



Sweeping a Domain with Line-Of-Sight Between Covisible Agents

Kien C. Huynh 

Communications and Transport Systems, ITN, Linköping University, Sweden

Joseph S. B. Mitchell 

Department of Applied Mathematics and Statistics, Stony Brook University, NY, USA

Valentin Polishchuk 

Communications and Transport Systems, ITN, Linköping University, Sweden

Abstract

We consider sweeping a polygonal domain using variable-length segments whose endpoints can be considered to be mobile agents moving with bounded speeds; a point in the domain is swept when it belongs to one of the segments. The objective is to sweep the domain as quickly as possible. We show that the problem is NP-hard even in simple polygons and even for a single segment (two agents), and give constant-factor approximation algorithms, both for simple polygons and polygons with holes.

Our approximations are obtained by introducing a new type of “window partition” of the polygon, which may find other applications. For domains with holes, our results are based on a non-trivial topological argument proving a surprising fact: a connected subset of the domain, whose points are swept but not directly touched by the agents, may contain at most one hole.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Polygon sweeping, collaborating agents, motion coordination, makespan optimization

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.39

Supplementary Material

Software (Source Code): https://github.com/KienHuynh/polygon_sweeping [20]

Funding This work is partially supported by the Swedish Research Council and the Swedish Transport Administration.

Acknowledgements We thank the anonymous reviewers for their helpful comments.

1 Introduction

We study sweeping a polygonal domain using mobile line segments whose lengths can change. Consider $2k$ agents starting at a given point s (the depot or the base, or the door through which the agents enter) on the boundary of a polygonal domain P . Each agent moves within P , with maximum speed 1. (We do not impose an acceleration bound.) The agents form pairs and each pair a, b defines a sweeping segment whenever the agents see one another; when this is the case, the points along the segment ab are seen by both agents, and we say that the points of the segment ab have been *swept*. The goal is to compute motions (trajectories) for the agents, such that all of P is swept and the agents return to the depot as soon as possible, i.e., minimizing the *makespan* of the sweep, or the time when the agents return to the base after all points of P are swept. See Figure 1 and videos [1, 2].

Search and surveillance are recurring applications of computational geometry: sweeping geometric domains is both a fundamental (in particular, minimum-length sweeping with a line is Problem CG-Open26 in the $\exists\mathbb{R}$ complexity class compendium [34]) and a practical



© Kien C. Huynh, Joseph S. B. Mitchell, and Valentin Polishchuk;
licensed under Creative Commons License CC-BY 4.0

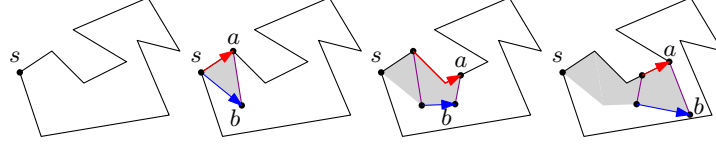
19th International Symposium on Algorithms and Data Structures (WADS 2025).

Editors: Pat Morin and Eunjin Oh; Article No. 39; pp. 39:1–39:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** A few steps in the sweeping: white is unswept, gray is swept. Agents a and b start at a point $s \in \partial P$; during the sweep, the agents are not required to stay on the boundary. The red and blue arrows show the trajectories of a and b , respectively.

(e.g., shaving is sweeping the skin area with a segment) problem, which has been studied for decades. Sweeping with a disk or a square is known as milling/lawnmowing [5, 8, 18], and sweeping with a visibility polygon of a mobile observer(s) is the *watchman route problem* (WRP) [10, 29, 30]. Both in WRP and in our problem, the sweep can catch (see) any static target in P , while a mobile target can escape being seen. This is different from, perhaps, a more standard notion of sweeping (by agents with no speed bounds), which ensures that swept points cannot be “recontaminated”: here, the archetypal problem is 2-walkability [9, 17, 21, 37] (Can 2 guards sweep a simple polygon by walking along its boundary while always seeing each other?), which was generalized to sweeping with a curve [25, 28, 33] and a chain of guards [12], also in terrains [13, 14]. (Problems in which the target, usually non-static, needs to be “touched,” not only seen, in order to be caught, are known as pursuit-evasion games, cops and robbers, lion and man, etc. [3, 11, 16, 22–24, 26, 32, 36].) Sweeping problems may be viewed as far-going extensions of computing the Frechét distance [4] – a classical motion coordination problem. Our solutions apply also to the Laser Tripwire Sensor Deployment for indoor localization and tracking (tripwire localization methods are common and well studied [6, 27]): suppose you have 2 robots that want to deploy a set of tripwire sensors (pairs of points that see each other) – the robots stop every so often (at step length δ) to place the pairs of sensors, and the goal is to complete the deployment as soon as possible.

Our Contributions

We formulate and give the first study of the algorithmic complexity and approximation of the sweeping problem with variable-length line segment(s):

- **Problem 1.** Given a polygonal domain P , an agent depot $s \in \partial P$, and k pairs of agents: find a motion plan to sweep all points in P and minimize the total time (makespan). Here, each pair of agents have to be covisible at all times (the line segment connecting the pair must stay within P). We call this the “unbreakable” constraint. This problem has two versions:
 - **SWEEPWITHRETURN:** The agents have to return to s (we call this the default version).
 - **SWEEPWITHNORETURN:** The agents can end anywhere in P .
- **Problem 2.** The setting and objective are similar to Problem 1. However, in this problem, we do not require the agents to always be covisible but they can only sweep something in P if they become covisible. In addition, the agents themselves also need to stay within P . We call this the **BREAKABLESEGMENTSWEeping** problem.

The results in this paper can be seen as contributing to the broader class of optimal coordinated motion planning problems involving mobile agents moving within a domain, with constraints (e.g., covisibility), and an objective to accomplish a covering task.

Specifically,

- Section 2 provides a proof of NP-hardness (from 3-PARTITION) for all versions, even in the case of an orthogonal, x -monotone simple polygon P (and even if the sweeping segment is required to be vertical at all times).
- Section 3 presents a constant-factor approximation for sweeping simple polygons for Problem 1. Our main technical contribution is the solution for the case of $k = 1$ segment, based on a “window” partition of P that we work out specifically for our purposes (we believe the worked-out decomposition may find also other applications, e.g., when considering similar search problems in terrains); the solution for $k > 1$ (Section 3.2) is obtained by assigning different parts of P to be swept by different pairs of agents.
- In Section 4 the approximation for 1 segment is extended to polygons with holes. The algorithm is simple: turn the domain into a simple polygon by bridging together the holes and the outer boundary of the domain; then apply the algorithm for simple polygons. However, proving that this is an $O(1)$ -approximation (i.e., giving a lower bound) is highly non-trivial (and crucially uses the fact that the segment is unbreakable). We leave open sweeping with $k > 1$ segments in polygons with holes.
- Section 5 considers Problem 2. We prove that in a simple polygon relaxing the covisibility constraint does not improve the makespan by more than a factor of 2, and we show that this bound is tight; thus, our constant-factor approximation algorithm for the unbreakable case applies for the breakable case as well. (We note that our NP-hardness proof (Section 2) for unbreakable sweeping of a simple polygon P applies as well to breakable sweeping of P .) In the case that P has holes, we give a polylog approximation algorithm in the special case that P is a polyomino and we sweep with a vertical segment.

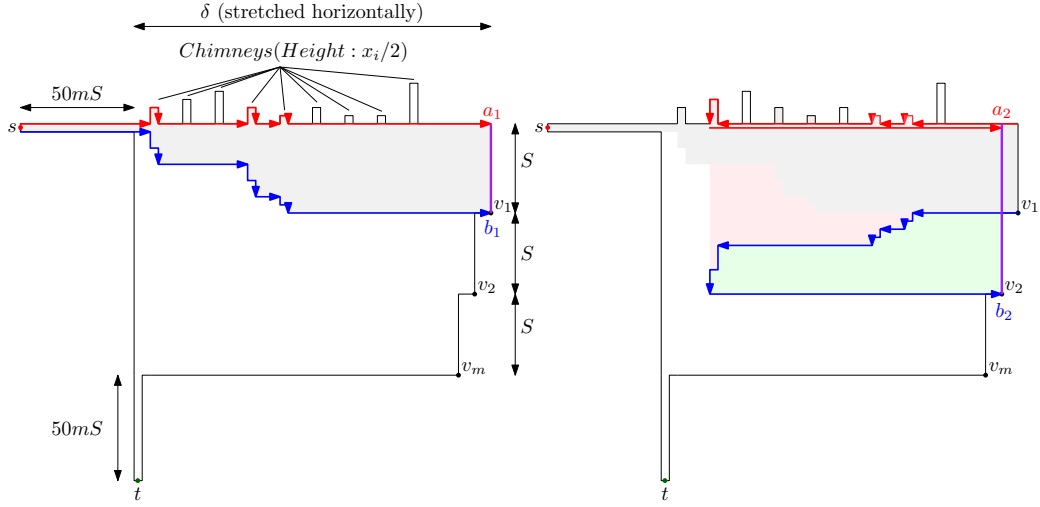
In addition to the above results, we also implemented the algorithm described in Section 3 and compared it with a heuristic algorithm on randomly generated data which we show in Appendix D. The codes of the algorithms can be found at [20].

