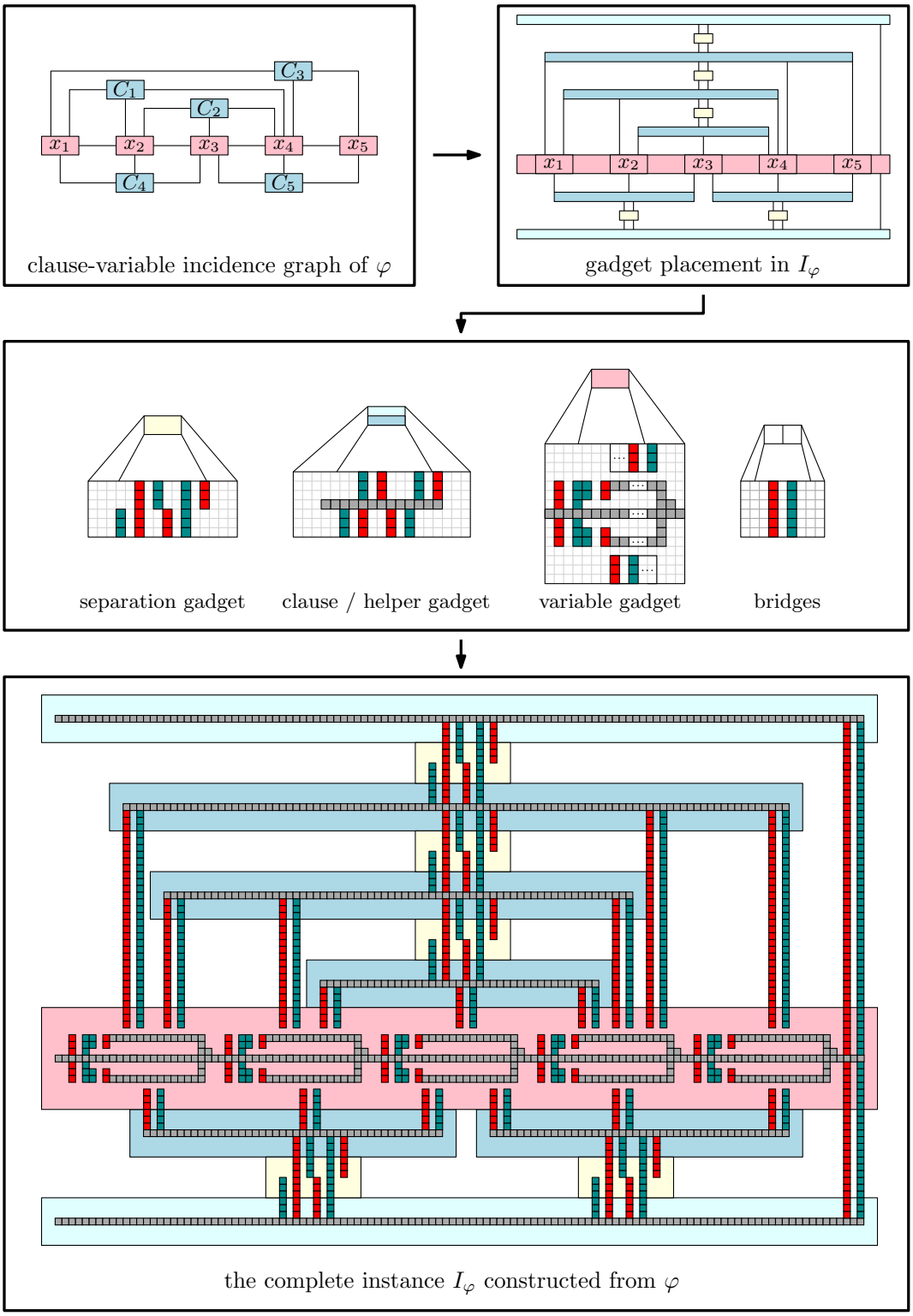

Figure 2 Symbolic overview of the NP-hardness reduction.

4 Bounded Stretch for Arbitrary Makespan

▶ **Theorem 4.** *There is a constant c such that for any pair of start and target configurations with a fatness of at least c, there is a stable schedule of constant stretch.*

On a high level, the overall schedule proceeds in four phases; see Figure 3 for an overview. In the preprocessing phase, we use a bottleneck matching algorithm for mapping the start configuration C_s to the target configuration C_t, minimizing the maximum distance d between a start and a target location. Furthermore, we establish the fatness in both configurations, set c to be the minimum of both fatness values, and compute a resulting set of *cd-tiles* that contain both C_s and C_t.

