Coordinated Motion Planning: Multi-Agent Path Finding in a Densely Packed, Bounded Domain

Sándor P. Fekete ✉ 🆔
Department of Computer Science, TU Braunschweig, Germany

Ramin Kosfeld ✉ 🆔
Department of Computer Science, TU Braunschweig, Germany

Peter Kramer ✉ 🆔
Department of Computer Science, TU Braunschweig, Germany

Jonas Neutzner ✉ 🆔
Department of Computer Science, TU Braunschweig, Germany

Christian Rieck ✉ 🆔
Department of Computer Science, TU Braunschweig, Germany

Christian Scheffer ✉ 🆔
Department of Electrical Engineering and Computer Science,
Bochum University of Applied Sciences, Germany

──────── **Abstract** ────────

We study MULTI-AGENT PATH FINDING for arrangements of labeled agents in the interior of a simply connected domain: Given a unique start and target position for each agent, the goal is to find a sequence of parallel, collision-free agent motions that minimizes the overall time (the *makespan*) until all agents have reached their respective targets. A natural case is that of a simply connected polygonal domain with axis-parallel boundaries and integer coordinates, i.e., a *simple polyomino*, which amounts to a simply connected union of lattice unit squares or *cells*. We focus on the particularly challenging setting of densely packed agents, i.e., one per cell, which strongly restricts the mobility of agents, and requires intricate coordination of motion.

We provide a variety of novel results for this problem, including (1) a characterization of polyominoes in which a reconfiguration plan is guaranteed to exist; (2) a characterization of shape parameters that induce worst-case bounds on the makespan; (3) a suite of algorithms to achieve asymptotically worst-case optimal performance with respect to the achievable *stretch* for cases with severely limited maneuverability. This corresponds to bounding the ratio between obtained makespan and the lower bound provided by the max-min distance between the start and target position of any agent and our shape parameters.

Our results extend findings by Demaine et al. [13, 14] who investigated the problem for solid rectangular domains, and in the closely related field of PERMUTATION ROUTING, as presented by Alpert et al. [6] for convex pieces of grid graphs.

35th International Symposium on Algorithms and Computation (ISAAC 2024).
Editors: Julián Mestre and Anthony Wirth; Article No. 29; pp. 29:1–29:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1   Introduction

Problems of coordinating the motion of a set of objects occur in a wide range of applications, such as warehouses [32], multi-agent motion planning [27, 28], and aerial swarm robotics [11]. In MULTI-AGENT PATH FINDING (MAPF) [31], we are given a set of agents, each with an initial and a desired target position within a certain domain. The task is to determine a coordinated motion plan: a sequence of parallel, collision-free movements such that the time by which all agents have reached their destinations (the *makespan*) is minimized.

Theoretical aspects of MAPF have enjoyed significant attention. In the early days of computational geometry, Schwartz and Sharir [29] developed methods for coordinating the motion of disk-shaped objects between obstacles, with runtime polynomial in the complexity of the obstacles, but exponential in the number of disks. The fundamental difficulty of geometric MAPF was highlighted by Hopcroft et al. [21, 22], who showed that it is PSPACE-complete to decide whether multiple agents can reach a given target configuration. In contrast, closely related graph-based variants of the MAPF problem permit for linear time algorithms for the same decision problem [35].

More recently, Demaine et al. [10, 13, 14] have provided methods to compute *constant stretch* solutions for coordinated motion planning in unbounded environments in which agents occupy distinct grid cells. The stretch of a solution is defined as the ratio between its makespan and a trivial lower bound, the *diameter d*, which refers to maximum distance between any agent's origin and destination. Their work therefore obtains collision-free motion schedules that move each agent to its target position in $\mathcal{O}(d)$ discrete moves, which corresponds to a constant-factor approximation. However, their methods assume the absence of a environmental boundary that may impose external constraints on the agents' movements.

1.1   Our contributions

In this paper, we study MULTI-AGENT PATH FINDING for densely packed arrangements of labeled agents that are required to remain within discrete grid domains, i.e., polyominoes. This is a natural constraint that occurs in many important applications, but provides considerable additional difficulties; in particular, a coordinated motion plan may no longer exist for domains with narrow bottlenecks. We provide a variety of novel contributions:

- We give a full characterization of simple polyominoes P that are universally reconfigurable. These allow *some* feasible coordinated motion plan for *any* combination of initial and desired target configurations, without regard for the makespan: We prove that this is the case if and only if P has a cover by 2×2 squares with a connected intersection graph.
- We model the shape parameters *bottleneck length $\zeta(P)$* (which is the minimum length of a cut dividing the region into non-trivial pieces) and *domain depth $\mu(P)$* (which is the maximum distance of any cell from the domain boundary). We provide refined upper and lower bounds on the makespan and stretch factor based on these shape parameters.
- For some instances, any applicable schedule may require a makespan of $\Omega(d + {}^{d^2}\!/_{\zeta(P)})$. We show how to compute schedules of makespan linear in the ratio of domain area and bottleneck, i.e., $\mathcal{O}({}^{n}\!/_{\zeta(P)})$.
- We characterize *narrow* instances, which feature very limited depth relative to the diameter d, and provide an approach for asymptotically worst-case optimal schedules.

Note that (parts of) the proofs of statements marked with (⋆) have been omitted in the main sections due to space constraints; we refer the reader to the full version [18] instead.

1.2 Related work

Motion planning. MULTI-AGENT PATH FINDING is a widely studied problem. Due to space constraints, we restrict our description to the most closely related work. For more detailed references, refer to the extensive bibliography in [14] and the mentioned surveys [11, 28, 31].

Of fundamental importance to our work are the results by Demaine et al. [14], who achieved reconfiguration with constant stretch for the special case of rectangular domains. A key idea is to consider a partition of the rectangle into tiles whose size is linear in diameter d. These tiles can then be reconfigured in parallel. First, flow techniques are applied to shift agents into their target tile; afterward, agents are moved to their respective target positions. Furthermore, they showed that computing the optimal solution is strongly NP-complete.

Fekete et al. [16, 19] considered the unconstrained problem on the infinite grid with the additional condition that the whole arrangement needs to be connected after every parallel motion. They considered both the labeled and the unlabeled version of the problem, providing polynomial-time algorithms for computing schedules with constant stretch for configurations of sufficient scale. They also showed that deciding whether there is a reconfiguration schedule with a makespan of 2 is already NP-complete, unlike deciding the same for a makespan of 1.

Eiben, Ganian, and Kanj [15] investigated the parameterized complexity of the problem for the variants of minimizing the makespan and minimizing the total travel distance. They analyzed the problems with respect to two parameters: the number of agents, and the objective target. Both variants are FPT when parameterized by the number of agents, while minimizing the makespan becomes para-NP-hard when parameterized by the objective target.