2 NP-Hardness

We show that even in the very special case of sweeping an orthogonal, monotone simple polygon with 2 agents a and b , computing an optimal sweeping schedule is strongly NP-hard. This is in contrast with sweeping a simple polygon with a visibility polygon (WRP), which has a polynomial-time exact algorithm [10, 30], and the sweeping of a simple polygon with a disk or square (the milling problem), which is conjectured to have a polynomial-time exact algorithm [8], as the closely related problems of determining Hamiltonicity of a simple triangular or square grid graph are decidable in polynomial time [7, 40].

► **Theorem 1.** *It is NP-hard to compute an optimal sweeping for SWEEPWITHNORETURN, even if P is an x -monotone, orthogonal simple polygon.*

Proof. We first consider the SWEEPWITHNORETURN version where the agents do not have to return to the depot s (so we are minimizing the time when the last point of P is swept). We then show how to modify the reduction to prove hardness of the SWEEPWITHRETURN version in which the agents must return to s (and the time of the return defines the makespan). Given an instance of 3-PARTITION (Can given integers $x_1 \dots x_{3m}$ be partitioned into m triples with equal sums?), we construct a polygon P with $O(m)$ vertices, as shown in Figure 2, consisting of a tall and very narrow rectangular room (whose width is exaggerated in the figure, to show details) with a staircase on the right and with two corridors (each of which



■ **Figure 2** Reducing 3-PARTITION to sweeping. On the left: the trajectories of the two agents (red and blue arrows) depicting their motions to sweep the first staircase step. After the top agent has swept 3 chimneys corresponding to an arbitrary triple in the 3-PARTITION solution, they will sweep the first step (including v_1) and end at a_1b_1 (purple). On the right: the motions of the agents in order to sweep the next step. First, the agents will move from a_1b_1 to the left to sweep another triple of chimneys (new areas swept during this motion are painted pink). Then, the agents move to the right to sweep the second staircase step (and v_2) and stop at a_2b_2 , the new area swept this time is colored green. Similarly, step m is swept after the top agent has visited the m -th triple.

has length $50mS$) extending from the room as shown. The height of the room is mS where $S = \sum x_i/m$ is the target sum in each of the sought triples of the given numbers; the width, δ , is very small (e.g., $\delta < 1/m^{10}$).

The corridors ending with s and t are very thin and long ($50mS$). Thus, in any optimal sweeping schedule each of the corridors is swept exactly once: the sweep starts at s (by definition) and ends with one of the agents at t (recall that for now we do not restrict where the agents are at the conclusion of the sweep). The top of the room has a chimney for each number in the set. The chimneys are very narrow ($\text{width} < \frac{\delta}{3m} < \frac{1}{3m^{11}}$) and chimney i has height $x_i/2$. Finally, the height of each step in the staircase is S .

If the 3-PARTITION instance is feasible, then P can be swept as follows (with both a and b moving at the maximum, unit speed, and the segment ab staying vertical throughout the motion): the top agent a will visit and ascend chimneys corresponding to triples in the solution. While a is in chimney i (either ascending or descending), the bottom agent b moves down by x_i . After a finishes a triple that sums to S , b has moved down by S . (Note that apart from the duration during which the segment sweeps the horizontal corridor, horizontal movements are not particularly relevant for the analysis, since the room is so narrow, and chimneys are even narrower, implying that the sweep segment must be nearly vertical at all times.) Finally, in this first pass, ab moves all the way to the right, arriving with length S so that it is just long enough to sweep the top step of the staircase (including v_1) without spending further time having agents move vertically. See the left of Figure 2 for an example of this motion to sweep the first step. Next, ab will move to the left of the polygon and sweep more chimneys before moving back to the right to sweep v_2 (Figure 2, right). Such sideways sweeps are repeated until all m of the stairs are swept. (Again, all horizontal motion of agents is not relevant to the overall makespan, since m left-to-right motions are still much less than length 1.) The time to sweep the room, along with the staircases, is at most $mS + 2m\delta$. The total time to sweep everything is $101mS + 2m\delta$.

Conversely, suppose the stairs are swept in any other way, then either b has to wait for at least a duration of 1 before aligning with a stair tread, or a has to waste time by not going into a chimney with a single up-and-down pass. The first case occurs when the agents pick a triplet of chimneys (x, x', x'') with a sum larger than S , then agent a still needs to trace the boundary of the chosen chimneys to sweep them completely. Even if b can already reach the depth of the next staircase vertex (say, v_j), it must still wait for a to finish sweeping the last chimney in the triplet so that it can finally move to sweep v_j , which takes longer than S . In the second case, the chosen triplet of chimneys has a sum smaller than S . In this case, agent a will have to wait for agent b to reach the depth of the next staircase vertex. This still costs the agents a time of S while leaving taller chimneys for other triplets (which leads back to the previous case). Note that there is no reward for a to partially sweep a chimney, since a must exit the chimney and align vertically with b when b touches v_j so that the both of them can sweep the vertical edge incident to v_j . Agent b can technically move diagonally downward to save time, but in that case, the most it can save is δ , which is chosen to be much smaller than $1/m^{10}$ (so in total it can save at most $m \cdot 1/m^{10} = 1/m^9 \ll 1$).

We now modify the above gadget to show the hardness of our SWEEPWITHRETURN version. See Appendix A for the modified gadget. As in the proof for the version without return to s , the segment will sweep the top chimneys (solving the 3-PARTITION) and sweep the convex vertices on the right in parallel. Once the segment is done with the right convex vertices and the top chimneys, the segment sweeps the inverted chimneys (also solving the same 3-PARTITION instance); each time it sweeps 3 bottom chimneys that make a sum S , the segment will retract itself (top endpoint moving down) so that it could reach the convex vertices on the left (v'_1, v'_2, \dots). The overall makespan will then be $3mS + O(\delta)$. As in the reduction for the version without return to s , any deviation from the described schedule incurs a wait of at least 1 by at least one of the agents. ◀

3 Approximation Algorithm for Simple Polygons

We begin with a simple observation (see Appendix B for the proof):

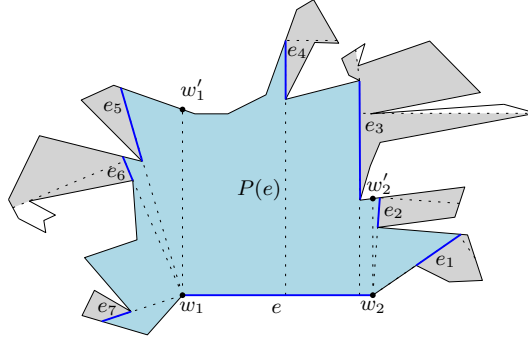
► **Lemma 2.** *Every convex vertex of P must be visited by at least one agent.*

We first consider the case when there are two agents, a and b , starting at the same point $s \in P$: we present an $O(1)$ -approximation algorithm to sweep any simple polygon using an unbreakable segment. As with our hardness proof (Section 2), we will first provide algorithmic results for the SWEEPWITHNORETURN version. We show that in linear time, one can find a schedule, for a single unbreakable segment, with makespan $16 \cdot OPT$ where OPT is the optimal makespan. Let $|x|$ denote the Euclidean length of any 1D structure. From Lemma 2,

► **Lemma 3.** $OPT \geq \frac{1}{4}|\partial P|$

Proof. By Lemma 2, the union of paths for a and b visits all convex vertices; thus, the union is at least as long as any path within P visiting all convex vertices, implying that the union cannot be shorter than $\frac{1}{2}|\partial P|$. This is because in a simple polygon, ∂P is the shortest tour that visits all convex vertices, and any path that does so must be at least half as long. ◀

Our algorithm partitions the polygon into subpolygons, each of which has a simple sweeping strategy. The dual of the partition is a tree, which allows the line segment to sweep all the subpolygons, traversing the tree recursively.



■ **Figure 3** The light blue region is the special visibility polygon $P(e)$, with the base edge $e = w_1w_2$. The additional chords, e_1, e_2, e_3, \dots also have visibility polygons (shaded gray) associated with them.

The details of the partition are as follows. Consider any edge $e = w_1w_2$ of the polygon. We first compute the histogram polygon with base e , $HP(e)$, which is defined to be the union of all interior chords perpendicular to e and having an endpoint on e (a chord is a line segment connecting two points of ∂P and stays completely within P). Let (w_1, w'_1) be such a chord anchored at w_1 . We then compute the visibility polygon of w_1 restricted to points that are left of (w_1, w'_1) ; call the polygon $VP(w_1)$. Similarly, we compute another visibility polygon to the right of (w_2, w'_2) , called $VP(w_2)$. Then, $P(e) = HP(e) \cup VP(w_1) \cup VP(w_2)$ is one subpolygon in our partitioning scheme. See Figure 3 for an example of $P(e)$. Notice that the boundary of $P(e)$ includes some new “shadow chords,” each of which becomes a histogram base for subsequent subpolygons in the partition. We will order these chords counterclockwise. The process is complete when every shadow chord that has been generated has been utilized as a histogram base for a subpolygon adjacent to the subpolygon that generated the shadow chord. For the first subpolygon, we will simply use the visibility polygon of s . The overall partitioning of P can be done in linear time, utilizing a triangulation of P (similar to computing window partition trees [35] for link distances in simple polygons).

