Is Limited Information Enough? An Approximate Multi-agent Coverage Control in Non-Convex Discrete Environments

Tatsuya Iwase
Toyota Motor Europe NV/SA
Zaventem, Belgium
tiwase@mosk.tytlabs.co.jp

Aurélie Beynier
Sorbonne Université, CNRS
LIP6
F-75005 Paris, France
aurelie.beynier@lip6.fr

Nicolas Bredeche
Sorbonne Université, CNRS
ISIR
F-75005 Paris, France
nicolas.bredeche@isir.upmc.fr

Nicolas Maudet
Sorbonne Université, CNRS
LIP6
F-75005 Paris, France
nicolas.maudet@lip6.fr

Jason R. Marden
University of California, Santa Barbara
Santa Barbara, USA
jrmarden@ece.ucsb.edu

ABSTRACT

Conventional distributed approaches to coverage control may suffer from lack of convergence and poor performance, due to the fact that agents have limited information, especially in non-convex discrete environments. To address this issue, we extend the approach of [12] which demonstrates how a limited degree of inter-agent communication can be exploited to overcome such pitfalls in one-dimensional discrete environments. The focus of this paper is on extending such results to general dimensional settings. We show that the extension is convergent and keeps the approximation ratio of 2, meaning that any stable solution is guaranteed to have a performance within 50% of the optimal one. The experimental results exhibit that our algorithm outperforms several state-of-the-art algorithms, and also that the runtime is scalable.

KEYWORDS

Multi-agent systems; Coverage control; Communication

ACM Reference Format:
Tatsuya Iwase, Aurélie Beynier, Nicolas Bredeche, Nicolas Maudet, and Jason R. Marden. 2024. Is Limited Information Enough? An Approximate Multi-agent Coverage Control in Non-Convex Discrete Environments. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Coverage control is a fundamental problem in the field of multiagent systems (MAS). The objective of a coverage control problem is to deploy homogeneous agents to maximize a given objective function, which basically captures how distant the group of agents as a whole is from a pre-defined set of Points of Interest (PoI). Coverage control has a wide range of applications, such as tracking, mobile sensing networks or formation control of autonomous mobile robots [3].

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

It is known that, even in a centralized context, finding an optimal solution for the coverage problem is an NP-hard problem [13]. Hence, most studies focus on approximate approaches. In distributed settings, game-theoretical control approaches seek to design agents that will be incentivized to behave autonomously in a way that is well-aligned with the designer's objective. This strategy has proven to be successful in a number of applications (see, e.g. [5]). The situation is made more difficult in practice since agents often have restricted sensing and communication capabilities. Consequently, agents must make decisions based on local information about their environment and the other agents. Unfortunately, algorithms based on local information may suffer from lack of convergence and degraded performance due to miscoordination between the agents. As for the convergence issue, it is known that a move of an agent can affect the cost of agents outside of the neighborhood, and thus, the decrease of the global cost cannot be guaranteed locally, especially in the case of discrete non-convex environment [21]. The degradation of performance can be explained with a worst-case scenario in which only a single agent can perceive a large number of valuable locations within an environment, while a number of other agents cannot perceive these locations.

Recently, Marden [12] made more precise the connection between the degree of locality related to the available information and the achievable efficiency guarantees. He showed that the achievable approximation ratio depends on the individual amount of information available to the agents. Consequently, distributed algorithms are inevitably subjected to poor worst-case guarantees because of the locality of the information used to make decisions. If all agents have full global information as in the case of centralized control, there exist decentralized algorithms that give a 2 approximation ratio. Conversely, under limited information (e.g. Voronoi partitions), no such algorithm provides such an approximation ratio. Rather, the best decentralized algorithm that adheres to these informational dependencies achieves, at most, an n approximation factor, where n is the number of agents. Then, the focus in MAS settings is on how to design agents that, through sharing a limited amount of information with neighborhood communications, achieve an approximation ratio close to 2.