Further related work studies (unlabeled) multi-robot motion planning problems in polygons. Solovey and Halperin [30] show that the unlabeled variant is PSPACE-hard, even for the specific case of unit-square robots moving amidst polygonal obstacles. Even in simple polygonal domains, a feasible motion-plan for unlabeled unit-disk robots does not always exist, if, e.g., the robots and their targets are positioned too densely. However, if there is some minimal distance separating start and target positions, Adler et al. [1] show that the problem always has a solution that can be computed efficiently. Banyassady et al. [8] prove tight separation bounds for this case. Agarwal et al. [2] consider the labeled variant with revolving areas, i.e., empty areas around start and target positions. They prove that the problem is APX-hard, even when restricting to weakly-monotone motion plans, i.e., motion plans in which all robots stay within their revolving areas while an active robot moves to its target. However, they also provide a constant-factor approximation algorithm.

The computational complexity of moving two distinguishable square-shaped robots in a polygon environment to minimize the sum of traveled distances is still open; Agarwal et al. [3] gave the first polynomial-time $(1 + \varepsilon)$-approximation algorithm.

The problem was the subject of the 2021 CG:SHOP Challenge; see [17, 12, 24, 34] for an overview and a variety of practical computational methods and results.

Token swapping and routing via matchings. The task in the TOKEN SWAPPING PROBLEM is to transform two vertex labelings of a graph into one another by exchanging *tokens* between adjacent vertices by sequentially selecting individual edges. This problem is NP-complete even for trees [4], and APX-hard [26] in general. Several approximation algorithms exist for different variants and classes of graphs [20, 26, 33]. The PERMUTATION ROUTING variant allows for parallelization, by selecting disjoint edge sets to perform swaps in parallel [5, 9, 23]. The *routing number* of a graph describes the maximal number of necessary parallel swaps between any two labelings. Recently, Alpert et al. [6] presented an upper bound on the routing number of convex pieces of grid graphs, which is very closely related to our setting.

1.3 Preliminaries

We define the considered motion of *agents* in a restricted environment (*domain*) as follows.

Domain. Consider the infinite integer grid graph, in which each 4-cycle bounds a face of unit area, a *cell*. Every planar edge cycle in this grid graph bounds a finite set of cells, which induces a domain that we call a *(simple) polyomino*, see Figure 1a. We exclusively consider simple polyominoes, i.e., those without holes. For the sake of readability, we might not state this for each individual polyomino in later sections. The *area* of a polyomino P is equal to the number of contained cells n. The bounding edge cycle and its incident cells are therefore called is the *boundary* and *boundary cells* of P, respectively. The dual graph of P, denoted by $G(P) = (V, E)$, has a vertex for every cell, two of which are *adjacent* if they share an edge in P, as shown in Figure 1b. The *geodesic distance* between two cells of a
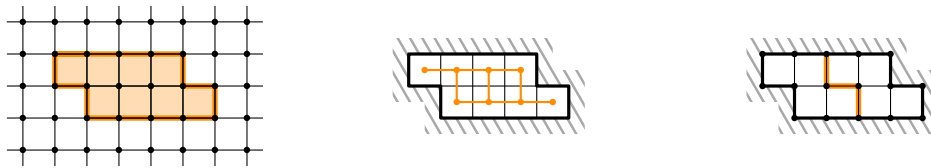
(a) A boundary cycle. **(b)** A polyomino's dual graph. **(c)** A cut through a polyomino.

Figure 1 A polyomino, its dual graph, and a cut. Subsequent illustrations will only show the boundary and any relevant cuts, foregoing the underlying integer grid.

polyomino P corresponds to the length of a shortest path between the corresponding vertices in $G(P)$. A *cut* through a polyomino P is defined by a planar path between two vertices on the boundary of P, as shown in Figure 1c. Its cut set corresponds exactly to those edges of $G(P)$ which cross the path. Geometrically, this induces two simple subpolyominoes Q, R and write $Q, R \subset P$. We say that a cut is *trivial* if its endpoints on the boundary of P have a connecting path on the boundary that is not longer than the cut itself.

Agents. We consider distinguishable *agents* that occupy the cells of polyominoes. A *configuration* of a polyomino P with $G(P) = (V, E)$ is a bijective mapping $C : V \to \{1, \ldots, n\}$ between cells and agent labels. We denote the set of all configurations of P as $\mathcal{C}(P)$.

In each discrete time step, an agent can either *move*, changing its position v to an adjacent position w, or hold its current position. We denote this by $v \to w$ or $v \to v$, respectively. Two parallel moves $v_1 \to w_1$ and $v_2 \to w_2$ are *collision-free* if $v_1 \neq v_2$ and $w_1 \neq w_2$. We assume that a *swap*, i.e., two moves $v_1 \to v_2$ and $v_2 \to v_1$, causes a collision, and is therefore forbidden. Configurations can be *transformed* by sets of collision-free moves that are performed in parallel. If a set of moves transforms a configuration C_1 into a configuration C_2, this set is also called a *transformation* $C_1 \to C_2$. For an illustrated example, see Figure 2. A *schedule* with *makespan* $M \in \mathbb{N}$ is then a sequence of transformations $C_1 \to \cdots \to C_{M+1}$, also denoted by $C_1 \rightrightarrows C_{M+1}$.

(a) A polyomino and a configuration. **(b)** Four agents move along a cycle.

Figure 2 An illustration of an configurations and a transformations.

Problem statement. We consider the Multi-Agent Path Finding Problem for agents in a discrete environment bounded by a simple polyomino. Thus, an *instance* of the problem is composed of two configurations $C_1, C_2 \in \mathcal{C}(P)$ of a simple polyomino P. We say that a schedule is *applicable* to the instance exactly if it transforms C_1 into C_2. The *diameter of an instance* is the maximum geodesic distance d between an agents' start and target positions, and the *stretch* of an applicable schedule is the ratio between its makespan and the diameter.

2 Reconfigurability

In this section, we provide a characterization of (simple) polyominoes for which any configuration can be transformed into any other. We say that a polyomino P is *universally reconfigurable* if there exists an applicable schedule for any two configurations $C_1, C_2 \in \mathcal{C}(P)$. We prove that this is the case if and only if P has a cover by cycles that have a connected intersection graph, and show how to compute an applicable schedule of makespan $\mathcal{O}(n)$.
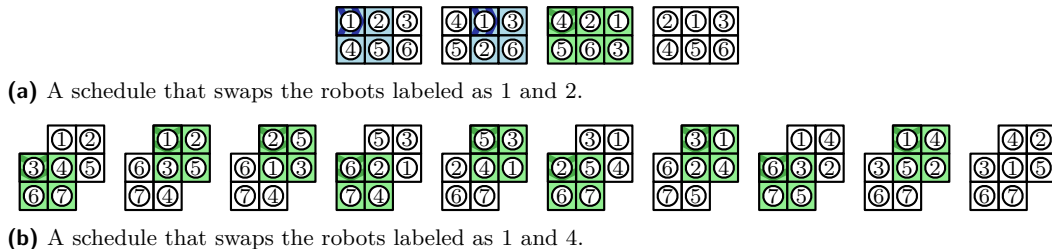
▶ **Theorem 1.** *A polyomino P is universally reconfigurable if and only if it has a cover by 2×2 squares with a connected intersection graph. For any $C_1, C_2 \in \mathcal{C}(P)$ of such a polyomino with area n, an applicable schedule $C_1 \rightrightarrows C_2$ of makespan $\mathcal{O}(n)$ can be computed efficiently.*

Due to the cyclic nature of all movement, the edge connectivity of a polyomino's dual graph plays a significant role for universal reconfigurability. We start with a negative result.