We now describe the recursive process to sweep a subpolygon $P(e)$ and its descendants, for an edge $e = w_1w_2$ of P (refer to Figure 3). The agents start at the endpoint w_1 of e ; once the process is completed, i.e. when the segment finishes sweeping $P(e)$ and its descendants, both agents return to w_1 . First, a sits at w_1 while b travels clockwise around the boundary of $VP(w_1)$ until reaching w'_1 ; this way the rotating segment ab sweeps $VP(w_1)$. Next, $HP(e)$ is swept by the segment perpendicular to e : b travels along the top polygonal path of the histogram while a moves along e (a slows itself in order that ab remains perpendicular to e). Then, the segment rotates clockwise around w_2 to sweep $VP(w_2)$; at this point $P(e)$ is completely swept and the segment will move to the furthest (w.r.t. w_2) endpoint of the first “shadow chord” (see e_1 in Figure 3). After the segment completely sweeps $P(e_1)$ (and its descendants), it will move to the next child of $P(e)$. This will be repeated until all descendants of $P(e)$ are swept. Notice that while sweeping the children, the segment will at times move counterclockwise along the boundary of $P(e)$. Once the segment is done with all children of $P(e)$, both of its endpoints will regroup at w_1 .

► **Lemma 4.** *The sweeping algorithm above produces a schedule with makespan $\leq 16 \cdot OPT$.*

Proof. We will first perform analysis on any subpolygon $P(e)$ other than the root. Without loss of generality, we assume that the segment always starts sweeping from the left vertex of e (i.e., w_1 in the figure). First, the endpoints sweep $P(e)$ by traveling from w_1 to w_2 , this will incur a cost of $|\partial P(e)| - |e|$. This is because a will move along e with a speed at most

that of b . Second, the endpoints travel counterclockwise to visit all children of $P(e)$. The cost of this counterclockwise traversal is at most $|\partial P(e)| - |e|$. Overall, the cost of sweeping $P(e)$ and moving counterclockwise to visit all of its children is

$$2(|\partial P(e)| - |e|) = 2\left(|\partial P(e) \cap \partial P| + \sum_{e_i \in C(e)} |e_i|\right),$$

where $\partial P(e) \cap \partial P$ are the portions of $\partial P(e)$ that belong to the boundary of P , and $C(e)$ are the base edges of children of $P(e)$. For the root subpolygon $P(s)$, the cost to sweep it and visit all of its children is at most

$$2|\partial P(s)| = 2\left(|\partial P(s) \cap \partial P| + \sum_{e_i \in C(s)} |e_i|\right).$$

Here, $C(s)$ is the set of all chords generated and associated with $P(s)$.

Let E be the set of all additional chords created during the partition process. The total cost to sweep P will be the sum of the two terms above:

$$2\left(\sum_{e \in E} (|\partial P(e) \cap \partial P| + |e|) + |\partial P(s) \cap \partial P|\right) \leq 4 \sum_{e \in E} |\partial P(e) \cap \partial P| + 2(|\partial P(s) \cap \partial P|).$$

We have $|e| \leq |\partial P(e) \cap \partial P|$ for each $e \in E$ because $\partial H_P(e) \cap \partial P \subseteq \partial P(e) \cap \partial P$ is an orthogonal projection of e onto ∂P . Thus overall, the sweeping cost of P is at most

$$4\left(\sum_{e \in E} |\partial P(e) \cap \partial P| + |\partial P(s) \cap \partial P|\right) \leq 4|\partial P| \leq 16 \cdot OPT.$$

The first inequality holds because the portions of each $P(e)$ that belong to ∂P , i.e. $\partial P(e) \cap \partial P$, are pairwise disjoint for different $e \in E$. Thus, $\sum_{e \in E} |\partial P(e) \cap \partial P| + |\partial P(s) \cap \partial P| = |\partial P|$. The second inequality follows directly from Lemma 3. \blacktriangleleft

3.1 Sweeping with return (default version)

The algorithm above works also when a and b must return to s . The only modification is that at the end of the schedule, both agents must return to s . The analysis remains the same, except that the lower bound of Lemma 3 becomes $OPT \geq \frac{1}{2}|\partial P|$.

► **Theorem 5.** *A sweeping schedule, for a single unbreakable segment, with makespan $8 \cdot OPT$ can be found in linear time.*

3.2 Sweeping with multiple segments with return

Suppose now that we have $2k > 2$ agents who start and return to the depot s (the makespan is when the last agent returns to the depot; for $k > 1$ we do not have a solution for the version in which the agents are not required to return to s). Here we have two lower bounds:

- $OPT \geq |\partial P|/(2k)$ where $|\partial P|$ is the perimeter – this follows from Lemma 2 (the $2k$ agents have to visit all convex vertices).
- Let d_{\max} be the geodesic distance from the depot s to the convex vertex furthest from s ; then $OPT \geq 2d_{\max}$ – an agent has to visit this furthest convex vertex and go back to the depot.

Partition P into k parts as follows: Pick $k - 1$ points on the boundary of P so that these $k - 1$ points, together with s , split the boundary of P into k parts of equal perimeter. From the depot, draw the geodesic shortest path to each of these $k - 1$ points; the paths decompose P into subpolygons. (Some subpolygons may have very small areas, since the geodesic paths might collapse onto themselves and share edges.)

Because each drawn shortest path is no longer than d_{\max} , each subpolygon has perimeter at most $2d_{\max} + |\partial P|/k$. We now simply reuse the approximation algorithm for the single segment case: there, the makespan is at most 4 times the perimeter of the polygon, so for k segments, the makespan will be $4(2d_{\max} + |\partial P|/k) \leq 4 \cdot OPT + 8 \cdot OPT = 12 \cdot OPT$.

4 Approximation Algorithm for Polygonal Domains with Holes

If the domain P has holes, for sweeping with 1 segment ab , we can leverage the result for simple polygons from the preceding section, provided we can reduce the multiply connected case to the simply connected case, via an appropriate bridging of holes to each other and to the outer boundary of P . The main technical challenge is proving lower bounds (Lemma 10) on OPT that enable this approach to yield the desired result – a constant-factor approximation. The version that we shall describe in this section is `SWEEPWITHRETURN`, but the approximation algorithm will also work for `SWEEPWITHNORETURN` (with a larger multiplicative factor).

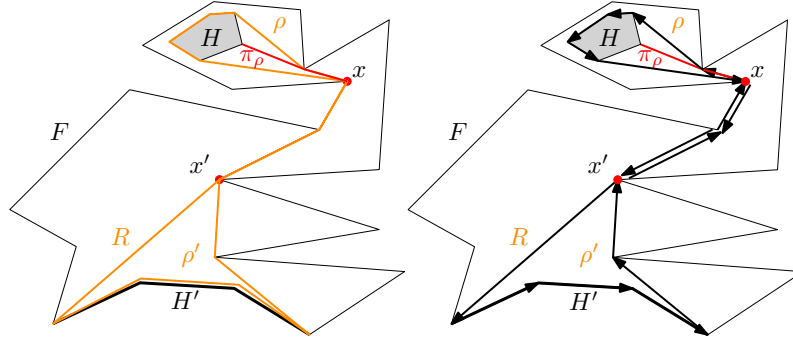
First, we compute an approximation of the minimum Steiner tree, MST^* , within P that spans all connected components of the boundary ∂P of P (both the holes and the outer boundary). This can be done using the PTAS in [31]. The computed tree and the boundaries can be viewed as bounding a degenerate simple polygon P' (these polygons are sometimes called “weakly” simple; the boundary cycle of such polygons may go through some points more than once). In the second phase, we apply the algorithm from Section 3 to compute a sweeping solution for P' with makespan $O(|\partial P'|)$, where $\partial P'$ includes the boundaries of the holes and the outer boundary of P , as well as the edges of the spanning tree.

To prove a lower bound of $\Omega(|\partial P'|)$ on OPT , we will need some definitions. A vertex of P is *convex* or *reflex* depending on whether its angle *interior to P* is less than or greater than π . Note that all vertices of a convex hole are actually reflex vertices of P . A *maximal reflex chain* is a chain on ∂P between two consecutive convex vertices; in particular, any edge with two convex endpoints is a maximal reflex chain. Unless otherwise stated, all reflex chains we consider will be maximal reflex chains. Note that a convex hole has no reflex chain.

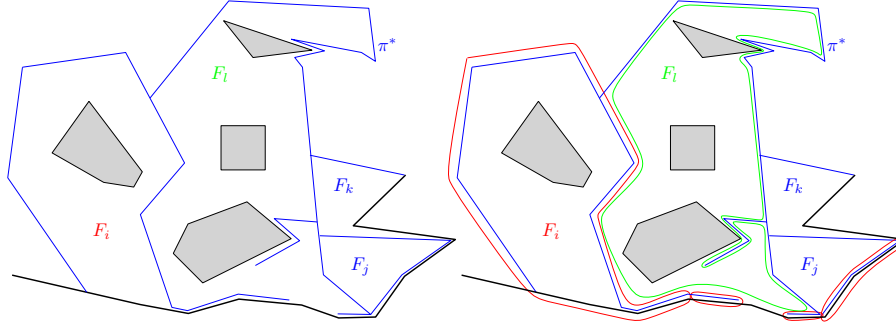
Given a polygonal domain F , a subset $R \subseteq F$ is *geodesically (or relatively) convex* if for any points $s, t \in R$ the shortest s - t path (within F) lies fully within R . The geodesic (or relative) convex hull R of a set S in F is the smallest geodesically convex set containing S . We will use GCH as a shorthand for geodesic convex hull. Reflex vertices of the hull are reflex vertices of F .

We use *obstacle* to denote a convex hole or a reflex chain. If we need to indicate that we are working with holes or reflex chains, we will use the specific term for them. We say that a set $S(\subseteq F)$ *contains* some obstacles H_1, \dots, H_m , if $S \cup (\bigcup_{i=1}^m H_i) \subseteq S$ (This containment definition is relaxed from the usual set inclusion definition: since H_i could be a hole and the interior of it does not belong to F , S is not actually a superset of H_i). Similarly, we say $R(\subseteq F)$ is a GCH of some obstacles if R is the smallest geodesically convex set containing all of these obstacles. Note that if H_i is a reflex chain and $H_i \subseteq R$ then $H_i \subseteq \partial R$.