Indeed, different settings exist depending on the assumed communication model, that is, how information can be shared within the system in order to potentially coordinate moves:

(1) agents may not communicate any information and can only be guided by their local perception;
(2) agents may communicate to their neighbors only (where neighbors can be defined as agents in a limited range, or connected via a network of communication);
(3) agents may communicate beyond their neighbors, either via broadcasting, or indirectly via gossip protocols.

Existing game-theoretical approaches can be classified into these types: classical Voronoi-based best-response approaches fall into either (1) or (2), depending on the assumption. Two recent approaches in type (3) explore different directions. Sadeghi *et al.* [16] proposed a distributed algorithm for non-convex discrete settings in which agents have the possibility to coordinate a move with a single other agent (meaning that an agent moves and assumes another agent takes her place simultaneously), possibly beyond their neighbors, when individual moves are not sufficient. However, the algorithm sacrifices the approximation ratio for convergence. Another related distributed approach proposed in [6] is to aim for pairwise optimal partitions (which, they show, are also Voronoi partitions with the additional property that no pairs of agents can cooperate to find a better partitioning of their two regions). Again, while their approach ensures convergence (to good solutions in practice), it does not come with a guaranteed approximation ratio. In a similar spirit, Marden [12] allows coordinated moves with several (possibly, beyond two) agents, but only under the restriction that these agents are within the neighborhood. While this approach achieves a convergent algorithm with an approximation ratio of 2, by sharing only the information of the minimum utility among agents, it is limited in that the only investigated case is the one-dimensional (line) environment. This severely limits real-world applications.

Other studies rely on different techniques. Distributed constraint optimization is a natural way to model this problem [22], and several standard algorithms can be exploited. However, they do not offer approximation guarantees either. Finally, it is possible to try to achieve global optimality by developing approaches akin to simulated annealing. For example, [1] attains global optimum with Spatial Adaptive Play (SAP, a.k.a Boltzmann exploration) and uses random search to escape from the local optimum. However, this approach suffers from a slow convergence rate when the search space is large. There is also no discussion about the relationship between information and efficiency. Note that we focus on the coverage problem, and multiagent path-planning issues [7, 15, 18] are out of the scope of the paper.

In this paper, we extend the algorithm of [12] to any-dimensional, non-convex discrete space and compare this approach with the aforementioned alternative variants of game-theoretical control. The remainder of this paper is as follows. Section 2 introduces the model and existing approaches. In Section 3 we detail the algorithm and prove that it guarantees convergence to a neighborhood optimum solution with an approximation ratio of 2 without any restriction on the dimensionality of the environment, i.e., the same guarantee as in the 1D case. Additionally, we propose a scalable extension of the algorithm and discuss the computation and communication complexity. Experiments reported in Section 4 indeed show that our algorithm outperforms existing ones, along with adhering to the theoretical approximation ratio. Furthermore, the runtime results confirm the scalability of the proposed algorithm.

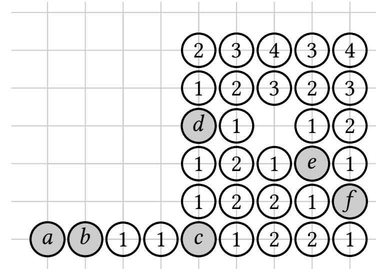

Figure 1: Coverage example with 6 agents: Circles in grey are agents. A number in a white circle shows the Manhattan distance to the closest agent.

2 MODEL

2.1 Coverage Problems

We start with a set of agents $\mathcal{N} = \{1, \dots, n\}$ and a set of resources $C = \{c_1, \dots, c_m\}$. In a coverage problem, resources are locations (or points) in a connected metric space. We assume that this is discrete finite space that is modelled as a connected graph $(C, \mathcal{E})$, where $\mathcal{E}$ is the set of edges that connect two adjacent resources. Though our approach can be extended to continuous settings, we omit the detail due to the space limit. We denote the distance between two resources $a, b \in C$ as $|a - b|$ that is the length of the shortest path.