In the second phase, we build a scaffolding structure around C_s, based on the boundaries of *cd*-tiles, resulting in a *tiled configuration*, see Figure 3(a). This structure provides connectivity throughout the actual reconfiguration.

In the third phase, we perform the actual reconfiguration of the arrangement. This consists of refilling the tiles of the scaffolding structure, achieving the proper number of robots within each tile, based on elementary flow computations. As a subroutine, we transform the robots inside each tile into a canonical triangular configuration, see Figure 3(b), and Figures 5 to 7.

In the fourth and final phase, we disassemble the scaffolding structure and move the involved robots to their proper destinations, see Figure 3(c).

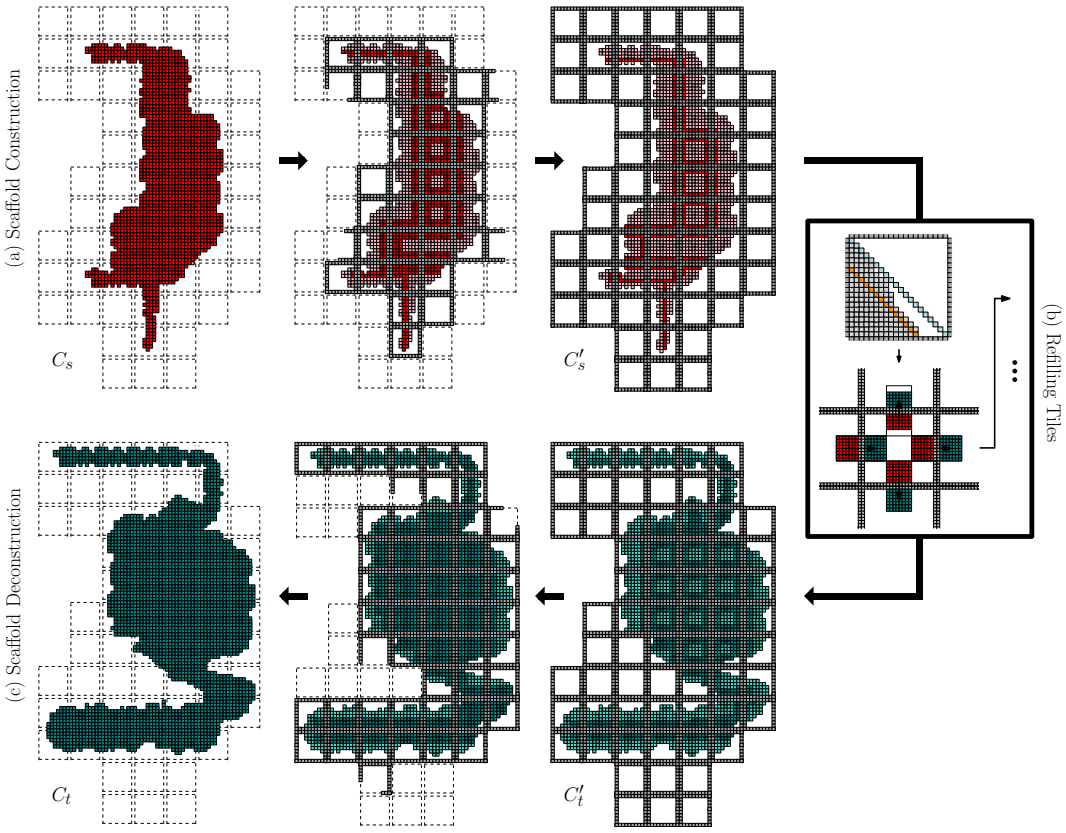

Figure 3 Overview of the computed schedule: (a) Constructing the scaffold: transforming the start configuration into a tiled configuration, (b) the refilling phase, and (c) deconstructing the scaffold: transforming the tiled configuration into the target configuration.