The interior of a GCH needs not to be connected: the boundary of the hull may visit a reflex vertex x of F more than once (so the GCH is a weakly simple polygon) – at some of the visits x is a convex vertex of the hull, while at others it is a reflex vertex (Figure 4,



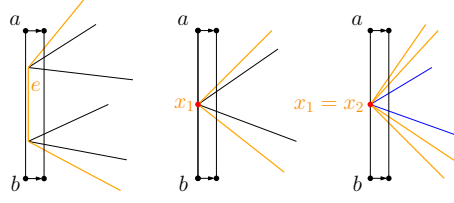
■ **Figure 4** Left: the geodesic convex hull R (drawn in orange) of two obstacles H and H' in a polygonal domain F . H is a hole and H' is a reflex chain. x and x' are collapsing vertices of R . Right: the PSLG constructed by walking around R counterclockwise has two non-empty cycles, which we label as pockets ρ and ρ' .



■ **Figure 5** Left: Black is part of ∂P (outer boundary or a non-convex hole), blue is π^* , and gray are convex holes; where blue touches black or gray, they are shown slightly separated for visual clarity. Right: Degenerate faces are circled with red: some faces, e.g., those consisting of a reflex chain followed by π^* have area 0; others, like F_i , have some degenerate parts where π^* follows ∂P . The boundary of F_ℓ (also a degenerate face) is delineated with green; the face contains 3 convex holes, but only one of them (the square) is not touched by blue (for the other two, the blue is slightly pulled away from the holes, for visual clarity).

left). When this occurs, we say that the hull *collapses* onto itself at x and x is a *collapsing* vertex. Consider the directed, planar straight-line graph (PSLG) formed by walking around the boundary of a GCH counterclockwise (Figure 4, right). When the GCH collapses, it will form more than one cycle in the PSLG (since at least one vertex is repeated during the walk). A cycle in this graph can be degenerate (i.e. a cycle from x to x' to x which has an empty area) or non-degenerate. If a region bounded by a cycle contains an obstacle, we will call it a pocket, denoted by ρ . Note that a degenerate cycle can actually be a pocket: it can contain an edge of some obstacle.

Let π^* be the union of the paths for the two segment endpoints, a and b , in an optimal solution. By Lemma 2, the union π^* of the two agents' paths touches all convex vertices of P . In particular, π^* , together with the reflex chains, partitions P into faces F_1, F_2, \dots (Figure 5, left) such that every convex vertex of every face has at least one incident edge from π^* . Note that in places where π^* follows (hugs) ∂P , some faces are degenerate; a degenerate face (a weakly simple polygon) is created also where π^* touches a convex hole or – more generally – a point of ∂P other than a convex vertex (See Figure 5, right).



■ **Figure 6** The (interior points of the) segment either enter the geodesic convex hull by aligning with its edge e (in which case the segment starts intersecting a hole when moving into the hull because e is supported by hole vertices), or by moving over a convex vertex x of R_i^1 . In the second case, if x is not a collapsing point of the hull, then it must be supported by a hole that ab will intersect after passing over x (middle figure).

The boundary ∂F_i of any face F_i consists of π^* , the reflex chains of ∂P , and convex holes of P ; the first two are the outer boundary of F_i . Note that only convex holes of P , if any, remain holes inside the faces; any non-convex hole is split into reflex chains that form boundaries of the faces (if a hole has only one convex vertex we treat it as two convex vertices connected by a degenerate, 0-length edge – so the hole contributes two reflex chains to the partition). For any maximal contiguous portion of ∂P in ∂F_i , it must be a reflex chain: if it contains a convex vertex of P then π^* would need to touch that vertex in order to sweep it.

Starting from here, when we say obstacles in F_i , we refer to convex holes or reflex chains of P in F_i . Let R_i be the GCH of all obstacles in a face F_i , we want to prove the following property of this GCH:

► **Lemma 6.** *If R_i contains at least 2 obstacles and does not collapse onto itself anywhere, the segment ab cannot sweep the interior of R_i .*

Proof. By definition of F_i , both endpoints of ab never enter the interior of F_i and R_i ; which means the interior of R_i must be swept by interior points of ab . Since the beginning of the schedule, a and b are both at s so they are outside of R_i (the intersection between the segment ab and the interior of R_i is empty). Consider the moment when some interior points of ab enter R_i (Figure 6): the segment is either aligned with an edge of the hull or touches a convex vertex x_1 of the hull. If ab is supported by a (necessarily reflex) vertex of P , then the segment will start intersecting the exterior of P as it moves. In order to *not* intersect with the exterior, the vertex x_1 must be supported by a reflex vertex of F_i , which may only happen if there is another (reflex) vertex x_2 of the hull collocated with x_1 . This is due to the fact that any convex vertex of a GCH is either a vertex of P , or is supported by F_i padded by another (reflex) vertex. Therefore, R_i must collapse at some vertex. ◀

Now we have the following result based on the fact that R_i has to collapse:

► **Lemma 7.** *If R_i has at least two obstacles and collapses onto itself at some vertices, then it will create at least two pockets.*

Proof. Note that by definition, a pocket must contain at least one obstacle. Let \mathcal{H} be the set of obstacles in R_i , and let ρ be the only pocket that R_i has, then ρ contains all obstacles in \mathcal{H} . Consider the PSLG created by walking along ∂R_i counterclockwise: since R_i collapses onto itself at some vertices, it will create more than one cycle in this graph. However, only one cycle (the one bounding ρ) has any obstacle in it, therefore the rest of the cycles in the PSLG are degenerate and empty. But if the PSLG contains all of these degenerate cycles, it means that R_i was not an actual GCH of the obstacles, since we can get rid of all of these degenerate cycles and only keep ρ to get a smaller set that contains all obstacles, which is a contradiction. Therefore, ρ cannot be the only pocket. ◀

Note that the above lemmas only state that R_i will collapse and create at least two pockets, each having at least one obstacle. Now, for each pocket ρ in R_i , we can reapply the same arguments: let R_ρ be the GCH containing all obstacles in ρ then R_ρ must also collapse onto itself somewhere, and create at least two pockets each containing some obstacles. We can keep repeating the same process and each time a GCH collapses it splits the set of obstacles into at least two subsets, each subset belonging to a unique pocket. Since there are a finite amount of obstacles, after a finite amount of these collapsing and splitting steps, each obstacle will be contained in its own pocket (*not necessarily* a pocket of the first GCH R_i):

► **Corollary 8.** *Each obstacle in the face F_i is contained in its own pocket of some GCH.*

Next, we obtain the following upper bound on the total perimeter of the pockets:

► **Lemma 9.** *Let Φ_i be the set of all pockets in F_i such that each pocket contains one obstacle:*

$$\sum_{\rho \in \Phi_i} |\partial \rho| \leq |R_i|.$$

Proof. To prove this, we define a tree structure where each node is either a GCH or a pocket. We call R_i the root GCH, and the pockets resulted from the collapsing of R_i its child pockets. For each pocket, we call the GCH containing all of its obstacles its child GCH. Similarly, each child GCH also has its child pockets. Note that a GCH node could have multiple children (pockets) but a pocket can only have one child. In this tree, only pocket nodes can be leaf nodes. A leaf pocket node is one in which there is exactly one obstacle in it. We use $child(\cdot)$ to denote the set of children of a node. For any GCH node R , we have the inequality

$$\sum_{\rho \in child(R)} |\partial \rho| \leq |\partial R|.$$

This is because each pocket is a different part of R (two pockets share at most one vertex in R), therefore their total perimeter must be less than $|\partial R|$. Next, for any pocket node:

$$|\partial child(\rho)| \leq |\partial \rho|.$$

This is because $child(\rho)$ is a GCH in ρ therefore its perimeter cannot be larger than $|\partial \rho|$. The tree we have now is one such that the perimeter of each node is greater than or equal to the total perimeter of its children, therefore we can conclude that the perimeter of any GCH node is greater than or equal to the total perimeter of all of its descendant leaf pocket nodes. This applies to the root GCH R_i as well, which proves the lemma. ◀

We finally prove the lower bound for the approximation algorithm:

► **Lemma 10.** *We have $|\partial P| \leq 2|\pi^*| \leq 4 \cdot OPT$ and $|MST^*| \leq 2|\pi^*| \leq 4 \cdot OPT$.*

(Recall that MST^* is the minimum spanning tree with Steiner points that spans all holes and the outer boundary of P .)