▶ **Lemma 2** ($\star$). *A polyomino P that does not have a cover by 2×2 squares with a connected intersection graph is not universally reconfigurable.*

Proof sketch. We observe that transformations are inherently cyclic, as the domain is fully occupied and collisions (and thus, swaps) are forbidden. Using this fact, it is easy to show that 2-edge-connectedness is necessary and sufficient for universal reconfigurability, provided an area of at least 6 cells. From here, it is possible to show that a cover 2×2 square must exist for any two adjacent cells, implying the property.   ◀

Because direct swaps of adjacent agents are not possible, an important tool is the ability to "simulate" a large number of adjacent swaps in parallel, using a constant number of transformation steps. Polyominoes that are unions of two 2×2 squares form an important primitive to achieve this. There exist two classes of such polyominoes; the squares can overlap either in one or two cells. Clearly, either dual graph can be covered by two 4-cycles that intersect in at least one vertex. In Figure 3, we illustrate schedules that perform adjacent swaps at the intersection of these cycles, implying universal reconfigurability of both classes. In fact, any instance of either class takes at most 7 or 14 transformations, respectively.

(a) A schedule that swaps the robots labeled as 1 and 2.

(b) A schedule that swaps the robots labeled as 1 and 4.

■ **Figure 3** In polyominoes composed of two 2×2 squares, we can realize swaps in $\mathcal{O}(1)$ steps.

▶ **Observation 3.** *Polyominoes of two overlapping 2×2 squares are universally reconfigurable.*

Using this observation for a primitive local operation, we show the following.

▶ **Lemma 4.** *For any matching in the dual graph of a universally reconfigurable polyomino, we can compute a schedule of makespan $\mathcal{O}(1)$ which swaps the agents of all matched positions.*

Proof. Let I refer to the connected intersection graph of a cover of a universally reconfigurable polyomino P by 2×2 squares, which can be computed in $\mathcal{O}(n)$. Due to Lemma 2, the vertices of each edge in the dual graph of P share a common 2×2 square in the cover.

We thus divide the edges $E(I)$ of I into 36 classes based on each edge's orientation and xy-minimal coordinates mod 3. These can be represented by $\{\uparrow, \nearrow, \rightarrow, \searrow\} \times [0, 2] \times [0, 2]$.

For any intersection $\{u, v\} \in E(I)$, let $R(\{u, v\})$ now be the union of the vertices covered by the squares u and v, which always corresponds exactly to one of the polyominoes outlined in Observation 3. Such a region has a bounding box no larger than 3×3, which means that the regions $R(e)$ and $R(f)$ are disjoint for any two edges e and f in a common class, allowing us to apply ROTATESORT in parallel to all the regions within one class.

As there are constantly many classes, we can realize the adjacent swaps induced by a matching of adjacent cells in $\mathcal{O}(1)$ transformations.   ◀

Marberg and Gafni [25] propose an algorithm called ROTATESORT that sorts a two-dimensional $n \times m$ array within $\mathcal{O}(n+m)$ parallel steps. Demaine et al. [14] demonstrate that this algorithm can be applied geometrically, using the local swap mechanism illustrated in Figure 3a. A geometric application of ROTATESORT is a sequence of sets of pairwise disjoint adjacent swap operations, i.e., sets consisting of pairs of adjacent cells, where swaps can be simulated by circular rotations. As our setting is not merely restricted to rectangular domains, we extend their approach using Lemma 4. We give a constructive proof of Theorem 1 in the shape of an algorithm, as follows.

Proof of Theorem 1. Our approach employs methods from PERMUTATION ROUTING. In this setting, the task is to transform two different vertex labelings of a graph into one another by exchanging labels between adjacent vertices in parallel [5]. A solution (or routing sequence) consists of a series of matchings, i.e., sets of independent edges, along which tokens are exchanged. Such a routing sequence of length at most $3n$ can be computed in almost linear time if the underlying graph is a tree [5]. Thus, we consider an arbitrary spanning tree of a polyomino P's dual graph and compute such a routing sequence. Due to Lemma 4, each parallel swap operation in the sequence can be realized by a schedule of makespan $\mathcal{O}(1)$. We conclude that the schedule derived from the routing sequence has makespan $\mathcal{O}(n)$.   ◀

3   The impact of the domain on the achievable makespan

Previous work has demonstrated that it is possible to achieve constant stretch for labeled agents in a rectangular domain. However, in the presence of a non-convex boundary, such stretch factors may not be achievable. We present the following worst-case bound.

▶ **Proposition 5.** *For any $d \geq 5$, there exist instances of diameter d in universally reconfigurable polyominoes, such that all applicable schedules have makespan $\Omega(d^2)$.*

Proof. We illustrate a class of such instances in Figure 4. In this class, agents located on different sides of a narrow passage must trade places. Theorem 1 tells us that these polyominoes are universally reconfigurable; however, any movement between the regions pass through the narrow passage at the center, limiting the number of agents exchanged between them to 2 per transformation. As the number of agents scales quadratically with d, any schedule for this class of instances requires a makespan of $\Omega(d^2)$.   ◀
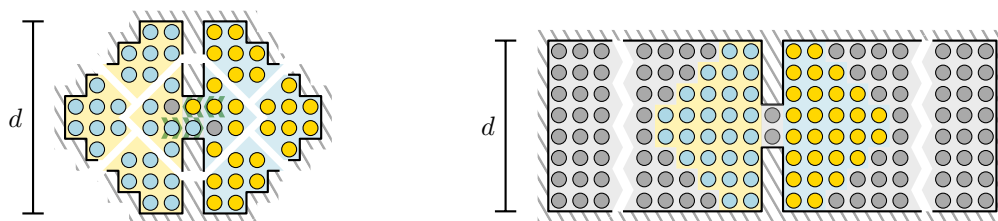
(a) A polyomino with narrow passage.   **(b)** This class can be extended such that $n > d^2$.

Figure 4 We illustrate a class of instances which require $\Omega(d^2)$ transformations.

For a more refined characterization of features that affect the achievable makespan and to formulate a precise lower bound, we introduce the following shape parameter for polyominoes.