An allocation x maps each agent i in $\mathcal{N}$ to a resource (i.e., a point) in C. An allocation is thus defined as a vector of resources $x = \langle x_i \in C | i \in \mathcal{N} \rangle$ where x_i is the resource assigned to agent i. Note that each agent must be allocated one and only one resource. An allocation is exclusive i.e., $x_i \neq x_j \; \forall i, j \in \mathcal{N}$ such that $i \neq j$. We denote the set of all possible allocations as $\mathcal{X} = \bigtimes_{i \in \mathcal{N}} \mathcal{X}_i$ where $\mathcal{X}_i$ is the set of possible positions for agent i.

Let $g : \mathbb{R}_+ \to \mathbb{R}_+$ denote a non-increasing function, $v_c \in \mathbb{R}_+$ be the weight of resource $c \in C$, and $C \subseteq C$ be a partial space. Then, the objective function for C is defined as follows:

$$G(x; C) = \sum_{c \in C} \max_{i \in \mathcal{N}} v_c \; g(|x_i - c|). \tag{1}$$

For simplicity, we denote $G(x) = G(x, C)$. The goal of the coverage problem is then to find an optimal allocation $x^* \in \mathcal{X}$ such that:

$$x^* \in \underset{x \in \mathcal{X}}{\operatorname{argmax}} \; G(x). \tag{2}$$

Example 1. *Let us then consider the coverage problem depicted in Figure 1. For the sake of exposure, let us assume that $g(d) = 1/(1+d)$, and $|\cdot|$ is Manhattan distance. Agents are identified by letters $a, b, \dots f$. The environment is a grid world, where each junction is a resource (i.e. location). Circled locations are valued $v_c = 1$, while the others are valued $v_c = 0$. The locations covered by an agent are represented in grey and we indicate the name of the agent. For unoccupied locations,* *we indicate on the node the (Manhattan) distance to the closest agent. Thus, a covered location gives a utility of 1, while a location at a distance of 1 from the closest agent gives a utility 1/2, and so on. The depicted allocation x has value $G(x) = 1 + 1 + \frac{1}{2} + \cdots = 16.4$. It is clearly sub-optimal. The reader can check that the allocation x' where agent e has moved one location up would induce $G(x') \simeq 16.8$.*

It is well-known that solving optimally this problem is NP-hard [13]. Nevertheless, thanks to the submodular nature of the objective function G, the centralized greedy algorithm which allocates agents one by one (starting from the empty environment) guarantees $1-1/e \approx 63\%$ of optimal [2]. We will use this algorithm as a baseline for comparison.

2.2 Game-Theoretic Control: Generalities

Since we focus on distributed algorithms, each agent has to make a choice about her position based on the partial information she has about the other agents. We thus adopt a game theoretic approach where each agent computes her best response based on her individual information [14, 17].

In the following, $-i$ will denote the set of agents excluding i: $-i = \mathcal{N} \setminus \{i\}$. A partial allocation for a subset of agents $S \subseteq \mathcal{N}$ will be denoted as $x_S = \{x_i\}_{i \in S}$.

This paper will focus on providing each agent i with a utility function of the form $u_i : \mathcal{X} \to \mathbb{R}$ that will ultimately guide their individual behavior. In the rest of the paper, we formulate the set of choices of agent i as $\mathcal{X}_i(x)$ assuming that it depends on a given allocation of all agents. For simplicity, we denote $\mathcal{X}(x) = \underset{i \in \mathcal{N}}{\times} \mathcal{X}_i(x)$.

