

Optimal Allocation of Many Robot Guards for Sweep-Line Coverage

Si Wei Feng

Teng Guo

Jingjin Yu

Abstract—We study the problem of allocating many mobile robots for the execution of a pre-defined sweep schedule in a known two-dimensional environment, with applications toward search and rescue, coverage, surveillance, monitoring, pursuit-evasion, and so on. The mobile robots (or agents) are assumed to have one-dimensional sensing capability with probabilistic guarantees that deteriorate as the sensing distance increases. In solving such tasks, a time-parameterized distribution of robots along the sweep frontier must be computed, to minimize the number of robots used to achieve some desired coverage quality guarantee or to maximize the probabilistic guarantee for a given the number of robots. We propose a max-flow-based algorithm for solving the allocation task, which builds on a decomposition technique of the workspace as a generalization of the well-known boustrophedon decomposition. Our proposed algorithm has a very low polynomial running time and completes in under two seconds for polygonal environments with over 10^5 vertices. Simulation experiments are carried out on three realistic use cases with randomly generated obstacles of varying shapes, sizes, and spatial distributions, demonstrating our proposed method’s applicability and scalability.

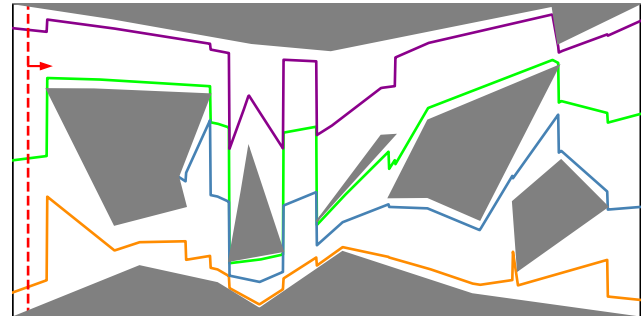
Introduction video: <https://youtu.be/8taX92rzC5k>.

I. INTRODUCTION

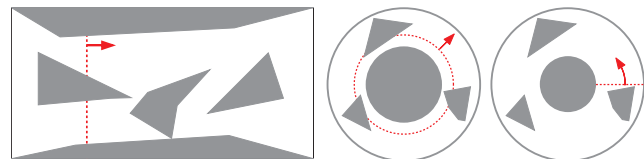
Searching for a static or moving target in a planar environment is a classic problem in robotics [1]–[5]. The setting applies to many real-world applications, including searching for lost person/object, checking for potential hazards, and generally, search and rescue tasks conducted in a known environment. Research tackling this problem mostly focuses on devising a search plan with different types of objectives, such as minimizing the total length of the frontier of the search schedule from the start to the end [6], minimizing the number of robots used for the search plan in a visibility-based robot sensing model [7], and so on.

In certain cases, the high-level plan for searching a given region may already be pre-determined and fixed. For example, in search and rescue efforts, a frequently carried out plan is to perform a single sweep of an environment with a marching frontier, which is easy to execute when many participating robots/agents are involved. The high-level search plan may also be determined by existing algorithms that compute only the search frontier. However, even in the case where the search plan consists of pre-determined sweep (frontier) lines; it remains non-trivial to find an optimal organization of the mobile robots to execute the search plan, for either minimizing the number of robots needed

¹S. W. Feng, T. Guo and J. Yu are with the Department of Computer Science, Rutgers, the State University of New Jersey, New Brunswick, NJ 08901, USA. E-mails: {siwei.feng, teng.guo, jingjin.yu}@rutgers.edu This work is supported in part by NSF award IIS-1845888 and an Amazon Research Award.



(a) Optimal robot allocations in a vertical sweep



(b) Vertical

(c) Circular

(d) Radial

Fig. 1: (a) An illustration of robots’ locations along a left-to-right vertical sweep schedule. Four robots are allocated to execute the vertical sweep, and their trajectories are illustrated in different colors. (b)(c)(d) Illustrations of three use cases: vertical, circular, and radial sweeps.

for a given sensing probability requirement or utilizing a fixed number of robots to maximize the minimum sensing probability of locating something in the environment.

In this work, we address the challenge of how to best allocate many robots to execute a pre-determined search schedule for a known environment. More specifically, for a two-dimensional closed and bounded workspace, and a known search schedule, which gives a search frontier for any given time, the robot guards are required to stay on the search frontier to carry out the sensing task. Because each robot’s coverage quality deteriorates with distance, their relative placement on the frontier must be carefully decided to maximize the coverage quality. For the setup, we focus on the problem of finding the minimum number of robots required to execute a pre-determined sweep schedule such that a minimum sensing quality is guaranteed for each point in the workspace. Solutions to this minimization problem readily translate to solutions for maximizing the coverage quality for a fixed number of robots, which is a dual problem.