Bottleneck. We say that the *bottleneck* of a polyomino P is the largest integer $\zeta(P)$, such that there is no non-trivial cut through P of length less than $\zeta(P)$. This means no interior "shortcut" of length less than $\zeta(P)$ exists between any two points on the boundary of P. A *bottleneck cut* through P is therefore a non-trivial cut of length $\zeta(P)$.

We now further refine the lower bound presented in Proposition 5, as follows.

▶ **Proposition 6.** *For any $d \geq 4$ and $z \in [2, d]$, there exists a universally reconfigurable polyomino P with $\zeta(P) = z$ that has instances of diameter d, for which any applicable schedule has a makespan of $\Omega(d^2/\zeta(P))$, i.e., a stretch factor of $\Omega(d/\zeta(P))$.*

Proof. We formulate a generalized version of the instances from Proposition 5. By scaling the boundary of the polyomino between the two regions by an arbitrary amount less or equal to d, we can create universally reconfigurable polyominoes with the targeted bottleneck value.

The movement between the two regions must then still be realized over the narrow grey region, limiting the number of robots exchanged between them to $\mathcal{O}(\zeta(P))$ per transformation. As the number of robots that need to traverse the bottleneck cut scales quadratically with d, any applicable schedule for this class of instances requires a makespan of $\Omega(d^2/\zeta(P))$. ◀

To further refine our understanding of the domain's impact on achievable makespans, we now characterize the size of widest passages, i.e., best case maneuverability. To this end, we consider the maximum shortest distance to the boundary within the given domain, its *depth*.

Depth. We say that the *depth* of a polyomino P is the smallest integer $\mu(P)$, such that every cell in P has geodesic distance at most $\mu(P)$ to the boundary of P.

The depth and bottleneck of a polyomino are very closely related, with depth implying a bound on the bottleneck of any (sub-)polyomino such that $\zeta(P') \leq 2\mu(P)$ for any $P' \subseteq P$. We take particular notice of the following property of depth.

▶ **Lemma 7** (⋆). *From any cell in a polyomino P, the maximal geodesic distance to a non-trivial geodesic cut of length at most $2\mu(P)$ is also at most $2\mu(P)$.*

4   Bounded makespan for narrow instances

In this section, we consider algorithms for bounded makespan in specific families of instances. Our central result is an approach for asymptotically worst-case optimal stretch in *narrow* instances, which we define as follows. An instance of diameter d in a polyomino P is *narrow*, if and only if $\pi \cdot d \geq \mu(P)$ for some constant $\pi \in \mathbb{N}$, i.e., $\mu(P) \in \mathcal{O}(d)$. Intuitively, these correspond to instances of large diameter relative to the domain's depth.

▶ **Theorem 8** (⋆). *Given an instance of diameter d in a universally reconfigurable polyomino P, we can efficiently compute an applicable schedule of makespan $\mathcal{O}\big((d+\mu(P))^2/\zeta(P)\big)$. This is asymptotically worst-case optimal for narrow instances.*

As our proof is fairly involved, we proceed with the special case of *scaled polyominoes* in Section 4.1, which we extend to arbitrary polyominoes of limited depth in the subsequent Section 4.2. In each section, we first establish bounds on the makespan relative to a polyomino's area and the corresponding shape parameter.

4.1   Bounded makespan and stretch based on scale

We now investigate *scaled polyominoes*, which we define as follows.

(a) A 3-scaled polyomino P and its tiles.   **(b)** The tile dual graph of P.
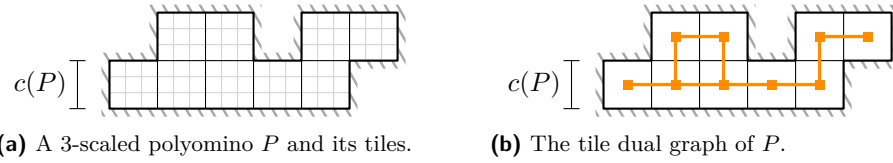
■ **Figure 5** An illustration of a scaled polyomino P, its tiles, and their corresponding dual graph.

Scaled polyomino. For any $c \in \mathbb{N}$, we say that a polyomino P is *c-scaled* exactly if it is composed of $c \times c$ squares that are aligned with a corresponding $c \times c$ integer grid. We call these grid-aligned squares *tiles*, which have a *dual graph* analogous to that of a polyomino. Finally, the *scale* of a polyomino P is the largest integer $c(P)$ such that P is c-scaled. This additionally represents a very natural lower bound on the bottleneck, $\zeta(P) \geq c(P)$.

▶ **Proposition 9.** *For any two configurations of a polyomino P with area n and $c(P) \geq 3$, we can compute an applicable schedule of makespan $\mathcal{O}(n/c(P))$ in polynomial time.*

Proof. We model our problem as an instance of PERMUTATION ROUTING, taking note of two significant results regarding the routing number of specific graph classes. Recall that the routing number $rt(G)$ of a specific graph G refers to the maximum number of necessary routing operations to transform one labeling of G into another. For the complete graph K_n with n vertices, it was shown by Alon, Chung, and Graham [5] that $rt(K_n) = 2$. Furthermore, we make use of a result by Banerjee and Richards [7] which states that for an h-connected graph G and any connected h-vertex induced subgraph G_h of G, the routing number $rt(G)$ is in $\mathcal{O}(rt(G_h) \cdot n/h)$. They also describe an algorithm that determines a routing sequence that matches this bound.

Given a polyomino P and two configurations $C_1, C_2 \in \mathcal{C}(P)$, our goal is to define a secondary graph over the vertices of the dual graph $G(P) = (V, E)$ such that a routing sequence over this graph can be transformed into a schedule $C_1 \rightrightarrows C_2$ of makespan $\mathcal{O}(n/c(P))$.

We define $G_c = (V, E_c)$ such that $\{u, v\} \in E_c$ exactly if the cells u and v are located in the same $c(P) \times c(P)$ tile, or two adjacent tiles. As a result, the cells of each tile in P form a clique, i.e., their induced subgraph is isomorphic to $K_{c(P)^2}$. Furthermore, the cliques of cells in any two adjacent tiles are connected by a set of complete bipartite edges, so they also form a clique. Hence, G_c is h-connected for $h \geq c(P)^2 - 1$ and contains $n/c(P)^2$ cliques of order at least $c(P)^2$. Due to Banerjee and Richards [7], we conclude that $rt(G_c)$ is in $\mathcal{O}(rt(K_{c^2}) \cdot n/h) = \mathcal{O}(n/c(P)^2)$ and can therefore compute a sequence of $\mathcal{O}(n/c(P)^2)$ matchings to route between any two labelings of $G(P)$, which correspond to configurations of P.