When each agent $i \in \mathcal{N}$ selects the position $x_i \in \mathcal{X}_i(x)$ that maximizes her utility given the other agent's positions x_{-i}, the resulting allocation $x = \langle x_1 \cdots x_n \rangle$ can be a (pure) Nash equilibrium such that:

$$\forall i \in \mathcal{N}\ u_i((x_i, x_{-i})) \geq u_i((x'_i, x_{-i})), \forall x'_i \in \mathcal{X}_i(x), x_i \neq x'_i. \quad (3)$$

In general, the efficiency of a Nash equilibrium $G(x)$ can be smaller than the optimal value $G(x^*)$. One of the reasons for this suboptimality is the miscoordination among self-interested agents, as agents are required to make independent decisions in response to available information. Other sources of suboptimality come from the structure of the agents' utility functions and available choice sets. The (worst-case, among all instances) ratio between the worst Nash equilibria and the social optimum is known as the price of anarchy (PoA) [11]. Here, the choice set $\mathcal{X}_i$ could encode physical constraints on choices, i.e. an agent can only physically choose among a subset of choices given the behavior of the collective x. Alternatively, the choice set could encode information availability of the agent, e.g. it is the set of choices for which the agent can evaluate its utility. Regardless of the interpretation, it is important to highlight that the structure of the choice sets can significantly alter the structure and efficiency of the resulting equilibria which we discuss in the next section.

2.3 Local Information

Several approaches in game theory seek to exploit the fact that agents typically have limited sensing power and only a local view of the situation. Marden [12] analyzed how miscoordination among agents with limited information leads to inefficient Nash equilibria. To this end, he introduced the concept of *information set* which is the set of choices each agent can perceive and compute the resulting utilities. In this paper, $\mathcal{X}_i(x)$ corresponds to the information set that is the set of locations agent i can perceive based on spatial proximity. With this notion, an allocation is a Nash equilibrium (Equation 3) if agents are at least as happy as any choice for which they can evaluate their utility. Observe that the information set $\mathcal{X}_i(x)$ is a state-dependent notion: the local information available may vary depending on the current allocation x.

To model to what extent the information is localized, Marden defined the following *redundancy index* associated to the agents' local information sets $\{\mathcal{X}_i\}_{i \in \mathcal{N}}$:

$$f = \min_{x \in \mathcal{X}} \min_{y \in C} |\{i \in \mathcal{N} : y \in \mathcal{X}_i(x)\}|. \quad (4)$$

Intuitively, f represents the minimum number of agents that perceive the same resource available for their choice. In particular, we note that:

- $f > 0$ guarantees that all the locations are always a possible choice for some agent of the system;
- If there is an allocation x where a resource is a possible choice for only one agent and all other resources are possible choices for at least one agent, then $f = 1$.
- $f = n$ is the extreme case of full information access where all resources are possible choices for all agents.

2.4 Linking Information and Inefficiency

Intuitively, local information can cause distributed systems based on game-theoretic control to get stuck in an inefficient allocation. Indeed, some agents who may obtain higher utilities for a resource may not have access to this information. The redundancy index hence gives insights into the amount of information available to the agents and about guarantees on the worst-case ratio.

Marden [12] investigated this interplay, in the broader context of resource allocation games. The global objective function G is assumed to be monotone submodular i.e., it satisfies the following conditions:

$$\begin{aligned} &G(x_T) \geq G(x_S), \forall S \subseteq T \subseteq N. \\ &G(x_S) - G(x_{S \setminus \{i\}}) \geq G(x_T) - G(x_{T \setminus \{i\}}), \forall S \subseteq T \subseteq \mathcal{N}, \forall i \in S. \end{aligned} \quad (5)$$

and satisfies two further properties. First, the utility of each agent is greater than her marginal contribution:

$$u_i(x) \geq G(x) - G(x_{-i}). \quad (6)$$

Second, social welfare is less than the global objective:

$$\sum_{i \in \mathcal{N}} u_i(x) \leq G(x). \quad (7)$$

In the later section, we will see that the global objective function and the utility function of the conventional coverage control both satisfy these assumptions.