Proof. Let Φ_i be the set of pockets where each pocket contains exactly one obstacle in F_i . If an obstacle H is a convex hole and ρ is its pocket, then $|\partial H| \leq |\partial \rho|$. However, if H is a reflex chain, then $2|\partial H| \leq |\partial \rho|$: this is because the perimeter of the GCH R of H in ρ is twice longer than ∂H (a GCH of a reflex chain is simply the chain traversed back and forth). We will denote \mathcal{H}_h (resp. \mathcal{H}_r) as the set of convex hole (resp. reflex chain) obstacles in F_i . We acquire the following using Lemma 9:

$$\sum_{H \in \mathcal{H}_h} |\partial H| + 2 \sum_{H \in \mathcal{H}_r} |\partial H| \leq \sum_{\rho \in \Phi_i} |\partial \rho| \leq |\partial R_i| \leq |\partial F_i|. \quad (1)$$

39:12 Sweeping a Domain with Line-Of-Sight Between Covisible Agents

Let $\pi_i = \pi^* \cap \partial F_i$ (the portion of π^* in ∂F_i), then by definition $\partial F_i = \pi_i \cup (\bigcup_{H \in \mathcal{H}_r} \partial H)$ so $|\partial F_i| = |\pi_i| + \sum_{H \in \mathcal{H}_r} |\partial H|$. Using the above inequality we obtain:

$$\sum_{H \in \mathcal{H}_h} |\partial H| + \sum_{H \in \mathcal{H}_r} |\partial H| \leq |\pi_i|.$$

Summing over all F_i , we achieve the first lower bound of the lemma:

$$|\partial P| \leq 2|\pi^*| \leq 4 \cdot OPT.$$

Note that $|\partial P| \leq 2|\pi^*|$ because any portion of π^* is counted twice during the sum: any edge of π^* is shared by at most two different faces.

To prove the second bound in the lemma, we introduce a way to connect all of the *convex holes* to their respective ∂F_i such that the total cost of these connections is bounded by $O(|\pi^*|)$. These connections, along with π^* , will serve as a network connecting all components of the boundary of P , and the total length of this network cannot be shorter than the length of MST^* (the minimum Steiner tree connecting these components), which will give us the second bound of the lemma. We only consider convex holes in this case because any reflex chain obstacle is already connected to π^* by definition.

Recall that when R_i collapses (and creates pockets), there will be at least one collapsing vertex which is supported by a reflex vertex of either ∂P or π^* . Let x be any collapsing vertex that belongs to pocket ρ and H be the convex hole in ρ , and let π_ρ be the (geodesic) shortest path from H to x (see Figure 4 for an illustration). π_ρ is the shortest path between a point $u \in \partial H$ and x . We note that u is the only point shared between π_ρ and ∂H because if there is another point $w \in \pi_\rho \cap \partial H$ then π_ρ is suboptimal (we can simply remove the subpath from u to w to get a better path). Therefore, if we remove H from ρ and compute the shortest path between u and x_ρ it will stay the same. Since π_ρ is the shortest path between two points in a simple polygon (ρ without H), it cannot be longer than a half of $\partial \rho$:

$$2|\pi_\rho| \leq |\partial \rho|. \quad (2)$$

Let Φ_h (resp. Φ_r) be the set of all pockets of the convex holes (resp. reflex chains) in F_i . Note that $\Phi_h \cup \Phi_r = \Phi_i$ and $\Phi_h \cap \Phi_r = \emptyset$. Using (1) and (2):

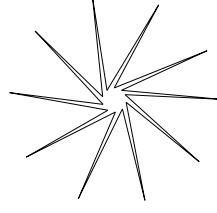
$$2 \sum_{\rho \in \Phi_h} |\pi_\rho| + \sum_{\rho \in \Phi_r} |\partial \rho| \leq \sum_{\rho \in \Phi_h} |\partial \rho| + \sum_{\rho \in \Phi_r} |\partial \rho| \leq |\partial F_i| = |\pi_i| + \sum_{H \in \mathcal{H}_r} |\partial H|.$$

Recall that for each pocket of a reflex chain obstacle, the perimeter of the pocket is at least double that of the reflex chain so $2 \sum_{H \in \mathcal{H}_r} |\partial H| \leq \sum_{\rho \in \Phi_r} |\partial \rho|$, therefore:

$$\begin{aligned} 2 \sum_{\rho \in \Phi_h} |\pi_\rho| + 2 \sum_{H \in \mathcal{H}_r} |\partial H| &\leq 2 \sum_{\rho \in \Phi_h} |\pi_\rho| + \sum_{\rho \in \Phi_r} |\partial \rho| \leq |\pi_i| + \sum_{H \in \mathcal{H}_r} |\partial H| \\ \Rightarrow \sum_{\rho \in \Phi_h} |\pi_\rho| &\leq \frac{1}{2} \left(|\pi_i| - \sum_{H \in \mathcal{H}_r} |\partial H| \right) \leq \frac{1}{2} |\pi_i|. \end{aligned}$$

Let Φ be the set of all pockets containing convex holes of every face, by summing the above inequality over all F_i we obtain $\sum_{\rho \in \Phi} |\pi_\rho| \leq |\pi^*|$. Finally, the length of the network connecting all boundary components, i.e. the union $\pi^* \cup (\bigcup_{\rho \in \Phi} \pi_\rho)$, is $L = \sum_{\rho \in \Phi} |\pi_\rho| + |\pi^*| \leq 2|\pi^*|$. Since L cannot be shorter than MST^* :

$$|MST^*| \leq L \leq 2|\pi^*| \leq 4 \cdot OPT. \quad \blacktriangleleft$$



■ **Figure 7** Pinwheel polygons show that T can be arbitrarily close to $2T'$.

With the above lemma, the perimeter of the degenerate polygon P' is $|\partial P'| \leq (12+\epsilon) \cdot OPT$ for any $\epsilon > 0$ (note that we need to double MST^* to create P'), assuming we are using a PTAS to compute the minimum Steiner tree connecting all boundaries of P . Based on Lemma 4, the schedule computed on P' using the approximation algorithm in Theorem 5 would yield a makespan within $4|\partial P'| \leq (48 + 4\epsilon) \cdot OPT$. Thus, we obtain the theorem:

► **Theorem 11.** *A $(48 + \epsilon)$ -approximation for sweeping a polygonal domain with holes where the agents must return to their depot can be computed in polynomial time.*

5 Breakable Segment Sweeping

We consider the breakable segment sweeping problem in which the agents defining the endpoints of the sweeping segments must stay within P , but we do not require that they remain covisible at all times; however, a point $p \in P$ is swept only if p lies on the segment ab at a moment when the agents a, b are covisible (i.e., $ab \subset P$).

5.1 Comparison with the unbreakable case

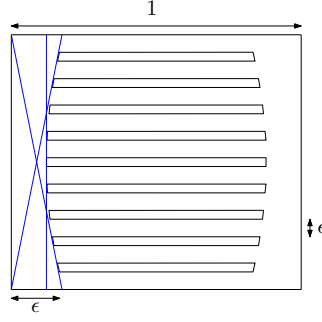
Allowing the sweeping segment to break can allow for a significantly reduced makespan. We first consider the case of a simple polygon P , for which we can bound the ratio of makespans between the breakable and unbreakable segment cases.

► **Lemma 12.** *Given a simple polygon P , let T (resp. T') be the optimal makespan of an unbreakable (resp. breakable) sweeping schedule. Then, $T' \leq T < 2T'$.*

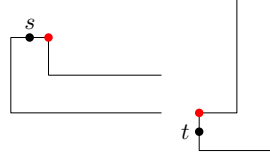
The proof of this lemma can be found in Appendix C.

The upper bound in the lemma is tight: consider pinwheel polygons in which the edges are angled so that no two convex vertices can see each other (Figure 7). To sweep this polygon, an unbreakable segment has to spend about $|\partial P \setminus \pi_r|$ where π_r is the longest reflex chain. This is because after sweeping one pin, one endpoint has to wait for the other before proceeding into the next pin. For a breakable segment, it only takes $0.5|\partial P \setminus \pi_r|$ to sweep all pins because once an endpoint of the segment reaches a convex vertex in a pin, the other endpoint can start moving into an adjacent pin in parallel. This leaves an unswept region in the center with the shape of a regular $\frac{n}{2}$ -gon. The perimeter of this region can be made arbitrarily shorter than the length of the pins so that it only takes a tiny amount of time for the breakable segment to clean it. Overall, the ratio T/T' would approach 2 if we keep adding more pins and stretching them.

Note that Lemma 12 does not apply to polygons with holes since a breakable segment is allowed to pass through them while an unbreakable one is not. Figure 8 shows an example in which T can be worse than T' by a factor of $\Omega(|\mathcal{H}|)$ where \mathcal{H} is the set of holes in the polygon. By Lemma 10, the optimal schedule of an unbreakable segment has makespan



■ **Figure 8** A square of side length 1 containing $|\mathcal{H}|$ horizontal rectangular holes of width at least $1 - \epsilon$. The length ϵ is not drawn to scale. The left (and right) edges of the bars are angled in such a way that it only takes $O(\epsilon)$ for a segment to sweep the portions left/right of the holes, if the segment begins as a unit-length segment along the left/right side of the square.



■ **Figure 9** Two notches (small squares) are attached to the end of the thin hallways in the gadget shown in Figure 2. To sweep the red convex vertices, both agents are forced to travel to the end of each hallway, incurring a cost of about $100ms$ in the makespan.

at least $0.25 \times 2(1 - \epsilon)|\mathcal{H}| = 0.5(1 - \epsilon)|\mathcal{H}|$. However, a breakable segment can sweep this polygon in a time of at most $3.5 + O(\epsilon)$: The agents begin at the midpoint of the left side of P , move apart vertically to extend along that unit-length side, spend time $O(\epsilon)$ to sweep the portion of P to the left of the holes, including the left sides of the holes, then move to the right side (while not sweeping points, as the segment passes through holes), sweep the portion on the right in time $O(\epsilon)$, then reposition in time 1 to lie along the top edge of P , and finally sweep downwards in time 1 to complete the sweep of the corridors between holes.