It remains to argue that we can realize the swaps induced by any matching in G_c by means of $\mathcal{O}(c(P))$ transformations. All pairwise swaps between cells within the same tile can be realized by applying ROTATESORT to all tiles in parallel, taking $\mathcal{O}(c(P))$ transformations.

We therefore turn our attention to swaps between adjacent tiles. Observe that the dual graph of the tiles of P is a minor of G_c; contracting the vertices in each of the tile-cliques defined above will give us a corresponding grid graph. Swaps between adjacent tiles can therefore be realized in four phases by covering this grid graph by matchings, and applying ROTATESORT to the union of matched tile pairs in parallel, again taking $\mathcal{O}(c(P))$ transformations. A cover by four matchings can be determined by first splitting the edges of the dual graph into two sets of horizontal and vertical edges, respectively. Each of these edge sets then induces a collection of paths in the tiling's dual graph, and can therefore be covered by two matchings.

We conclude that constantly many phases of parallel applications of ROTATESORT suffice to realize any matching in G_c. As $\mathcal{O}(n/c(P)^2)$ matchings can route between any two configurations of P, we conclude that this method yields schedules of makespan $\mathcal{O}(n/c(P))$. ◄

We now apply this intermediate result to compute schedules of bounded stretch in narrow instances of scaled polyominoes. Our approach hinges on the ability to divide the instance into subproblems that can be solved in parallel, which corresponds to cutting the polyomino and performing a sequence of preliminary transformations such that each subpolyomino can then be reconfigured locally, obtaining the target configuration.

Domain partitions. A *partition* of a polyomino P corresponds to a set of disjoint subpolyominoes that cover P. We observe that not every polyomino permits a partition into disjoint universally reconfigurable subpolyominoes. However, any subpolyomino of a universally reconfigurable polyomino P can be made universally reconfigurable by including cells of geodesic distance at most 2 in P.

To efficiently determine such a partition, we employ breadth-first search as follows.

Breadth-first search. For any polyomino Q, let $\texttt{BFS}(Q, v, r)$ refer to the subpolyomino of Q that contains all cells reachable from some cell v in Q by geodesic paths of length at most r. Further, let $\overline{\texttt{BFS}}(Q, v, r)$ refer to the set of connected components of $Q \setminus \texttt{BFS}(Q, v, r)$. We define the *wavefront* of $\texttt{BFS}(Q, v, r)$ as the set of cuts through Q that define the components of $\overline{\texttt{BFS}}(Q, v, r)$. Each connected component (cut) of the wavefront is called a *wavelet*.

▶ **Lemma 10** (⋆)**.** *For any polyomino Q, the wavefront of $\texttt{BFS}(Q, v, r)$ consists of wavelets of length $\mathcal{O}(\mu(Q))$ each, i.e., the wavelet length is independent of the search radius r.*

Having established all necessary tools, we prove the following statement.

▶ **Proposition 11.** *Given an instance of diameter d in a polyomino P with $c(P) \geq 3$, we can efficiently compute an applicable schedule with makespan $\mathcal{O}((d+\mu(P))^2/c(P))$. This is asymptotically worst-case optimal for narrow instances.*

Proof. We consider an instance of diameter d in a simple polyomino P. We proceed in three phases, which we briefly outline before giving an in-depth description of each.

(I) We partition P into $c(P)$-scaled *patches* of area $\mathcal{O}(d^2)$, using non-trivial cuts of bounded length such that the partition's dual graph is a rooted tree T.

(II) We combine parent/child patches according to T into regions with $c(P)$ scale, allowing us to apply Proposition 9 to reorder them in $\mathcal{O}((d+\mu(P))^2/c(P))$.

(III) Finally, we exploit these combined regions to place all agents at their destination.

Phase (I).   A step-by-step illustration of Phase (I) can be found in Figure 6. For this phase, we consider the polyomino P' induced by the tile dual graph of P, recall Figure 5b. This scales the shape parameters and geodesic distance by $1/c(P)$:

$$c(P') = c(P)/c(P) = 1, \qquad \mu(P') \cong \mu(P)/c(P), \qquad \zeta(P') \cong \zeta(P)/c(P).$$

Let $\delta = 3d/c(P)$. We subdivide P' using a recursive breadth-first-search approach and argue by induction. Given a boundary cell v_0 in P', we determine a patch $P'_0 \subseteq P'$ based on $\texttt{BFS}(P', v_0, \delta) \subseteq P'$. We say that the components of $\overline{\texttt{BFS}}(P', v_0, \delta)$ are either *small* or *large*; a component R is small exactly if $R \subset \texttt{BFS}(P', v_0, 2\delta)$, and large otherwise, see Figure 6b.

We define P'_0 as the union of the initial $\texttt{BFS}$ and the small components of its complement, meaning that for $\overline{\texttt{BFS}}(P', v_0, \delta)$ with large components $R_1, \ldots, R_\ell$, P'_0 takes the shape

$$P'_0 := \quad P' \setminus (R_1 \cup \ldots \cup R_\ell).$$

Due to Lemma 10, the cut Γ_i that separates a component R_i of $\overline{\texttt{BFS}}(P', v_0, \delta)$ from P'_0 has length $\mathcal{O}(\mu(P'))$. By definition, the geodesic distance from each cell in R_i to v_0 is at least δ.

We now iteratively subdivide each component of $P' \setminus P'_0$ by simply increasing the maximal depth of our $\texttt{BFS}$ from v_0 by another δ units and again considering large and small components of the corresponding subdivision separately, as illustrated in Figures 6c and 6d.
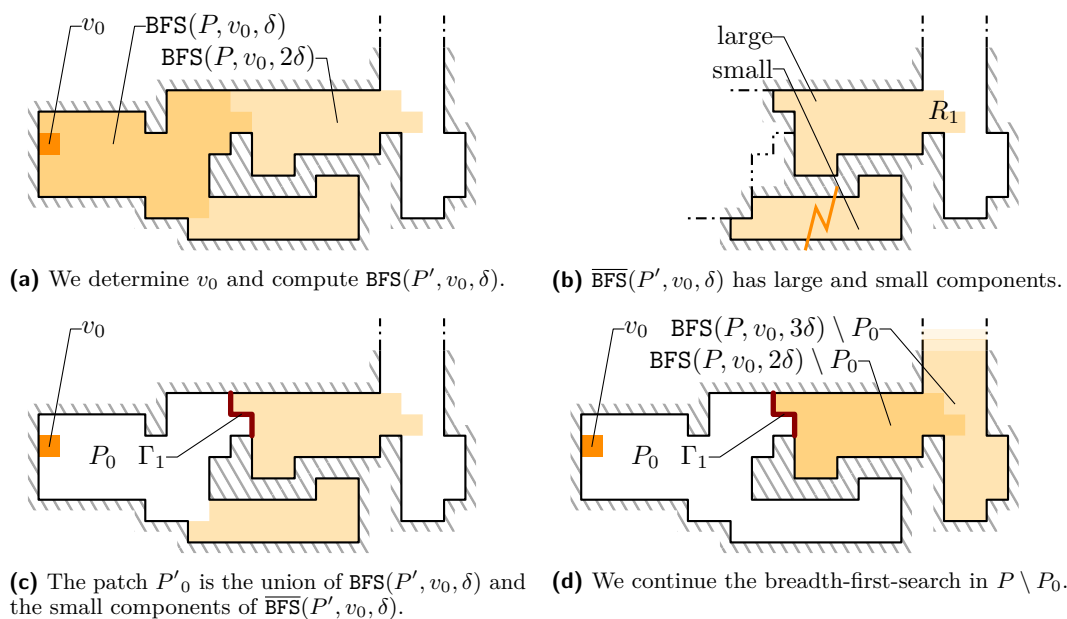
To obtain a partition of P, we map each patch P'_i to the tiles in P that its cells correspond to. Since P is a simple polyomino, the dual graph of our patches forms a tree T rooted at P_0.