The following theorem shows how the value of f impacts the efficiency of Nash equilibrium allocations:

Theorem 1 (from [12]). *If G is a monotone submodular set function and u_i satisfies (6) and (7), the worst case efficiency of Nash* *equilibrium x is lower bounded by*

$$G(x) \geq \frac{f}{n+f} G(x^*). \tag{8}$$

Furthermore, there exists a case such that:

$$G(x) = \frac{f}{n} G(x^*). \tag{9}$$

Thus, we know that the approximation ratio of a distributed allocation algorithm can be 2 in the case of full information ($f = n$), because $G(x) \geq \frac{1}{2} G(x^*)$ in this case (from Equation 8). Also, it is impossible to guarantee an approximation ratio better than n in case of $f = 1$ (from Equation 9). We will use this result in the later section to analyze the efficiency of coverage control algorithms.

2.5 Voronoi-Based Control

Most standard approaches for coverage control are based on Voronoi partitioning [8]. A Voronoi partition divides the space into local regions for each agent. Formally, for a given allocation x, a Voronoi partition $\mathcal{V}_i(x; C)$ of space C is defined as follows:

$$\mathcal{V}_i(x; C) = \{c \in C | i = \operatorname*{argmin}_{j \in \mathcal{N}} |x_j - c|\}. \tag{10}$$

When ties occur (i.e., when several agents are at the same minimal distance from some location), agents are prioritized lexicographically. Let $\mathcal{P} = (\mathcal{P}_1, \ldots, \mathcal{P}_N)$ denote a partition of the space. In the rest of the paper, we define the utility function of agent i depending on partition $\mathcal{P}$ as follows:

$$u_i(x, \mathcal{P}) = \sum_{c \in \mathcal{P}_i} v_c \, g(|x_i - c|). \tag{11}$$

In this subsection, we assume that the locations an agent can perceive are limited to those within their Voronoi region, that is $\mathcal{P}_i = \mathcal{X}_i(x) = \mathcal{V}_i(x; C)$. Therefore, the utility function (Equation 11) satisfies the assumptions of Equation 6 and Equation 7. We also denote the neighborhood of i as the agents connected in the dual Delaunay graph of the Voronoi partition of i, i.e., the neighborhood of $\mathcal{N}_i \subseteq \mathcal{N}$ are the agents $j \in \mathcal{N}$ whose Voronoi region is connected to the Voronoi region of i. Voronoi partition plays an important role in distributed coverage control because agents can compute the best responses improving the objective function locally, with limited communication. The process is just a sequence of best response updates (in the sense defined above) of the different agents to compute their next locations in their local Voronoi region. Once agents are assigned these new locations, Voronoi regions are updated and the process iterates, until convergence. However, in the non-convex discrete setting the move of an agent within its partition can affect not only neighbors, but the whole set of agents in the worst case [21]. We will see an example later in Figure 5. In theory, the guarantee of convergence requires avoiding moves that could impact beyond an agent's neighbors [16, 21].

Observe that the Voronoi partition induces a redundancy index of $f = 1$ because all the points in Voronoi regions $\mathcal{X}_i$ are available only for agent i. Then, the efficiency $G(x)$ can be $1/n$ of the optimum value in the worst case.

Example 2. *(Ex. 1, cont.). Figure 2 indicates the Voronoi region of each agent by coloring the locations in the same color. Recall that ties are broken lexicographically. Figure 2 gives the corresponding neighboring graph: agent e has 3 neighbors $\{c, d, f\}$. The utilities of the agents are as follows: $u_a = 1$, $u_b = 1.5$, $u_c \simeq 3.2$ $u_d = 4.2$ $u_e \simeq 5$ and $u_f = 1.5$*

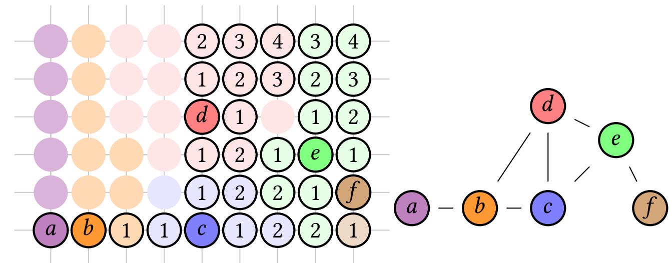

Figure 2: The environment of Example 1 with Voronoi partitions and the corresponding neighborhood graph