In summary, the main contributions of this work are twofold. First, we generalize the notion of boustrophedon decomposition [8], which partitions the plane via vertical

sweep, into a decomposition of the plane with a *continuous monotone sweep schedule*, which we prove can always be represented as a directed acyclic graph (DAG). Then, we show the problem of minimizing the number of line guards required to execute a sweep schedule can be transformed into a network flow problem. Since the generalized boustrophedon decomposition and the network flow problem can be solved in low polynomial time; our method achieves high levels of scalability. The strengths of our method are further corroborated in extensive simulation experiments on three realistic use cases: *vertical sweep*, *circular sweep*, and *radial sweep* (see, e.g., Fig. 1 (b)(c)(d)).

Related Work. The study in this paper draws inspiration from the study of several lines of related problems. The Graph-Clear problem, formulated in [5], tasks a group of robots to search and clear an environment with the operations of blocking and clearing. A follow-up work on Line-Clear [6] uses line guards with more focus on computational geometry in that the objective is to minimize the maximum sweep line distance. Both of these problems are NP-hard, establishing the difficulties of finding a sweep schedule for a planar environment. The more general pursuit-evasion problem dates back to the research on *search number* on a discrete graph [7], followed by studies on pursuit and evasion continuous environment with visibility-based model [1]–[4]. The problem becomes the well-known art-gallery problem [9] when a static deployment of robots is sought after. When working with known patrolling search frontiers, e.g., vertical sweep lines, this problem is analogous to the perimeter defense problem by placing guards on a static perimeter to defend intruders [10]–[12]. Previously, we have also studied a version of static range guard placement problems for securing perimeters and regions [13]. In contrast to the pursuit-evasion algorithms that deal with searching dynamic and unpredictable targets that could escape, coverage planning/control-related algorithms become more suitable for searching or covering predictable or stationary targets, e.g., room sweeping, pesticide and fertilizer spraying, persistent monitoring, and so on [14]–[22].

II. PRELIMINARIES

In this section, we first describe a general probabilistic robot sensing model used in this paper and introduce the notion of a *continuous monotone sweep schedule*, along which robots must be deployed to carry out the search task. Then, the problem of finding the minimum number of robots for a certain sweep schedule is formally defined.

A. Robot Sensing Model

In this work, a robot is assumed to be a *line guard* capable of sensing relevant events happening on a continuous line segment passing through the robot. On the line segment guarded by a robot, for a (point) target at a distance of r to the robot, the robot has probability $\rho(r)$ of detecting or capturing the target, where $\rho : \mathbb{R}^+ \rightarrow [0, 1]$ is a decreasing sensing probability function.

Individual robots' 1D sensing range aligns to form a *sweep line* or *sweep frontier*. It is assumed that robots carry out their sensing functions independently without interfering with each other. For a point p in the sweep line, if it falls between two robots r_1, r_2 , the coverage of that point is provided by r_1 and r_2 , and the probability of target detection at that point is computed as $1 - (1 - \rho(\ell_1)) \cdot (1 - \rho(\ell_2)) = \rho(\ell_1) + \rho(\ell_2) - \rho(\ell_1) \cdot \rho(\ell_2)$, where ℓ_1 (resp., ℓ_2) is the distance between p and r_1 (resp., r_2). If it lies at the ends of the segment, the coverage of that point is only provided by the closest robot, and the coverage probability is simply $\rho(\ell)$, where ℓ is the distance between p and the robot. These are illustrated in Fig. 2. We note that *sweep lines* are not necessarily straight. For example, a sweep line could be part of a circle in the case of circular sweeps (Fig. 1(c)).

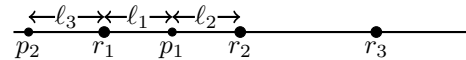


Fig. 2: Illustration of the robot sensing model. The segment is being covered by three robot guards $r_1 \sim r_3$. The coverage probability of p_1 , falling between robots r_1 and r_2 , is given as $\rho(\ell_1) + \rho(\ell_2) - \rho(\ell_1) \cdot \rho(\ell_2)$, and the coverage probability of p_2 , covered only by r_1 is given as $\rho(\ell_3)$.