Consider any patch P'_i and recall that, due to Lemma 10, all cuts induced by $\texttt{BFS}$ have individual length $\mathcal{O}(\mu(P'))$. Tracing along the boundaries of tiles, we conclude that the corresponding cuts in P have individual length $\mathcal{O}(\mu(P')c(P)) = \mathcal{O}(\mu(P))$. Due to triangle inequality, it follows that any two cells in each patch P_i have geodesic distance $\mathcal{O}(d + \mu(P))$. From this, we conclude that the area of P_i is bounded by $\mathcal{O}((d + \mu(P))^2)$. It directly follows that for any patch P_j with hop distance $k \in \mathbb{N}^+$ to P_i in T, the geodesic distance between two cells in P_i and P_j is bounded by $\mathcal{O}(k(d + \mu(P)))$. The union of patches in a subtree T' of T with height k therefore has area $\mathcal{O}((k(d + \mu(P)))^2)$.

Phase (II).   We use this partition of P into patches to subdivide the instance into disjoint tasks that can be solved in parallel; recall that our target makespan is $\mathcal{O}((d+\mu(P))^2/\zeta(P))$. The patches are spatially disjoint and all have scale at least $c(P)$, as well as area $\mathcal{O}((d + \mu(P))^2)$. Proposition 9 therefore implies that the patches can be locally reconfigured in parallel, by schedules of makespan $\mathcal{O}((d+\mu(P))^2/\zeta(P))$. In order to solve the original instance, it therefore remains to make each patch a subproblem that can be solved independently.

We argue that we can efficiently move robots into their target patches. In Phase (I), we gave an upper bound of $\mathcal{O}(k(d + \mu(P)))$ on the geodesic distance between cells in patches that have hop distance at most $k \in \mathbb{N}^+$ in T. We now provide a lower bound: The geodesic distance between cells in patches that are not in a parent-child or sibling relationship in T is at least d, as the distance between cells in any patch and its "grandparent" patch according to T is at least d by construction, see Phase (I).

It follows that the target cell of each robot is either in the same patch as its initial cell, or in a parent or sibling thereof. To realize the movement of agents between patches, we thus simply form the spatial union F_i of each patch P_i and its children according to T. Each of the resulting subpolyominoes F_i has area $\mathcal{O}((d + \mu(P))^2)$. As T is bipartite, we can split them into two sets $\mathcal{F}_A$ and $\mathcal{F}_B$, each comprised of pairwise spatially disjoint subpolyominoes.

(a) We determine v_0 and compute $\mathrm{BFS}(P', v_0, \delta)$.

(b) $\overline{\mathrm{BFS}}(P', v_0, \delta)$ has large and small components.

(c) The patch P'_0 is the union of $\mathrm{BFS}(P', v_0, \delta)$ and the small components of $\overline{\mathrm{BFS}}(P', v_0, \delta)$.

(d) We continue the breadth-first-search in $P \setminus P_0$.

Figure 6 Phase (I): We divide P' into patches of area $\mathcal{O}((\delta + \mu(P'))^2)$.

Phase (III).   It remains to show that we can efficiently exchange agents between patches. Note that, as P is simple, the number of agents that need to cross any cut in either direction is equal to that for the opposite direction.

By construction, every pair of patches that needs to exchange agents between one another is fully contained in some $F_i \in (\mathcal{F}_A \cup \mathcal{F}_B)$. We proceed in three iterations: By applying Proposition 9 to each of the patches in $\mathcal{F}_A$ in parallel, we swap agents across cuts by swapping them with agents moving in the opposite direction. We repeat this process for $\mathcal{F}_B$ and finally perform a parallel reconfiguration of the individual patches, which allows us to place every robot in its target cell. Each iteration takes $\mathcal{O}\big((d+\mu(P))^2/\zeta(P)\big)$ transformations. ◄

## 4.2   Bounded makespan and stretch based on bottleneck

Finally, this section concerns itself with the transfer of results from Section 4.1 to arbitrary polyominoes. As this requires significantly more intricate local mechanisms, we only provide a high-level description of the necessary tools and modifications, and refer to the full version.

Skeleton. A *skeleton* of a polyomino P is a connected, λ-scaled subpolyomino $S \subseteq P$ with $\lambda = \lfloor \zeta(P)/4 \rfloor$, as illustrated in Figure 7a. Such a skeleton can easily be determined as the union of all $\lambda \times \lambda$ squares in P that are aligned with the same $\lambda \times \lambda$ integer grid.

Watershed. The *watershed* of a skeleton tile t corresponds to the union of all $2\lambda \times 2\lambda$ squares in P that fully contain t, see Figure 7b. We will show that at least one such square always exists, and their union forms a convex polyomino with bottleneck at least λ.

To swap agents from $P \setminus S$ into the skeleton S, we exploit the watersheds of its tiles. We first prove the existence of a skeleton $S \subseteq P$ as above, and that its watersheds fully cover P. This allows us to apply techniques from Section 4.1 to arbitrary polyominoes.

▶ **Lemma 12** (⋆). *For any polyomino P with $\zeta(P) \geq 8$, we can identify a skeleton $S \subseteq P$ in polynomial time. The skeleton S is λ-scaled for $\lambda = \lfloor \zeta(P)/4 \rfloor$, and every $2\lambda \times 2\lambda$ square inside P contains at least one of its scaled tiles.*
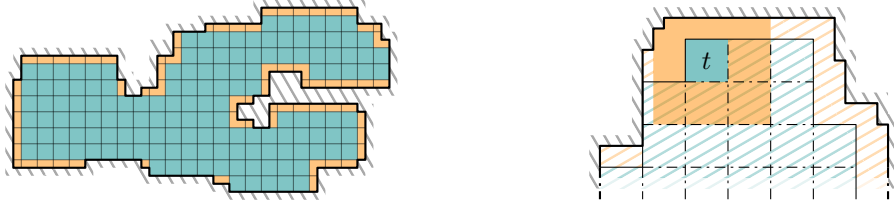
(a) A polyomino P and its skeleton S (cyan).