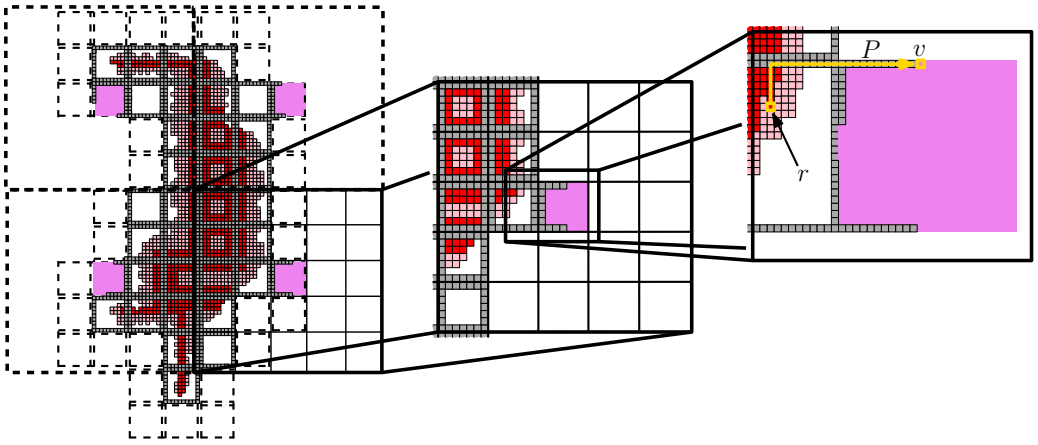

Figure 4 Constructing the scaffold. Tiles with currently constructed boundary are marked in pink, corresponding to one of the 25 tile classes. The zoom into the start configuration C_s shows the 5×5-neighborhood $N[T]$ of an active tile T (middle) and a further zoom into T with an associated robot motion (right). In each transformation step a robot from the interior of a tile $T' \in N[T]$ is swapped with a free position on the boundary of T based on a path P in a BFS-tree.

We will not go into detail regarding the preprocessing. Because disassembling the scaffold is the reverse of the building process, we will only describe the second and third phase:

Building the Scaffold. For the construction of the scaffold, we consider 25 different classes of tiles, based on x- and y-coordinates modulo $5cd$; see Figure 4. We process a single class as follows: For each tile T we consider its neighborhood $N[T]$ consisting of 5×5 tiles centered at T. Hence, the neighborhoods of different tiles of the same class are disjoint. For constructing the boundary of T, we make use of robots from the interior of a single tile in the neighborhood of T. In particular, we swap a free position on the boundary of T with an occupied position in the interior of a tile in $N[T]$, which is a leaf in a respective BFS-tree. Because the neighborhood of all tiles of the current class are disjoint, and a leaf cannot break connectivity, we obtain:

▶ **Lemma 5.** *There is a stable schedule within a makespan of $\mathcal{O}(d)$ transforming C_s into a tiled configuration C'_s, such that the interior of C'_s is a subset of the start configuration C_s.*

Reconfigure Single Tiles. We first compute $C_s \rightrightarrows_\chi C_s^m$ and $C_t \rightrightarrows_\chi C_t^m$, where C_s^m and C_t^m are monotone configurations. These reconfigurations are achieved by a specific sequence of down and left movements, maintaining connectivity after each move. Proceeding from these monotone configurations, the robots are arranged into a triangular configuration C_Δ that occupies the lower left positions (defined by a diagonal line with a slope of -1) of the interior of T. This is achieved by swapping pairs of occupied and empty positions within a carefully defined area in several one-step moves along L-shaped paths. The property of C_Δ is that it is the same for all initial configurations with equally many robots. Thus, to get the stable schedule $C_s \rightrightarrows_\chi C_\Delta \rightrightarrows_\chi C_t$ to reconfigure C_s into C_t, we can simply revert $C_t \rightrightarrows_\chi C_\Delta$ and combine the result with $C_s \rightrightarrows_\chi C_\Delta$; consider Figure 5 for illustration.

Because all distances are upper-bounded by $\mathcal{O}(d)$ and all robot movements stay within the interior of T, we can reconfigure all tiles in parallel to obtain the following:

▶ **Lemma 6.** *Let C'_s, C'_t be two tiled configurations such that C'_s and C'_t contain the same number of robots in the interior of T for each tile T. There is a stable schedule transforming C'_s into C'_t within a makespan of $\mathcal{O}(d)$.*