For covering a line segment with a length of ℓ and sensing function of ρ , we denote as $\zeta : \mathbb{R}^+ \rightarrow \mathbb{N}^+$ a primitive for computing the minimum number of robots needed to achieve the required minimum covering probability of ρ_0 .

Take the exponential decaying sensing probability function as an example, where $\rho(r) = e^{-c \cdot r}$, $c > 0$ is some constant. In this case, the maximum distance at the two ends to guarantee the sensing probability of ρ_0 is $d_1 = -\ln(\rho_0)/c$. If a point is between two robots with distance ℓ_1 and ℓ_2 to the two robots, its sensing probability is bounded by

$$\rho(\ell_1) + \rho(\ell_2) - \rho(\ell_1) \cdot \rho(\ell_2) \geq 2e^{-\frac{c \cdot d}{2}} - e^{-c \cdot d} \\ \text{where } d = \ell_1 + \ell_2$$

So, the required maximum distance between two neighboring robots is $d_2 = -2 \ln(1 - \sqrt{1 - \rho_0})/c$. Therefore, the minimum number of robots required to cover a line segment with length ℓ is $\zeta(\ell) = \max(1, \lceil (\ell - 2 \cdot d_1)/d_2 \rceil)$.

B. Problem Formulation

In this paper, we work with a 2D compact (i.e., closed and bounded) workspace $\mathcal{W} \subset \mathbb{R}^2$, which can contain a set of obstacles. An existing *sweep schedule* is an input to the problem, which we define the *continuous monotone sweep schedule* for \mathcal{W} as

Definition 1 (Continuous Monotone Sweep Schedule). *A function $P(t)$ that maps a positive timestamp t to a continuous curve is a continuous monotone sweep schedule for \mathcal{W} if $P(t)$ changes continuously, and for each point $p \in \mathcal{W}$, there exists a single t' such that $p \in P(t')$.*

Intuitively, a continuous monotone sweep schedule defines a function that maps the time step to a 1-D curve in the workspace that sweeps through each point in \mathcal{W} only once. Depending on the continuous curve, $P(t)$ could take various forms. For example, the sweep line may be straight in a search-and-rescue scenario. Or the sweep line may be circular in a search-and-capture scenario. In the rest of this paper, we simply refer to a continuous monotone sweep schedule as a *sweep schedule*. Next, we introduce the notions of *arrival time* and *monotone chain*.

Definition 2 (Arrival Time). *For a search schedule $P(t)$, and a point in the workspace $o \in \mathcal{W}$, the arrival time at o , $arrival(o)$, is defined as the time step t when $o \in P(t)$.*

Definition 3 (Monotone Chain). *For a bounded 2D chain: $\tau(s) : [0, 1] \rightarrow \mathcal{W}$, where τ is a continuous function, it is considered as a monotone chain to the search schedule $P(t)$ if and only if*

$$s_1 < s_2 \Leftrightarrow arrival(\tau(s_1)) < arrival(\tau(s_2)).$$

In a search schedule or plan, $P(t)$ may be intersected by obstacles. In this case, $P(t)$ is separated into multiple continuous segments; each of these segments requires a dedicated group of robots. That is, robots on one continuous segment cannot provide coverage of other segments.

With the probabilistic robot sensing model, we formulate the robot team scheduling problem for a given sweep schedule as the following,

Problem 1. *Given a sweep schedule $P(t)$ for a 2D region \mathcal{W} , and a group of robots with coverage capability function ρ , what is the minimum number of robots required to execute the sweep schedule on the sweep line, such that for every point o in \mathcal{W} , the probability that o is covered is at least some fixed $0 < \rho_0 \leq 1$?*

III. OPTIMAL ROBOT ALLOCATION

Our proposed algorithm can be divided into two steps at the high level. In the first step, the algorithm conducts a *generalized boustrophedon decomposition* for the given environment along the sweep line, which generates a *directed acyclic graph* (DAG) representation of the workspace \mathcal{W} for a given sweep schedule. Using the DAG, a max-flow-based algorithm is then applied to compute the minimum number of robots required for executing the sweep schedule, as well as the corresponding arrangement of robots.

A. Generalized Boustrophedon Decomposition

In solving search and coverage problems, various decomposition techniques have been proposed, including trapezoidal, Voronoi, boustrophedon, Morse decompositions, and so on [8], [23]–[25]. For our robot allocation task, it is also natural to start with a decomposition of the environment. However, we need a decomposition supporting non-straight boundaries between the decomposed cells, created by the

sweep schedule. For this, we propose a generalization of boustrophedon decomposition.