5.2 Hardness and approximation

As in the case of sweeping with an unbreakable segment, the problem of makespan optimization is NP-hard for sweeping a simple polygon with a breakable segment:

► **Theorem 13.** *It is NP-hard to compute an optimal sweeping schedule of a simple polygon P using a breakable line segment, even if P is an orthogonal simple polygon.*

Proof. We use reduction from 3-PARTITION, similar to the proof of Theorem 1. We modify the gadget by adding a small notch at the end of each narrow corridors (Figure 9), forcing both agents to travel the entire lengths of these corridors in order to sweep the notches. ◀

The algorithm given in Section 3 applies in the breakable segment sweeping case, yielding a 9-approximation when P is a simple polygon, as the same lower bound applies. For the case in which P has holes, we take a different approach to obtain a weaker and more specialized approximation result than we obtained in the unbreakable case.

Consider the case in which P is a polyomino, a union of integer-coordinate unit squares whose planar dual graph, joining pairs of squares (“pixels”) that share a common unit-length edge, is connected. Consider the case in which we are to sweep P with a single

breakable vertical segment. The problem is strongly NP-hard, even if P is a simply connected polyomino – the polygon in Figure 2 from our hardness proof in Section 2 can be turned into a polyomino. Let N be the number of pixels making up P :

► **Theorem 14.** *There exists a polynomial-time algorithm (in N) achieving polylogarithmic approximation of the minimum makespan sweep of a polyomino using a breakable vertical sweeping segment.*

See Appendix C for details of the proof. As pointed out by a reviewer of a previous version of the paper, this result can be extended to polyominoes built from rectangles whose perimeter is bounded by a constant. The above algorithm also applies to sweeping a polyomino with an axis-aligned (horizontal or vertical) breakable segment: in this version, the agents can move arbitrarily but the sweep coverage happens only when the agents are covisible and aligned vertically or horizontally.

6 Conclusion

We introduced a novel motion planning problem where two (or multiple pairs of) agents can sweep a polygonal domain with their line of sight. The problem is strongly NP-hard, and we designed constant-factor approximation algorithms for both simple domains and domains with holes. We intend to improve the constants in these algorithms in our future work, either by a tighter analysis or better methods to decompose the polygon. Many other variants of the problem are to be considered as well, e.g., the starting point can be in the interior of P or no starting point was given. One may also want to limit the maximum distance of the pair, as this is a realistic assumption. In the multiple-segment case, if the agents are allowed to change the pairing, the problem will become much more complex as well as interesting (we thank our reviewer for raising this question).

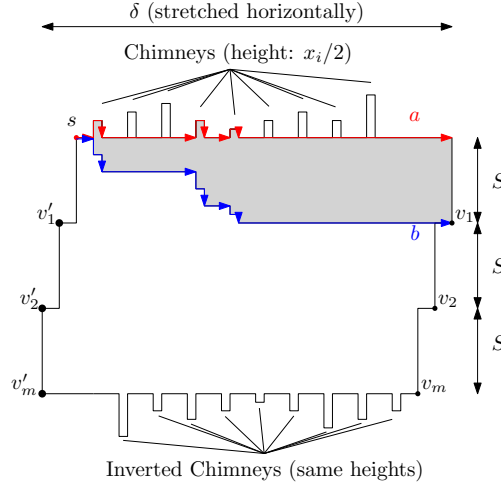
References

- 1 <https://emalm.com/?v=38JGp>.
- 2 <https://emalm.com/?v=8JzrX>.
- 3 Mikkel Abrahamsen, Jacob Holm, Eva Rotenberg, and Christian Wulff-Nilsen. Best laid plans of lions and men. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry, SoCG 2017, July 4–7, 2017, Brisbane, Australia*, volume 77 of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.SoCG.2017.6.
- 4 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995. doi:10.1142/S0218195995000064.
- 5 Esther M Arkin, Michael A Bender, Erik D Demaine, Sándor P Fekete, Joseph S. B. Mitchell, and Saurabh Sethia. Optimal covering tours with turn costs. *SIAM Journal on Computing*, 35(3):531–566, 2005. doi:10.1137/S0097539703434267.
- 6 Esther M Arkin, Rathish Das, Jie Gao, Mayank Goswami, Joseph SB Mitchell, Valentin Polishchuk, and Csaba D Tóth. Cutting polygons into small pieces with chords: Laser-based localization. In *28th Annual European Symposium on Algorithms (ESA 2020)*. Schloss Dagstuhl – Leibniz Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.7.
- 7 Esther M Arkin, Sándor P Fekete, Kamrul Islam, Henk Meijer, Joseph S. B. Mitchell, Yurai Núñez-Rodríguez, Valentin Polishchuk, David Rappaport, and Henry Xiao. Not being (super) thin or solid is hard: A study of grid hamiltonicity. *Computational Geometry*, 42(6-7):582–605, 2009. doi:10.1016/J.COMGEO.2008.11.004.

- 8 Esther M Arkin, Sándor P Fekete, and Joseph S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1-2):25–50, 2000. doi:10.1016/S0925-7721(00)00015-8.
- 9 Binay Bhattacharya, Asish Mukhopadhyay, and Giri Narasimhan. Optimal algorithms for two-guard walkability of simple polygons. In *Workshop on Algorithms and Data Structures*, pages 438–449. Springer, 2001. doi:10.1007/3-540-44634-6_40.
- 10 Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph S. B. Mitchell. Touring a sequence of polygons. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 473–482, 2003. doi:10.1145/780542.780612.
- 11 Adrian Dumitrescu, Ichiro Suzuki, and Paweł Żyliński. Offline variants of the “lion and man” problem – Some problems and techniques for measuring crowdedness and for safe path planning. *Theoretical Computer Science*, 399(3):220–235, 2008. doi:10.1016/j.tcs.2008.02.039.
- 12 Alon Efrat, Leonidas J Guibas, Sarel Har-Peled, David C Lin, Joseph S. B. Mitchell, and TM Murali. Sweeping simple polygons with a chain of guards. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 927–936, 2000.
- 13 Alon Efrat, Joseph S. B. Mitchell, Swaminathan Sankararaman, and Parrish Myers. Efficient algorithms for pursuing moving evaders in terrains. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 33–42, 2012. doi:10.1145/2424321.2424327.
- 14 Alon Efrat, Mikko Nikkilä, and Valentin Polishchuk. Sweeping a terrain by collaborative aerial vehicles. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 4–13, 2013. doi:10.1145/2525314.2525355.
- 15 Naveen Garg, Goran Konjevod, and Ramamoorthi Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000. doi:10.1006/JAGM.2000.1096.
- 16 Leonidas J Guibas, Jean-Claude Latombe, Steven M LaValle, David Lin, and Rajeev Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry & Applications*, 9(04n05):471–493, 1999. doi:10.1142/S0218195999000273.
- 17 Paul J Heffernan. An optimal algorithm for the two-guard problem. In *Proceedings of the Ninth Annual symposium on computational geometry*, pages 348–358, 1993. doi:10.1145/160985.161163.
- 18 Martin Held. Survey of direction-parallel milling. In *On the Computational Geometry of Pocket Machining*, chapter 3, pages 37–52. Springer, 2005.
- 19 Stefan Hertel and Kurt Mehlhorn. Fast triangulation of simple polygons. In *Foundations of Computation Theory: Proceedings of the 1983 International FCT-Conference Borgholm, Sweden, August 21–27, 1983 4*, pages 207–218. Springer, 1983. doi:10.1007/3-540-12689-9_105.
- 20 Kien C. Huynh. Polygon sweeping with line of sight between covisible agents. Software (visited on 2025-08-19). URL: https://github.com/KienHuynh/polygon_sweeping, doi:10.4230/artifacts.24586.
- 21 Christian Icking and Rolf Klein. The two guards problem. *International Journal of Computational Geometry & Applications*, 2(03):257–285, 1992. doi:10.1142/S0218195992000160.
- 22 Rufus Isaacs. *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1999.
- 23 Bo Jiang and Xuehou Tan. Searching for mobile intruders in circular corridors by two 1-searchers. *Discrete applied mathematics*, 159(16):1793–1805, 2011. doi:10.1016/J.DAM.2010.10.007.
- 24 Tsunehiko Kameda, Ichiro Suzuki, and Masafumi Yamashita. An alternative proof for the equivalence of ∞ -searcher and 2-searcher. *Theoretical Computer Science*, 634:108–119, 2016. doi:10.1016/J.TCS.2016.04.016.
- 25 Borislav Karaivanov, Minko Markov, Jack Snoeyink, and Tzvetalin S Vassilev. Decontaminating planar regions by sweeping with barrier curves. In *CCCG*, 2014.