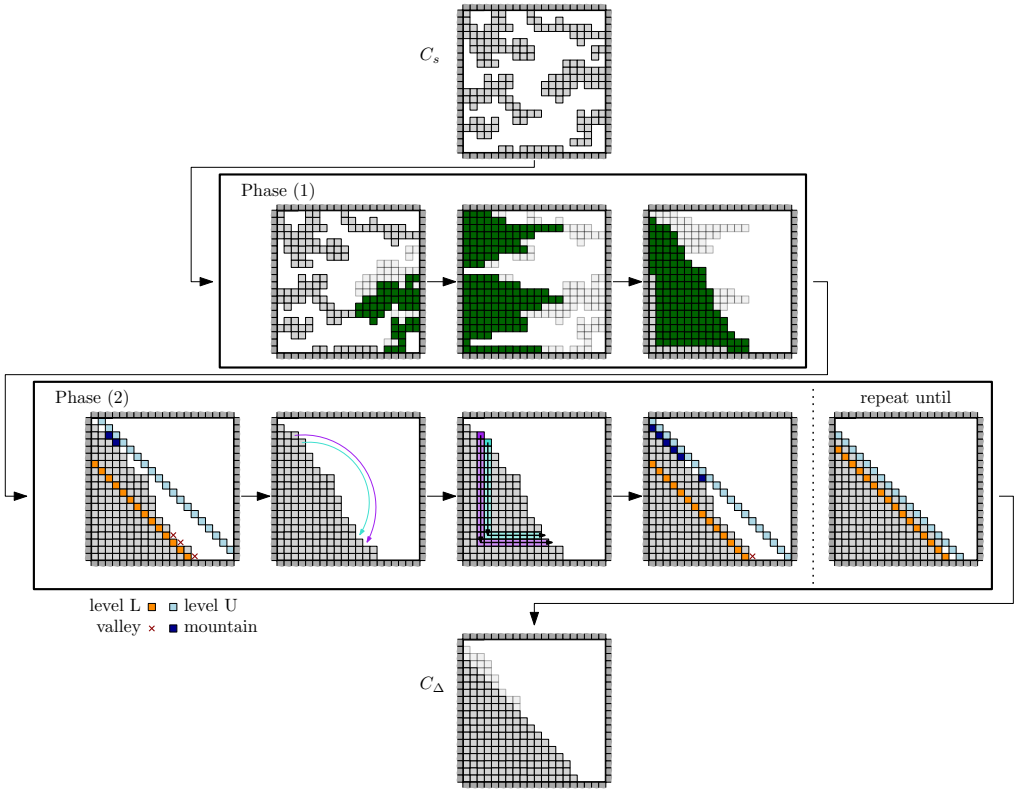

Figure 5 Turning arrangement C_s (top) into a canonical triangular configuration C_Δ (bottom). Phase (1) (left to right), achieves a monotonic arrangement; light grey indicates previous positions of active robots (shown in green). Phase (2) transforms the monotonic configuration into C_Δ.

Refilling Tiles. In general, tiles in C'_s and C'_t differ in the number of contained robots, so that we have to transfer robots between tiles. We model this robot transfer by a *supply and demand flow*, see Figure 6. By partitioning the flow into $\mathcal{O}(1)$ subflows, each subflow can be realized within a makespan of $\mathcal{O}(d)$. For realizing a single subflow, we use the reconfiguration of single tiles as a preprocessing step. In particular, we partition the interior of each tile T into nine *subtiles* with equal side lengths (up to rounding), see Figure 7. For each transfer path between T and T', we place the desired amount of robots inside the subtile of T that shares an edge with the boundary of T adjacent to T'. As mentioned, these configurations can be formed for all tiles in parallel in $\mathcal{O}(d)$, so that all robots can be moved into their respective target tiles. By repeating this approach, we obtain the following:

▶ **Lemma 7.** *We can efficiently compute a stable schedule transforming C'_s into C'_t within a makespan of $\mathcal{O}(d)$.*

The correctness of the approach (Theorem 4) follows from Lemmas 5, 6, and 7. As the diameter of the pair (C_s, C_t) is a lower bound for the makespan of any schedule transforming C_s into C_t, we obtain the following.

▶ **Corollary 8.** *There is a constant-factor approximation for computing stable schedules with minimal makespan between pairs of start and target configurations with a fatness of at least c, for some constant c.*