Before describing the generalization of boustrophedon decomposition, we briefly introduce boustrophedon decomposition (readers are referred to [8] for further details), which in turn is based on trapezoidal decomposition. The difference is that it removes the sweeping events that cross inner vertices. As illustrated in Fig. 3, compared with trapezoidal decomposition, boustrophedon decomposition has fewer cells, which leads to fewer (back-and-forth) boustrophedon motions.

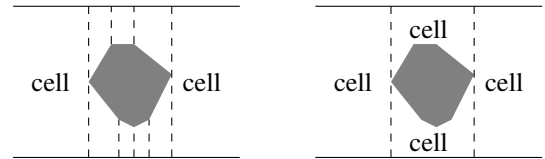


Fig. 3: [left] A trapezoidal decomposition, creating a total of 9 cells. [right] Boustrophedon decomposition of the same environment, creating only 4 cells.

We extend the boustrophedon decomposition from using only vertical sweep lines to allow the use of any *continuous monotone sweep schedule*. We are given a sweep schedule $P(t)$ for a workspace \mathcal{W} , which could take any curved form (see Fig. 4). By following the sweep schedule, it is possible to construct a decomposition of \mathcal{W} , based on the events of cell splitting and merging.

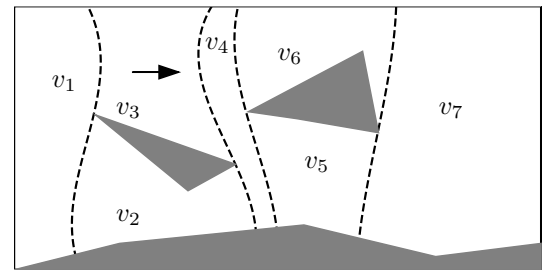


Fig. 4: Suppose we have a sweep schedule that sweeps the environment from left to right, with curved sweep fronts. The curves, as they cross critical vertices of objects, are shown as the dashed lines. The generalized boustrophedon decomposition decomposes \mathcal{W} into 7 cells, $v_1 \sim v_7$. It is important to note here that, for any continuous monotone sweep schedule, a decomposition can be obtained.

Theorem 1. *A sweep schedule can be organized in a DAG by the generalized boustrophedon decomposition.*

Proof. Following the sweep schedule, we can conduct a generalized boustrophedon decomposition of the workspace \mathcal{W} . A node in the DAG will represent a cell after the decomposition, whose parents and children are the predecessor and successor cells along the sweep schedule.

Additionally, we add a source node that links to all nodes without a parent and add a terminal node that links from all nodes without a child. Since obstacles in the environment

can contain concave vertices, we must consider two special cases involving concave vertices for the construction of DAG, as illustrated in Fig. 5. When a line segment of the sweep schedule ends at a concave vertex, the corresponding node in the DAG is linked to the terminal node t . When a line segment of the sweep schedule starts at a concave vertex, the corresponding node in the DAG is linked from the source node s . In mapping these scenarios to detailed plans, it means that some robots will start later or end earlier compared with others.

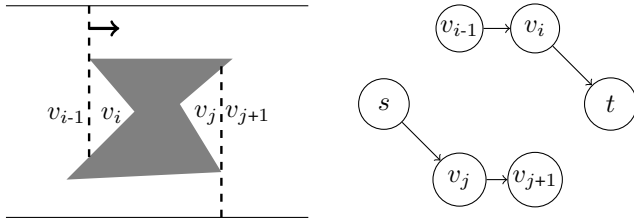


Fig. 5: An example that contains two concave scenarios. As a result of the DAG construction, some robots will start their work at v_j (by having s as its parent) and some robots will end their work at v_i (by having t as its child).

□

The implementation of the generalized boustrophedon decomposition is similar to vertical decomposition, we provide here an implementation Alg. 1 adapted from [26] under the *generation position* assumption (i.e., there are no degenerative settings). Basically, the algorithm works by maintaining a binary search tree for *boundary chains* representing cell boundaries. The algorithm considers two types of events during the sweeping process: splitting a cell into two cells and merging two cells into one cell. Since the time cost mainly comes from maintaining the binary search tree of chains, with an efficient implementation, Alg. 1 requires $O(n \log n)$ time, where n is the complexity of the environment.