- 26 Steven M LaValle, Borislav H Simov, and Giora Slutzki. An algorithm for searching a polygonal region with a flashlight. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 260–269, 2000. doi:10.1145/336154.336212.
- 27 Kin Sum Liu, Brent Schiller, Jie Gao, Shan Lin, and Joseph SB Mitchell. Optimizing sensor deployment with line-of-sight constraints: Theory and practice. In *EWSN*, pages 95–105, 2019.
- 28 Minko Markov, Vladislav Haralampiev, and Georgi Georgiev. Lower bounds on the directed sweepwidth of planar shapes. *Serdica Journal of Computing*, 9(2):151–166, 2015.
- 29 Joseph S. B. Mitchell. Approximating watchman routes. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 844–855. SIAM, 2013. doi:10.1137/1.9781611973105.60.
- 30 Bengt J Nilsson and Eli Packer. An approximation algorithm for the two-watchman route in a simple polygon. In *European Workshop on Computational Geometry, EuroCG, Lugano, Switzerland (20160330-20160401)*, pages 111–114, 2016.
- 31 J Scott Provan. An approximation scheme for finding steiner trees with obstacles. *SIAM Journal on Computing*, 17(5):920–934, 1988. doi:10.1137/0217057.
- 32 Günter Rote. Pursuit-evasion with imprecise target location. In *SODA*, pages 747–753, 2003. URL: <http://dl.acm.org/citation.cfm?id=644108.644231>.
- 33 Dorian Rudolph. Approximating the sweepwidth of polygons with holes. In *EuroCG*, 2019.
- 34 Marcus Schaefer, Jean Cardinal, and Tillmann Miltzow. The existential theory of the reals as a complexity class: A compendium. *arXiv preprint arXiv:2407.18006*, 2024. doi:10.48550/arXiv.2407.18006.
- 35 Subhash Suri. On some link distance problems in a simple polygon. *IEEE transactions on Robotics and Automation*, 6(1):108–113, 1990. doi:10.1109/70.88124.
- 36 Ichiro Suzuki and Masafumi Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on computing*, 21(5):863–888, 1992. doi:10.1137/0221051.
- 37 Xuehou Tan and Bo Jiang. Minimization of the maximum distance between the two guards patrolling a polygonal region. *Theoretical Computer Science*, 532:73–79, 2014. doi:10.1016/J.TCS.2013.03.019.
- 38 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6 edition, 2023. URL: <https://doc.cgal.org/5.6/Manual/packages.html>.
- 39 Auer Thomas and Held Martin. Heuristics for the generation of random poly-gons. In *Proceedings of the 8th Canadian Conference on Computational Geometry (CCCG'96)*, pages 38–44. Citeseer, 1996. URL: http://www.cccg.ca/proceedings/1996/cccg1996_0007.pdf.
- 40 Christopher Umans and William Lenhart. Hamiltonian cycles in solid grid graphs. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 496–505. IEEE, 1997. doi:10.1109/SFCS.1997.646138.

A Gadget for NP-hardness proof of SweepWithReturn



■ **Figure 10** Modified polygonal gadget for the hardness of SWEEPWITHRETURN. s is at the top left of the polygon.

B Proof of Lemma 2

We restate the lemma here:

► **Lemma 2.** *Every convex vertex of P must be visited by at least one agent.*

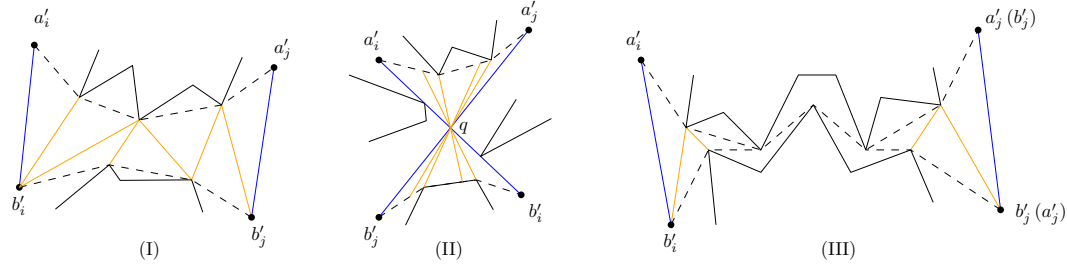
Proof. In order for a convex vertex v to be swept, the point v must at some moment lie on the sweeping segment. If v is not visited by an endpoint of the sweeping segment, then it must be visited at some moment by a point interior to the sweeping segment. But this implies that, at that moment, in the neighborhood of v the sweeping segment, ab , enters the exterior of P , since the two collinear segments av and vb cannot both lie within the convex cone with apex v that is defined by the edges of P incident to v . ◀

C Breakable Segment Sweeping

► **Lemma 12.** *Given a simple polygon P , let T (resp. T') be the optimal makespan of an unbreakable (resp. breakable) sweeping schedule. Then, $T' \leq T < 2T'$.*

Proof. The first inequality ($T' \leq T$) is immediate. We will now provide proof that $T < 2T'$: given any sweeping schedule of a breakable segment, we can convert it into a schedule for an unbreakable segment, while at most doubling the makespan. Recall that the two endpoints of the breakable segment might, at some times, not see each other; however, points of P are only being swept during times when the endpoints are covisible. Therefore, a schedule of a breakable segment can be seen as a sequence of periods (intervals of time) during each of which the endpoints are either always covisible or always hidden from each other. During a covisible period, an unbreakable segment can simply follow the sweeping schedule of the breakable segment (since it is not broken during this time interval), with no change in makespan. For any hidden period, let $a'_i b'_i$ be the first position of the breakable segment and $a'_j b'_j$ be the last. If T'_{ij} is the makespan of the breakable segment in this period, then

$$T'_{ij} \geq \max\{|\pi(a'_i, a'_j)|, |\pi(b'_i, b'_j)|\}, \quad (3)$$



■ **Figure 11** Three cases that can happen when $a'b'$ moves from $a_i'b'_i$ to $a'_jb'_j$. The dashed lines denote the geodesic shortest paths between the pairs of points, the black polygonal lines represent the boundary of P and the orange lines show how we decompose each case into sweepable components for the unbreakable segment.

where $\pi(a'_i, a'_j)$ and $\pi(b'_i, b'_j)$ are the geodesic shortest paths from a'_i to a'_j and b'_i to b'_j respectively.

There are now three cases (Figure 11):

- (I) $a_i'b'_i$ does not intersect $a'_jb'_j$ and neither do $\pi(a'_i, a'_j)$ and $\pi(b'_i, b'_j)$. The union $a_i'b'_i \cup a'_jb'_j \cup \pi(a'_i, a'_j) \cup \pi(b'_i, b'_j)$ will bound a region P_1 , which is an hourglass polygon (i.e. a polygon consisting of two reflex chains that are connected by two edges $a_i'b'_i$ and $a'_jb'_j$). Now, consider the triangulation of the hourglass polygon. Let the triangle containing $a_i'b'_i$ be Δ_i and the triangle containing $a'_jb'_j$ be Δ_j . If we consider each triangle as a node of a graph and create an edge between two nodes if their corresponding triangles share an edge in the triangulation, then this graph (the dual of the triangulation) is a path from node Δ_i to node Δ_j . This is always true for an hourglass polygon. Note that other than Δ_i and Δ_j , a triangle in the triangulation consists of exactly two chords of P_1 and one edge on its boundary (an edge of $\pi(a'_i, a'_j)$ or $\pi(b'_i, b'_j)$). To let the unbreakable segment sweep this hourglass, we use the following motion plan:
 - If ab lines up with an edge of some triangle Δ_k , this edge must be $a_i'b'_i$, $a'_jb'_j$ or a chord of P_1 , but not an edge of $\pi(a'_i, a'_j)$ or $\pi(b'_i, b'_j)$.
 - Assume ab is lining up with one edge of an unswept triangle Δ_k , either a or b must be incident to an edge of $\pi(a'_i, a'_j)$ or $\pi(b'_i, b'_j)$. Whoever is incident to such an edge will move along that edge while the other stays stationary during the whole period. This will sweep Δ_k and let ab line up with an edge of the next triangle.
 - Repeat the above step until ab lines up with $a'_jb'_j$.
 This whole process will cost no more than $|\pi(a'_i, a'_j)| + |\pi(b'_i, b'_j)|$.
- (II) $a_i'b'_i$ intersects with $a'_jb'_j$. Let $q = a_i'b'_i \cap a'_jb'_j$. $\pi(a'_i, a'_j)$ and $\pi(b'_i, b'_j)$ will be two reflex chains visible from q . We can view $a_i'b'_i \cup \pi(a'_i, a'_j) \cup a'_jb'_j \cup \pi(b'_i, b'_j)$ as a self-intersecting polygonal chain, or the boundary of a simple polygon with a degenerate vertex q . To move from $a_i'b'_i$ to $a'_jb'_j$ using an unbreakable segment, we can rotate the segment around q while a moves along $\pi(a'_i, a'_j)$ and b moves along $\pi(b'_i, b'_j)$ in opposite directions. This can be done as follows: we project all vertices from each reflex chain onto the other chain using q as a pinhole. The projections decompose the degenerate polygon into smaller double wedges, which can be swept one by one. Each such wedge has two bases, one of which is at least as long as the other. Sweeping a double wedge will require a time equal to the length of the longer base.
- (III) $\pi(a'_i, a'_j)$ and $\pi(b'_i, b'_j)$ share some vertices. In this case, the two of them will create two funnels whose bottom vertices are attached to the two ends of the shared portion. Similarly to the hourglass case in (I), we can triangulate the funnels and:

- Let the segment sweep the first funnel polygon by moving from one triangle to another. Both a and b will eventually meet at the bottom vertex of the first funnel polygon.
 - Since a and b are now colocated, they can both move along the shared portion of $\pi(a'_i, a'_j)$ and $\pi(b'_i, b'_j)$ until both of them arrive at the bottom endpoint of the second funnel polygon.
 - Finally, the segment ab can sweep the second funnel polygon by moving from one triangle to another until they line up with $a'_j b'_j$.
- This process also costs at most $|\pi(a'_i, a'_j)| + |\pi(b'_i, b'_j)|$.