As mentioned in the introduction, the communication requirements of these approaches is very low, as agents never need to exchange beyond their neighbors in the Delaunay graph (note that this can also be achieved with a range-limited gossip protocol, as long as an appropriate motion protocol is designed [6]). In the next section, we introduce our protocol which assumes agents can always communicate with their neighbors, and requires them in addition to build a communication tree to spread a limited amount of information to facilitate coordination. Going beyond this communication model is left for future work.

3 A NEIGHBORHOOD OPTIMAL ALGORITHM

To address the inefficiency issue of coverage control through inter-agent communication, we extend the solution for 1-dimensional space [12] to general settings by making agents share additional information.

3.1 Key Notions

The idea of the algorithm is to incrementally compute an allocation x based on a partition $\mathcal{P}$, which is not necessarily a Voronoi partition. Note that the neighborhood $\mathcal{N}_i(\mathcal{P})$ is also defined over $\mathcal{P}$, as the set of agents that have partitions adjacent to $\mathcal{P}_i$. The algorithm is anytime and updates a solution $(x, \mathcal{P})$ for each iteration.

We follow [12] and define the maximum gain in G when adding k new agents into space C, as:

$$M_k(x, C) = \max_{y_1, \ldots, y_k \in C} G(y_1, \ldots, y_k, x; C) - G(x; C), \tag{12}$$

where this best allocation of the k new agents is denoted as follows:

$$B_k(x, C) = \operatorname*{argmax}_{y_1, \ldots, y_k \in C} G(y_1, \ldots, y_k, x; C) - G(x; C). \tag{13}$$

Note that determining the locations $y_1, \ldots y_k$ maximizing Equation 13 is actually an NP-hard problem since it consists in general in solving a multiagent coverage problem.

We denote $\mathcal{P}_{ij} = \mathcal{P}_i \cup \mathcal{P}_j$. Informally, we say that an allocation is *neighborhood optimal* when no coalition of agents within the same neighborhood could reallocate themselves in a way which would improve the global objective, and would not benefit enough from accommodating a third agent in their combined regions. We will make these notions precise below. But before going into the details, we state our main result:

Theorem 2 (Convergence with Performance guarantee). *Under our communication model, Algorithm 1 terminates in a neighborhood optimum allocation. Its approximation ratio is 2.*

Note that this approximation ratio of 2 is equal to the lower bound predicted by Equation 8 in case of full information ($f = n$). In our case it is achieved by only requiring agents to communicate beyond their neighborhood (i) the utility and location of the worst-off agent, and (ii) the identity of the agent which would contribute the most to the global objective by adding an agent in her region. In case of ties a deterministic choice mechanism is used, for instance based on agents' id. Hence, the following notations will be useful:

$$i^+_{\max}(x,\mathcal{P}) = \operatorname*{argmax}_{i\in\mathcal{N}} M_1(x_i;\mathcal{P}_i) \qquad V(x,\mathcal{P}) = \max_{i\in\mathcal{N}} M_1(x_i;\mathcal{P}_i)$$
$$i_{\min}(x,\mathcal{P}) = \operatorname*{argmin}_{i\in\mathcal{N}} u_i(x;\mathcal{P}) \qquad u_{\min}(x,\mathcal{P}) = \min_{i\in\mathcal{N}} u_i(x;\mathcal{P})$$

Example 3. *(Ex. 1, cont.). We have $i_{\min} = a$, since $u_a = 1$. Furthermore, the agents which would contribute to $G(x)$ the most from adding a single agent within their regions are agents e and d. For instance, adding an agent (depicted as '+') would induce G of 6.5 in her region, thus $M_1(x_e,\mathcal{P}_e) \simeq 6.5 - 5 = 1.5$ (See Figure 3 for the illustration). We assume $i^+_{\max} = d$ by lexicographic tie-breaking.*