B. Reduction to Circulation with Demand

Since we are working with *monotone sweep schedules*, for any point p in \mathcal{W} , it can only be contained in $P(t')$ for a single t' , i.e., each point $p \in \mathcal{W}$ is only swept once. Denote the length of the line segment where it is contained as L , which requires at least $\zeta(L)$ robots inside that segment at time t' (recall that ζ is the primitive defined in Sec. II-A). For each decomposed cell, the minimum number of robots required is $\zeta(\ell_{max})$, where ℓ_{max} is the maximum length of the sweep line inside that cell. Given a DAG $G(V, E)$ that represents the sweep schedule, the arrangement problem of the robots along the sweep line can be transformed into a network flow problem on the DAG. For each node $v \in G$, there is a requirement of coverage for the node, $demand(v)$, which can be computed as $\zeta(v.\ell_{max})$. Given the DAG and the demands, we are then to “flow” the robots through the schedule, allocating a certain number of robots to each decomposed cell along the way to satisfy these demands.

Specifically, our problem may be further cast as a *circulation with demand* problem [27], with the following augmentation. We replace each node v in G with two vertices v_1 and v_2 and replace every edge uv in the previous graph with edge u_2v_1 , as illustrated in Fig. 7. The edge between v_1 and v_2 has flow demand of $demand(v)$. The following minimum circulation with demand problem is then obtained.

Algorithm 1 GenBouDecomp(P, \mathcal{W}): Generalized Boustrophedon Decomposition

Data: $P(t)$: a sweep schedule. \mathcal{W} : the workspace.

Result: a DAG from the decomposition.

1. Separate the boundaries of obstacles into a set of *monotone chains*, where each chain has the points on it *arrival time* arranged in an increasing order;
 2. $T \leftarrow$ A binary search tree of the chains
 3. Construct an *event array* that contains the events of the start of a chain and the end of a chain, sorted by their occurrence time during the sweep.
 4. Iterate over the event array, which inserts and deletes chains from the binary search tree T , while making sure that T represents the current order of the chains.
 - 4.1. When two chains are added to T , a cell is split into two new cells in the case of a convex vertex, or a new cell is created for a concave vertex, where an edge directed from s is added.
 - 4.2. When two chains are removed from T , two cells are merged into a new cell in the case of a convex vertex or a cell disappears for a concave vertex, where an edge is added directed to t .
 5. Return the DAG constructed based on $P(t)$.
-

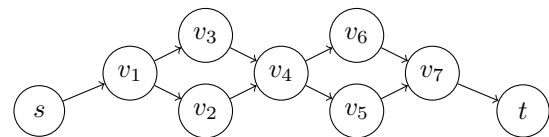


Fig. 6: The constructed DAG from the example in Fig. 4

Problem 2 (Minimum Circulation with Demand). *There is a graph $G(V, E)$, where s, t are the source and terminal nodes. Every edge uv in G has a demand of $d(uv)$ and a capacity of $c(uv)$. Compute minimal network flow from s to t that saturates all the edge demands.*

Readers are referred to [27] for the details of the classical “circulation with demand” problem solution with max-flow.

Alg. 2 outlines the operations used in this section. We note that the graph data structure needs to have the functionality of adding vertex and adding edge with edge demand, and $G'.flow[v_1][v_2]$ denotes the flow from v_1 to v_2 on G' .

C. The Complete Allocation Algorithm

The overall algorithm for robot allocation is described as in Alg. 3. To analyze the running time of Alg. 3 for a polyg-

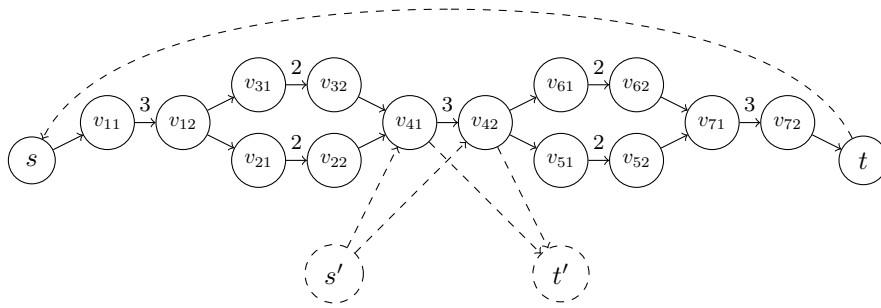


Fig. 7: Transform the DAG in Fig. 6 into a circulation with demand problem. The value above each edge represents the demand of the edge, which is eliminated when it is zero. The source node is s , and the target node is t . s' and t' are the auxiliary source and target for solving the “circulation with demand” problem. Also, auxiliary edges are added between s' , t' and every other vertex.