(b) An illustration of a skeleton tile t's watershed.

Figure 7 The central tools used in our proof of Theorem 15.

We make use of the following result obtained by Alpert et al. [6] for the routing number of convex grid graphs, where $w(P)$ and $h(P)$ refer to the width and height of P, respectively.

▶ **Theorem 13** (Alpert et al. [6])**.** *Let P be a connected convex grid piece. Then the routing number of P satisfies the bound $rt(P) \le C(w(P) + h(P))$ for some universal constant C.*

Finally, we demonstrate a method for efficient reconfiguration of an arbitrary skeleton tile's watershed in order to swap robots into and out of the skeleton.

▶ **Lemma 14.** *Given two configurations of a skeleton tile's watershed in a universally reconfigurable polyomino, we can efficiently compute an applicable schedule of makespan $\mathcal{O}(\lambda)$.*

Proof. Consider a $\lambda \times \lambda$ skeleton tile $t \subset P$ of a universally reconfigurable polyomino P, and let H refer to its watershed. Recall that we assume $\zeta(P) \ge 8$, so $\lambda = \lfloor \zeta(P)/4 \rfloor \ge 2$. As H is the union of all $2\lambda \times 2\lambda$ squares in P that contain t, it is thus universally reconfigurable and orthoconvex. We apply Theorem 13: Alpert et al. [6] presented a constructive proof in the form of an algorithm, which we can use to compute a routing sequence of length $\mathcal{O}(w(H) + h(H)) = \mathcal{O}(\lambda)$ between any two configurations of H, based on its dual graph. Such a routing sequence corresponds to a series of matchings in the dual graph of H that exchange tokens of adjacent vertices. Sequentially realizing these matchings by swapping adjacent agents as outlined in Lemma 4, we can arbitrarily reorder H in $\mathcal{O}(\lambda)$ transformations. ◀

This provides us with all necessary tools to prove the following generalization of Proposition 9, which, in turn, is a central tool for our proof of Theorem 8.

▶ **Theorem 15** (⋆)**.** *For any two configurations of a universally reconfigurable polyomino P of area n, we can compute an applicable schedule of makespan $\mathcal{O}(n/\zeta(P))$ in polynomial time.*

▶ **Theorem 8** (⋆)**.** *Given an instance of diameter d in a universally reconfigurable polyomino P, we can efficiently compute an applicable schedule of makespan $\mathcal{O}((d+\mu(P))^2/\zeta(P))$. This is asymptotically worst-case optimal for narrow instances.*

Proof sketch. Assuming that $c(P) < \zeta(P)$, our approach consists of three phases:

(I) We compute a skeleton $S \subset P$ which we split into patches S_i according to Phase (I) of Proposition 11. To each of these patches, we add cells of its skeleton tile's watersheds. This partitions P into patches P_i, each with a skeleton patch $S_i \subset S$ such that $S_i \subseteq P_i$.

(II) We use the rooted dual tree T of the skeleton patches S_i and, for each patch S_i with children $S_\ell, \dots, S_{\ell+k}$ according to T, we combine the patches $P_i, P_\ell, \dots, P_{\ell+k}$ to a (not necessarily connected) region F_i that can be reordered in $\mathcal{O}((d+\mu(P))^2/\zeta(P))$.

(III) Finally, we exploit these combined regions to place all agents at their destination.

These act analogously to Phases (I) – (III) of Proposition 11. ◀

5   Conclusions and future work

We provide a number of novel contributions for MULTI-AGENT PATH FINDING in simple polyominoes. We establish a characterization for the existence of reconfiguration schedules, based on different shape parameters of the bounding polyomino. Furthermore, we establish algorithmic methods that achieve worst-case optimal stretch for any instance in which the polyomino's bottleneck does not exceed the instance's diameter by more than a constant factor. There are a variety of directions in which these insights should give rise to further generalizations and applications.

Non-simple polyominoes.   Our results regarding universal reconfigurability are directly applicable to non-simple polyominoes. As noted in Section 2, the geometric characterization for simple polyominoes is formed as a special case based on the dual graph of a polyomino.

For any non-simple polyomino that is either 2-scaled or 2-square-connected, Theorem 1 and Proposition 9 are also directly applicable. The same is not true for Theorem 15, as our definition of the bottleneck based on cuts does not work in this case. However, with a separate definition that accounts for the minimal distance between inner and outer boundaries, Theorem 15 may be applicable.

Permutation routing.   Our results can be generalized to solid grid graph routing, which is a generalization of the findings of Alpert et al. [6]. We provided results regarding bounded stretch for this setting, therefore tackling a special case of their Open Question 2.

Further questions.   Our work is orthogonal to that of Demaine et al. [13, 14]: Their setting considered domains of large depth in conjunction with large bottleneck, i.e., the case that $\mu(P) \in \Omega(d)$ and $\zeta(P) \in \Omega(d)$. We establish asymptotically worst-case optimal results for narrow domains, which implies that $\mu(P) \in \mathcal{O}(d)$ and $\zeta(P) \in \mathcal{O}(d)$. In particular, instances where $\zeta(P) \in o(d)$ while $\mu(P) \in \omega(d)$, i.e., instances in which the gap between bottleneck and depth is unbounded relative to d, remain a challenge even for simple domains. We conjecture that this question for simple domains is equivalent to that of bounded stretch for non-simple domains with limited depth; considering an instance of large depth, we can create an analogous non-simple instance that features regularly distributed, small holes based on some grid graph. This may motivate research into the special case of instances in which the diameter is less or equal to the circumference of the smallest hole in the domain.

─── **References** ───

1   Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Transactions on Automation Science and Engineering*, 12(4):1309–1317, 2015. `doi:10.1109/TASE.2015.2470096`.
2   Pankaj K. Agarwal, Tzvika Geft, Dan Halperin, and Erin Taylor. Multi-robot motion planning for unit discs with revolving areas. *Computational Geometry: Theory and Applications*, 114:102019, 2023. `doi:10.1016/J.COMGEO.2023.102019`.
3   Pankaj K. Agarwal, Dan Halperin, Micha Sharir, and Alex Steiger. Near-optimal min-sum motion planning for two square robots in a polygonal environment. In *Symposium on Discrete Algorithms (SODA)*, pages 4942–4962, 2024. `doi:10.1137/1.9781611977912.176`.
4   Oswin Aichholzer, Erik D. Demaine, Matias Korman, Anna Lubiw, Jayson Lynch, Zuzana Masárová, Mikhail Rudoy, Virginia Vassilevska Williams, and Nicole Wein. Hardness of token swapping on trees. In *European Symposium on Algorithms (ESA)*, pages 3:1–3:15, 2022. `doi:10.4230/LIPICS.ESA.2022.3`.