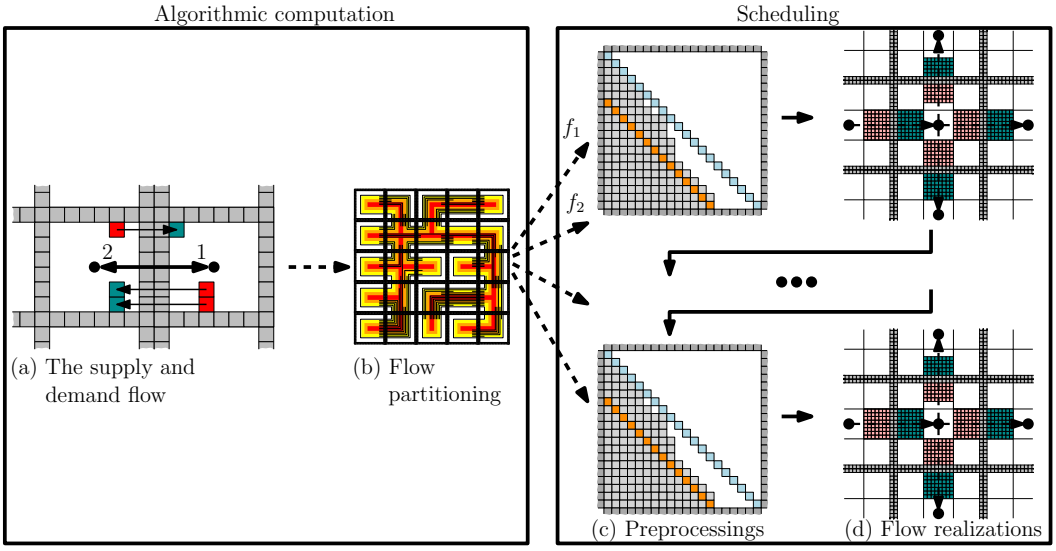

Figure 6 An overview of the schedule refilling tiles: transforming C'_s into C'_t by realizing a partition of a supply and demand flow that is computed in advance.

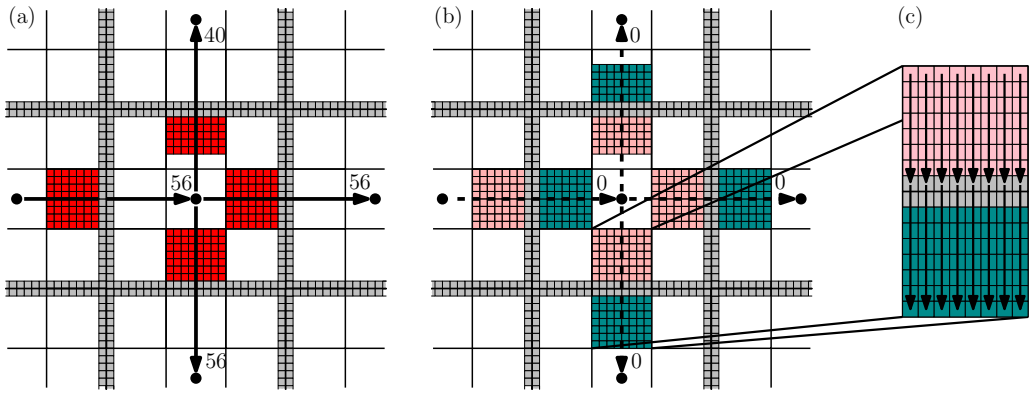

Figure 7 (a) A portion of a set of paths containing a vertex v: 56 paths are passing v from left to right, while $40 + 56 = 96$ paths are starting in v. (b) The configuration after realizing the set of paths shown in the previous figure. (c) How positions of robots have changed after realization.

5 Conclusion

We have shown that coordinated motion planning for a connected swarm of robots is a challenging problem, even in relatively simple cases. On the other hand, we have shown that (assuming sufficient connectivity of the swarm), it is possible to compute efficient reconfiguration schedules with constant stretch.

It is straightforward to extend our approach to other scenarios, e.g., to three-dimensional configurations. Other questions appear less clear. Can we show that constant stretch cannot be achieved for "thin" arrangements? Can we extend our methods to the labeled case? To what extent can the overall algorithm be turned into a set of distributed protocols, with only limited central computation and coordination?

References

1 Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Matthias Konitzny, Lillian Lin, and Christian Scheffer. Coordinated motion planning: The video. In *Proc. Symposium on Computational Geometry (SoCG)*, pages 74:1–74:6, 2018. Video at `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/CoordinatedMotionPlanning.mp4`.

2 Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855, 2018.

3 Daniel Delahaye, Stéphane Puechmorel, Panagiotis Tsiotras, and Eric Féron. Mathematical models for aircraft trajectory design: A survey. In *Air Traffic Management and Systems*, pages 205–247, 2014.

4 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. In *Proc. Symposium on Computational Geometry (SoCG)*, pages 29:1–29:17, 2018.

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

6 Sándor P. Fekete, Björn Hendriks, Christopher Tessars, Axel Wegener, Horst Hellbrück, Stefan Fischer, and Sebastian Ebers. Methods for improving the flow of traffic. In Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, editors, *Organic Computing — A Paradigm Shift for Complex Systems*, volume 1 of *Autonomic Systems*. Birkhäuser, 2011.

7 John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

8 Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

9 Erol Šahin and Alan Winfield (editors). Special issue on swarm robotics. *Swarm Intelligence*, 2(2–4), 2008.

10 Michael Schreckenberg and Reinhard Selten (editors). *Human Behaviour and Traffic Networks*. Springer, 2004.

11 Jacob T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. *Int. J. Robotics Res.*, 2(3):46–75, 1983.

12 M. Turpin, K. Mohta, N. Michael, and V. Kumar. Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots*, 37(4):401–415, 2014.