Algorithm 2 MinSweepDAG (dag, ζ)

Data: $dag(V, E)$: the DAG obtained from GenBouDecomp.

ζ : the sensing requirement primitive function.

Result: dag : the updated dag with the robot allocation information

$G' \leftarrow$ a new empty graph;

for $v \in dag$ **do**

$G'.add_vertex(v_1), G'.add_vertex(v_2)$;

$G'.add_edge(v_1, v_2, capa=\infty, demand=\zeta(v.\ell_{max}))$;

for $u \in dag.neighbor[v]$ **do**

$G'.add_edge(v_2, u, capa=\infty, demand=0)$;

end

end

MinCirculationWithDemand (G');

for $v \in dag$ **do**

$dag.v.guards_num \leftarrow G'.flow[v_1][v_2]$;

for $u \in dag.neighbor[v]$ **do**

$dag.flow[v][u] = G'.flow[v_2][u_1]$;

end

end

return dag

onal environment, we denote the environment complexity (number of vertices) as n . As mentioned in the previous section, the generalized boustrophedon decomposition takes $O(n \log n)$ time. As the generation of a node in the DAG comes from some chain insertion or deletion events, and an event would require one vertex to happen, there is at most $O(n)$ decomposed cells, i.e., nodes in the DAG. Similarly, adding edges between nodes comes from some chain insertion or deletion events, and the number of events is at most $O(n)$. So, both the number of edges and the number of nodes in the DAG is $O(n)$. Moreover, it is easy to see that the DAG is a planar graph since a node corresponds to a decomposed cell, and there is an edge between nodes only if the two corresponding cells are adjacent. The circulation with demand problem requires solving two max-flow problems based on the DAG. If we use the push-relabel algorithm [28] that runs in $O(V^2\sqrt{E})$, solving the max flow for this

problem will cost $O(n^{2.5})$, since $|V|, |E| = O(n)$.

Algorithm 3 MinSweep (P, \mathcal{W}, ρ_0): Computing Minimum Number of Robots for a Sweep Schedule

Data: $P(t)$: a sweep schedule. \mathcal{W} : a compact workspace.

ρ_0 : required sensing probability guarantee.

Result: n : the minimum number of robot guards required.

$plan$: the corresponding allocation plan

Construct ζ based the sensing model and ρ_0

%Primitive function ζ is to compute the minimum number of robots required for a continuous line segment

$dag \leftarrow$ GenBouDecomp(P, \mathcal{W});

$dag \leftarrow$ MinSweepDAG(dag, ζ);

$plan \leftarrow dag$;

%The plan of the robots can be constructed based on the flow on the dag .

return $dag.s.guards_num, plan$;

Remark 1. In the case of having a fixed number of robots, and the objective is to maximize the minimum coverage probability of a point in \mathcal{W} . We can apply binary search on the minimum coverage probability we can guarantee, where Alg. 3 can be used to decide whether some coverage probability can be guaranteed by the fixed number of robots.

IV. SIMULATION EVALUATION

In this section, we perform a numerical evaluation of our proposed method with two goals: (1) confirms the scalability and (2) observe the behavior of the method as input parameters change. We implemented our proposed algorithms in C++. Dinic’s algorithm is used to solve the max-flow problem [29] for simplicity. The methods are evaluated at an Intel® Core™ i5-10600K CPU at 4.1HZ. For the three use cases (Fig. 1 (b)(c)(d)), we programmatically create a large number of test cases, with up to 6,000 randomly generated polygonal obstacles. The number of vertices of each polygon ranges from 3 to 50, making the total vertices

up to 100,000+. Fig. 8 shows the largest problem instances for the experiments with around 100,000 vertices.

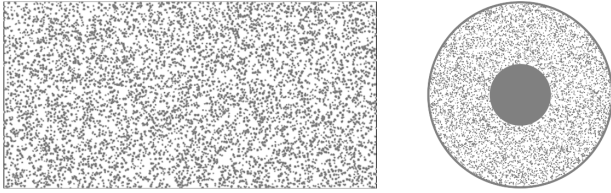


Fig. 8: Examples of programmatically generated test environments with total vertices around 100,000.