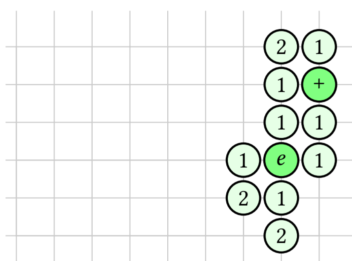

Figure 3: Agent e would contribute to $G(x)$ the most from the addition of an agent (depicted as '+') in her region

3.2 High Level Description of the Algorithm

The principle of our algorithm follows [12], but is adapted to the 2D setting. Essentially, the idea is for neighboring agents to compare what they would gain by optimizing over their combined regions, or by optimizing over their combined regions with a third agent (which is necessarily at least as good). When the difference between these two situations is significant enough (larger than the utility of the worst-off agent), the decision is to make room for a third agent. This way, the space left open can eventually be filled.

As mentioned before, agents will communicate a limited amount of information. Spreading the information of $i_{\min}$ and $i^+_{\max}$ within the system can be achieved by standard distributed algorithms, eg. flooding and gossip protocols [19]. At the end of this process, a communication spanning tree is built. Figure 4 shows a communication tree obtained for Example 1. In our algorithm, interaction

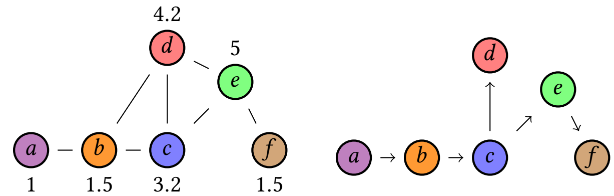

Figure 4: (Left): the neighborhood graph in Example 1. The numbers represent utilities. (Right): a communication tree in convergence. In this case, agent 'a' is $i_{\min}$ and therefore the root of the tree.

takes place in the neighborhood defined by such communication trees, which are updated when required. In the following we denote by *parent(i,$\mathcal{P}$)* the parent of agent i, by *neigh*$(S,\mathcal{P})$ the neighbors of a set of agents S, and by $E(\mathcal{P})$ the set of all pairs of neighboring agents. All these notions are understood as restricted to the current communication tree (in its undirected version for the notion of neighborhood).

As in [12], it is useful to classify the solution $(x,\mathcal{P})$ into 4 states. First, the solution space is divided into the following two states Z_1 and Z_2, by checking if $i^+_{\max}$ would gain more from adding an agent in her region:

$$Z_1 = \{(x,\mathcal{P}) | V(x,\mathcal{P}) > u_{\min}(x,\mathcal{P})\}, \tag{14}$$
$$Z_2 = \{(x,\mathcal{P}) | V(x,\mathcal{P}) \le u_{\min}(x,\mathcal{P})\}. \tag{15}$$

In Z_2, no single agent would contribute enough from accommodating $i_{\min}$. Next the solutions in Z_2 are further classified depending on whether integrating a further agent in the neighborhood of two agents could induce a significant enough marginal gain. In the following state Z_3, it is not the case:

$$Z_3 = \{(x,\mathcal{P}) \subseteq Z_2 | \; M_3(\emptyset,\mathcal{P}_{ij}) - M_2(\emptyset,\mathcal{P}_{ij}) \le u_{\min}(x,\mathcal{P}),\ \forall (i,j) \in E(\mathcal{P})\} \tag{16}$$

Equation 16 means that the gain in the neighborhood optimum by accommodating $i_{\min}$ in $\mathcal{P}_{ij}$ cannot be larger than $u_{\min}$.