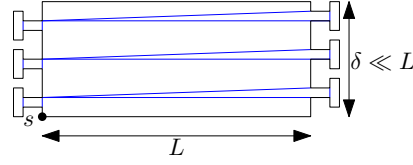
In all cases, the produced schedule for the unbreakable segment is at most $2T'_{ij}$ due to (3). ◀

► **Theorem 14.** *There exists a polynomial-time algorithm (in N) achieving polylogarithmic approximation of the minimum makespan sweep of a polyomino using a breakable vertical sweeping segment.*

Proof. At any moment in time, the sweeping segment ab has a *combinatorial type*, (p, q) , indicating the pixel p currently containing a and the pixel q currently containing b (possibly $p = q$). At time 0, the combinatorial type is (p_s, p_s) , where p_s is the pixel on whose boundary the starting point s lies. For each pixel $\gamma \in P$, in order for points in γ to be swept by ab , at least one of the set \mathcal{C}_γ of combinatorial types must be visited, where \mathcal{C}_γ is the set of combinatorial types (p, q) for which pq is a vertical column of pixels within P that contains γ (i.e., $\gamma \in pq$). Over the time window $[0, T]$ of an optimal sweep, the segment ab has a sequence, possibly with repeats, of combinatorial types, $(p_1, q_1), (p_2, q_2), \dots, (p_k, q_k)$; this sequence must contain at least one combinatorial type from \mathcal{C}_γ , for every $\gamma \in P$. (Note that some of the types (p_i, q_i) may specify vertical columns of pixels that do not all lie within P ; we do allow ab to lie partially outside of P in the breakable segment sweeping problem.) By continuity of motion of the agents a and b , for each i , the L_1 distance between pixel p_i and p_{i+1} , and between q_i and q_{i+1} , is at most 1 (it is 0 or 1). We define the distance between (p_i, q_i) and (p_{i+1}, q_{i+1}) to be $\max\{|p_i p_{i+1}|_1, |q_i q_{i+1}|_1\}$. Then, in this metric, we consider the one-of-a-set (generalized) TSP path, to visit every set \mathcal{C}_γ , for $\gamma \in P$. The optimal makespan T must be at least the length of such a TSP path. Further, for any TSP path in the “combinatorial type space” that visits all of the sets \mathcal{C}_γ , we can, with only an $O(1)$ factor greater length, execute a sweep with a vertical segment ab , with $a \in p_i$ and $b \in q_i$, shifting the vertical segment (of fixed length) around the boundaries of pixels p_i and q_i , so that all points in the vertical column $p_i q_i$ of pixels within P are swept. Since the one-of-a-set TSP has a polylog factor approximation algorithm [15], applying it here will result in Theorem 14. ◀

D Experiments

Sweeping a simple polygon with 1 segment can be done with the following heuristic which, similar to our approximation algorithm (Section 3), has two phases: first, decompose P into convex pieces ($O(n)$ time [19]) – the dual graph of the decomposition is a tree; then, sweep the pieces one-by-one in DFS order around the tree, starting with the component that contains s (Figure 12 shows that the heuristic’s makespan can be $\Theta(n \cdot OPT)$). We implemented one-segment sweeping with return both with our algorithm (Section 3) and the above heuristic; Python was used for the main procedures, while C++ was used as a proxy for calling CGAL modules [38]. We generated 3 sets of random simple polygons (the vertices are either within the unit square or unit disk), using the algorithm of [39] (for each



■ **Figure 12** P is a large central room, with small rooms (nooks) attached on the left/right sides; sweeping the large central room first and then sweeping the small rooms on the left and then those on the right takes $O(L + n\delta) = O(L)$ time if $\delta \ll L$. The convex decomposition (blue segments) splits the middle room into convex pieces: sweeping every node in the dual tree (i.e., using the heuristic) forces the agents to move from left to right and vice versa $\Theta(n)$ times, taking $\Theta(nL)$ time.

polygon, s is a random vertex): 1000 polygons with 20 to 100 vertices each, 500 polygons with 500 to 1000 each, and 50 polygons with 5000 to 10000 vertices each. The results of experimenting with the polygons are shown in Fig. 13: it can be seen that although the heuristic algorithm (HA) has no approximation guarantee, it outperforms the (provable) approximation algorithm (AA) in most instances (the heuristic takes better advantage of parallelism during the second step, by moving both agents at maximum speed when they are able to do so; in the approximation algorithm, the agents move in parallel only when sweeping histogram subpolygons $HP(e)$, and even then the bottom agent slows down in order to accommodate the top agent).

Table 1 shows the average and maximum ratios of the makespans of AA and HA; the average speedup of HA is about 1.333 and the maximum speedup observed is 1.743. The average ratio of the AA makespans and the theoretical lower bounds is between 3.903 and 4.032. In the worst instance of the experiments, the schedule computed by AA takes 5.116 more time than the proven lower bound; this observed factor is still significantly less than the factor of 8 proved in Theorem 5; we believe that the reason is that the theoretical lower bound is usually unachievable (except for some specific classes of polygons like convex).

■ **Table 1** Average and maximum makespan ratios between AA, HA, and the lower bound (LB).

Makespan ratio		AA/LB	HA/LB	AA/HA
Mean	$n \in [20, 100]$	3.903	2.933	1.337
	$n \in [500, 1000]$	4.007	3.022	1.327
	$n \in [5000, 10000]$	4.032	3.035	1.328
	All n	3.941	2.965	1.333
Max	All n	5.116	3.928	1.743

Figure 14 shows typical sweeping schedules of our algorithms; videos of our sweeps can be viewed at [1, 2]. Teal regions are swept earlier, red regions later. In some cases, red regions are isolated from those of similar color, as in the bottom left region of the top left figure, where the segment backtracks to sweep a region near s . Ideally, regions of similar color should be close to each other; however, achieving this requires a better ordering to explore the children of each node (better than the DFS).

39:22 Sweeping a Domain with Line-Of-Sight Between Covisible Agents

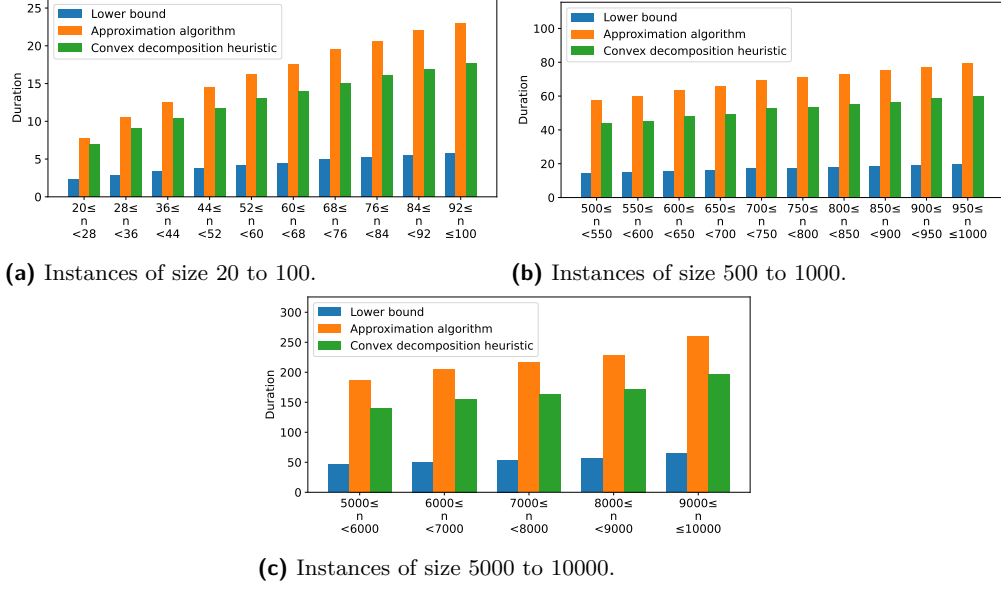


Figure 13 The makespans computed using both algorithms in Section 3 for all three sets of randomized polygons. The makespan lower bound for each instance is $\frac{1}{2}|\partial P|$ as mentioned in Subsection 3.1. We grouped the results into bins by the number of vertices of the polygons (n) and calculate the average value for each algorithm as well as the lower bound per group.

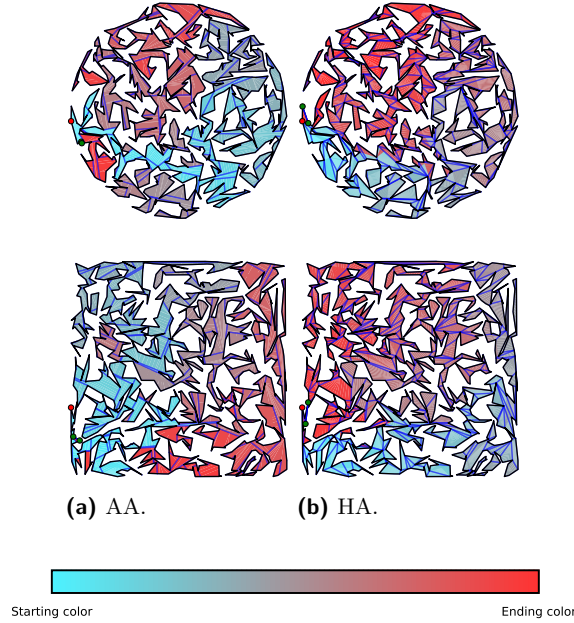


Figure 14 Visualization of the sweeping schedules computed by the two algorithms. The top polygon has 774 vertices and the bottom one has 895 vertices. The red dot indicates the starting location of the agents and the green dots indicate where the agents are at the end of the schedule before returning to red. Blue line segments within each polygon show how the polygon was decomposed into subpolygons for the corresponding algorithm. To show the positions of agents over time, we employ a transitioning color scheme in which the segment connecting them is teal at the beginning and red at the end.