Algorithm performance. In a first set of evaluations, under an exponentially decaying sensing model, $\rho(r) = e^{-c \cdot r}$, we test the performance of our algorithm over the set of instances. Example allocation of robots along the sweep frontiers for the three cases are shown in Fig. 1(a) and Fig. 9. We limited the number of robots to be small so that the trajectories are more easily observed. It can be seen that the trajectories can vary significantly along the sweep frontiers for each case.

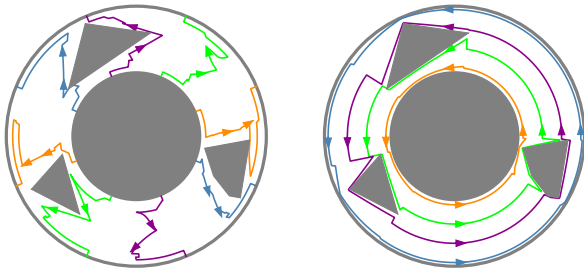


Fig. 9: Example robot trajectories computed by our method for circular and radial sweep use cases, respectively.

In Fig. 10, the computational performance is thoroughly evaluated. As can be observed, our method scales fairly well, taking less than two seconds to handle all cases, even those involving over 100,000 vertices. Interestingly, the experimental running time is almost linear, even using the less efficient Dinic’s algorithm. We suspect the near linear running time is due to the fact that the DAG is a planar graph; studies show that max-flow for planar graphs can be computed in $O(n \log n)$ [30]. Although the graph constructed when solving the “circulation with demand” problem is not planar, large portions are planar. This could reduce the actual time complexity of running the max-flow algorithm.

Different environment settings. Lastly, we evaluate the impact of environmental changes, considering factors including the spatial distribution of polygonal obstacles as well as the size distribution of the obstacles. All in all, three settings are considered: (1) The polygons are regularly distributed and are of similar size, (2) the polygons are randomly distributed and are of similar size, and (3) the polygons are regularly distributed, and their sizes can vary dramatically. The first and the third settings are illustrated in the top row

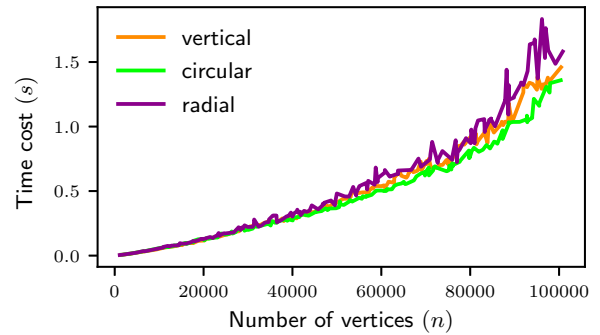


Fig. 10: Running time in seconds with respect to environment complexity (number of vertices).

of Fig. 11. For these settings, we compare the time it takes to compute solutions for many polygonal obstacles and also the number of robots required to achieve 80% probabilistic guarantee under the exponential decay sensing model. As we can observe, there is little difference as the settings change.

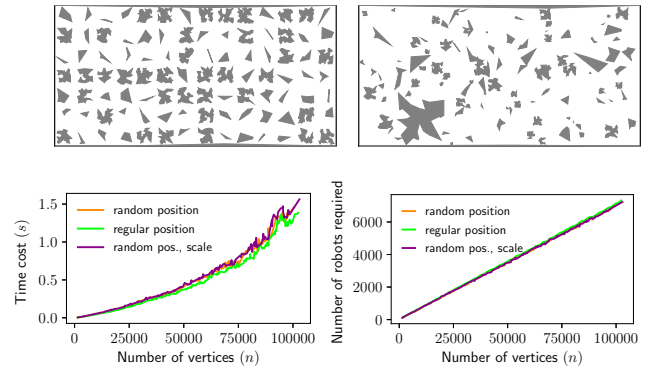


Fig. 11: [top] Random instance with regularly distributed obstacles and instance with random obstacle scales and positions. [bottom] Running time for different randomly generated environments and the number of robots required for different randomly generated environments.

V. CONCLUSION

In this paper, we studied the problem of allocating a minimum number of robots for a sweep schedule with a probabilistic line sensing model, where a desired level of scanning quality can be guaranteed. Towards this, a novel decomposition technique is proposed that generalizes the well-known boustrophedon decomposition. The decomposition leads naturally to a transformation of the problem into a network-flow problem. Due to the decomposition and the transformation, our proposed algorithm runs in low polynomial time and even near $\tilde{O}(n)$ in simulation experiments for polygonal environments, where n is the complexity of the environment, measured as the number of vertices of polygons. Extensive simulation-based evaluation corroborates the effectiveness of our algorithm, which is applicable to multiple types of environments.