5   Noga Alon, Fan R. K. Chung, and Ronald L. Graham. Routing permutations on graphs via matchings. *SIAM Journal on Discrete Mathematics*, 7(3):513–530, 1994. `doi:10.1137/S0895480192236628`.

6   H. Alpert, R. Barnes, S. Bell, A. Mauro, N. Nevo, N. Tucker, and H. Yang. Routing by matching on convex pieces of grid graphs. *Computational Geometry*, 104:101862, 2022. `doi:10.1016/j.comgeo.2022.101862`.

7   Indranil Banerjee and Dana Richards. New results on routing via matchings on graphs. In *Fundamentals of Computation Theory (FCT)*, pages 69–81, 2017. `doi:10.1007/978-3-662-55751-8_7`.

8   Bahareh Banyassady, Mark de Berg, Karl Bringmann, Kevin Buchin, Henning Fernau, Dan Halperin, Irina Kostitsyna, Yoshio Okamoto, and Stijn Slot. Unlabeled multi-robot motion planning with tighter separation bounds. In *Symposium on Computational Geometry (SoCG)*, pages 12:1–12:16, 2022. `doi:10.4230/LIPICS.SOCG.2022.12`.

9   Marc Baumslag and Fred S. Annexstein. A unified framework for off-line permutation routing in parallel networks. *Mathematical Systems Theory*, 24(4):233–251, 1991. `doi:10.1007/BF02090401`.

10   Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Matthias Konitzny, Lillian Lin, and Christian Scheffer. Coordinated motion planning: The video. In *Symposium on Computational Geometry (SoCG)*, pages 74:1–74:6, 2018. `doi:10.4230/LIPICS.SOCG.2018.74`.

11   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. `doi:10.1109/TRO.2018.2857475`.

12   Loïc Crombez, Guilherme Dias da Fonseca, Yan Gerard, Aldo Gonzalez-Lorenzo, Pascal Lafourcade, and Luc Libralesso. Shadoks approach to low-makespan coordinated motion planning. *ACM Journal of Experimental Algorithmics*, 27:3.2:1–3.2:17, 2022. `doi:10.1145/3524133`.

13   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 *Symposium on Computational Geometry (SoCG)*, pages 29:1–29:15, 2018. `doi:10.4230/LIPICS.SOCG.2018.29`.

14   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. `doi:10.1137/18M1194341`.

15   Eduard Eiben, Robert Ganian, and Iyad Kanj. The parameterized complexity of coordinated motion planning. In *Symposium on Computational Geometry (SoCG)*, pages 28:1–28:16, 2023. `doi:10.4230/LIPICS.SOCG.2023.28`.

16   Sándor P. Fekete, Phillip Keldenich, Ramin Kosfeld, Christian Rieck, and Christian Scheffer. Connected coordinated motion planning with bounded stretch. *Autonomous Agents and Multi-Agent Systems*, 37(2):43, 2023. `doi:10.1007/S10458-023-09626-5`.

17   Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing coordinated motion plans for robot swarms: The CG:SHOP challenge 2021. *ACM Journal of Experimental Algorithmics*, 27:3.1:1–3.1:12, 2022. `doi:10.1145/3532773`.

18   Sándor P. Fekete, Ramin Kosfeld, Peter Kramer, Jonas Neutzner, Christian Rieck, and Christian Scheffer. Coordinated motion planning: Multi-agent path finding in a densely packed, bounded domain, 2024. `doi:10.48550/arXiv.2409.06486`.

19   Sándor P. Fekete, Peter Kramer, Christian Rieck, Christian Scheffer, and Arne Schmidt. Efficiently reconfiguring a connected swarm of labeled robots. *Autonomous Agents and Multi-Agent Systems*, 38(2):39, 2024. `doi:10.1007/S10458-024-09668-3`.

20   Lenwood S. Heath and John Paul C. Vergara. Sorting by short swaps. *Journal of Computational Biology*, 10(5):775–789, 2003. `doi:10.1089/106652703322539097`.

21   John E. Hopcroft, Jacob T. Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the warehouseman's problem. *International Journal of Robotics Research*, 3(4):76–88, 1984. `doi:10.1177/027836498400300405`.

22   John E. Hopcroft and Gordon T. Wilfong. Reducing multiple object motion planning to graph searching. *SIAM Journal on Computing*, 15(3):768–785, 1986. `doi:10.1137/0215055`.

23   Jun Kawahara, Toshiki Saitoh, and Ryo Yoshinaka. The time complexity of permutation routing via matching, token swapping and a variant. *Journal of Graph Algorithms and Applications*, 23(1):29–70, 2019. `doi:10.7155/jgaa.00483`.

24   Paul Liu, Jack Spalding-Jamieson, Brandon Zhang, and Da Wei Zheng. Coordinated motion planning through randomized *k*-opt. *ACM Journal of Experimental Algorithmics*, 27:3.4:1–3.4:9, 2022. `doi:10.1145/3524134`.

25   John M. Marberg and Eli Gafni. Sorting in constant number of row and column phases on a mesh. *Algorithmica*, 3:561–572, 1988. `doi:10.1007/BF01762132`.

26   Tillmann Miltzow, Lothar Narins, Yoshio Okamoto, Günter Rote, Antonis Thomas, and Takeaki Uno. Approximation and hardness of token swapping. In *European Symposium on Algorithms (ESA)*, pages 66:1–66:15, 2016. `doi:10.4230/LIPICS.ESA.2016.66`.

27   Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014. `doi:10.1126/science.1254295`.

28   Erol Şahin and Alan F. T. Winfield. Special issue on swarm robotics. *Swarm Intelligence*, 2(2-4):69–72, 2008. `doi:10.1007/s11721-008-0020-6`.

29   Jacob T. Schwartz and Micha Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. *International Journal of Robotics Research*, 2(3):46–75, 1983. `doi:10.1177/027836498300200304`.

30   Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *International Journal of Robotics Research*, 35(14):1750–1759, 2016. `doi:10.1177/0278364916672311`.

31   Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Symposium on Combinatorial Search (SOCS)*, pages 151–158, 2019. `doi:10.1609/SOCS.V10I1.18510`.

32   Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9–19, 2008. `doi:10.1609/aimag.v29i1.2082`.

33   Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping labeled tokens on graphs. *Theoretical Computer Science*, 586:81–94, 2015. `doi:10.1016/J.TCS.2015.01.052`.

34   Hyeyun Yang and Antoine Vigneron. Coordinated path planning through local search and simulated annealing. *ACM Journal of Experimental Algorithmics*, 27:3.3:1–3.3:14, 2022. `doi:10.1145/3531224`.

35   Jingjin Yu and Daniela Rus. Pebble motion on graphs with rotations: Efficient feasibility tests and planning algorithms. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 729–746, 2015. `doi:10.1007/978-3-319-16595-0_42`.