REFERENCES

- [1] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," *International Journal of Computational Geometry & Applications*, vol. 9, no. 04n05, pp. 471–493, 1999.
- [2] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM Journal on computing*, vol. 21, no. 5, pp. 863–888, 1992.
- [3] S. M. LaValle, B. H. Simov, and G. Slutzki, "An algorithm for searching a polygonal region with a flashlight," in *Proceedings of the sixteenth annual symposium on Computational geometry*, 2000, pp. 260–269.
- [4] N. M. Stiffler, A. Kolling, and J. M. O’Kane, "Persistent pursuit-evasion: The case of the preoccupied pursuer," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5027–5034.
- [5] A. Kolling and S. Carpin, "The graph-clear problem: definition, theoretical properties and its connections to multirobot aided surveillance," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 1003–1008.
- [6] A. Kolling, A. Kleiner, and S. Carpin, "Coordinated search with multiple robots arranged in line formations," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 459–473, 2017.
- [7] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou, "The complexity of searching a graph," *Journal of the ACM (JACM)*, vol. 35, no. 1, pp. 18–44, 1988.
- [8] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 2000.
- [9] J. O’Rourke, *Art gallery theorems and algorithms*. Oxford New York, NY, USA, 1987, vol. 57.
- [10] D. Shishika, J. Paulos, and V. Kumar, "Cooperative team strategies for multi-player perimeter-defense games," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2738–2745, 2020.
- [11] D. G. Macharet, A. K. Chen, D. Shishika, G. J. Pappas, and V. Kumar, "Adaptive partitioning for coordinated multi-agent perimeter defense," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7971–7977.
- [12] A. K. Chen, D. G. Macharet, D. Shishika, G. J. Pappas, and V. Kumar, "Optimal multi-robot perimeter defense using flow networks," in *International Symposium Distributed Autonomous Robotic Systems*. Springer, 2021, pp. 282–293.
- [13] S. W. Feng and J. Yu, "Optimally guarding perimeters and regions with mobile range sensors," *Robotics: Science and Systems*, 2020.
- [14] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.
- [15] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of field robotics*, vol. 26, no. 8, pp. 651–668, 2009.
- [16] R. N. Haksar, S. Trimpe, and M. Schwager, "Spatial scheduling of informative meetings for multi-agent persistent coverage," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3027–3034, 2020.
- [17] M. Wei and V. Isler, "Coverage path planning under the energy constraint," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 368–373.
- [18] D. Deng, W. Jing, Y. Fu, Z. Huang, J. Liu, and K. Shimada, "Constrained heterogeneous vehicle path planning for large-area coverage," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4113–4120.
- [19] X. Lan and M. Schwager, "Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2415–2420.
- [20] C. G. Cassandras, X. Lin, and X. Ding, "An optimal control approach to the multi-agent persistent monitoring problem," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 947–961, 2012.
- [21] J. Yu, S. Karaman, and D. Rus, "Persistent monitoring of events with stochastic arrivals at multiple stations," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 521–535, 2015.
- [22] J. M. Palacios-Gasós, Z. Talebpour, E. Montijano, C. Sagüés, and A. Martinoli, "Optimal path planning and coverage control for multi-robot persistent coverage in environments with obstacles," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1321–1327.
- [23] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 1. IEEE, 2001, pp. 27–32.
- [24] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 4982–4989.
- [25] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *The international journal of robotics research*, vol. 21, no. 4, pp. 331–344, 2002.
- [26] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006, ch. 6.2.
- [27] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson Education, 2006, ch. 7.7.
- [28] J. Cheriyan and S. Maheshwari, "Analysis of preflow push algorithms for maximum network flow," *SIAM Journal on Computing*, vol. 18, no. 6, pp. 1057–1086, 1989.
- [29] Y. A. Dinitz, "An algorithm for the solution of the problem of maximal flow in a network with power estimation," in *Doklady Akademii Nauk*, vol. 194, no. 4. Russian Academy of Sciences, 1970, pp. 754–757.
- [30] G. Borradaile and P. Klein, "An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph," *Journal of the ACM (JACM)*, vol. 56, no. 2, pp. 1–30, 2009.