Lastly, a solution is classified as state Z_4 if neighborhood optimality [20] is achieved for all agents. Z_4 is the terminal state that is reached from a solution in Z_3 if it satisfies the following condition:

$$Z_4 = \{(x,\mathcal{P}) \subseteq Z_3 | \; u_i(x,\mathcal{P}) + u_j(x,\mathcal{P}) = M_2(\emptyset,\mathcal{P}_{ij}),\ \forall (i,j) \in E(\mathcal{P})\} \tag{17}$$

Example 4. *(Ex. 1, cont.) The allocation x depicted is in state Z_1 since $V(x,\mathcal{P}) = 1.5 > 1 = u_a$, with $i_{\min} = a$.*

Given this classification, we propose a distributed algorithm returning an equilibrium solution $(x,\mathcal{P})$. The algorithm is summarized in Algorithm 1. Firstly, all agents initialize $\mathcal{P}_i$ with a Voronoi partition based on geodesic distances [9] (Line 2). As far as the state is not Z_4, agents build a communication tree and share the necessary information, $u_{\min}$, $x_{i_{\min}}$ and $i^+_{\max}$ based on a consensus algorithm via neighborhood communication. (Lines 3-5).

Algorithm 1 Neighborhood optimum algorithm

1: **procedure** x =NeighborOpt(x, C)
2: $\quad \mathcal{P}_i = \mathcal{V}_i(x; C),\ \forall i \in \mathcal{N}$
3: $\quad$ **while** $(x, \mathcal{P}) \notin Z_4$ **do**
4: $\quad\quad$ Build communication tree
5: $\quad\quad$ Communicate $u_{\min}(x, \mathcal{P})$, $x_{i_{\min}}(x, \mathcal{P})$ and $i^+_{\max}(x, \mathcal{P})$
6: $\quad\quad$ **if** $(x, \mathcal{P}) \in Z_1$ **then**
7: $\quad\quad\quad i \leftarrow i^+_{\max}$
8: $\quad\quad$ **else**
9: $\quad\quad\quad$ pick $i \in \mathcal{N}$
10: $\quad\quad j \leftarrow parent(i, \mathcal{P})$
11: $\quad\quad$ **if** $i_{\min}(x, \mathcal{P}) = j$ **or**
12: $\quad\quad\quad M_3(\emptyset, \mathcal{P}_{ij}) - M_2(\emptyset, \mathcal{P}_{ij}) \le u_{\min}(x, \mathcal{P})$ **then**
13: $\quad\quad\quad x_{\{i,j\}} \leftarrow B_2(\emptyset, \mathcal{P}_{ij})$ **(Step a)**
14: $\quad\quad\quad \mathcal{P}_k \leftarrow \mathcal{V}_k(x', \mathcal{P}_{ij}),\ \forall k \in \{i, j\}$
15: $\quad\quad$ **else**
16: $\quad\quad\quad$ Consider a virtual agent l **(Step b)**
17: $\quad\quad\quad x' \leftarrow B_3(\emptyset, \mathcal{P}_{ij})$
18: $\quad\quad\quad \mathcal{P}'_k \leftarrow \mathcal{V}_k(x', \mathcal{P}_{ij}),\ \forall k \in \mathcal{N}'_i = \{1, 2, 3\}$
19: $\quad\quad\quad \tilde{i}_{\min} = \underset{k \in neigh(\{i,j\}, \mathcal{P})}{\operatorname{argmin}} |x_j - x_{i_{\min}}|$
20: $\quad\quad\quad k_l = \underset{k \in neigh(\tilde{i}_{\min}, \mathcal{P}) \cap \mathcal{N}'_i}{\operatorname{argmin}} |x'_k - x_{\tilde{i}_{\min}}|,\ x_l \leftarrow x'_{k_l}$
21: $\quad\quad\quad x_{\{i,j\}} \leftarrow x' \setminus \{x_l\}$
22: $\quad\quad\quad i_+ \leftarrow \underset{k \in neigh(l, \mathcal{P}) \cap \{i,j\}}{\operatorname{argmin}} |x_j - x_l|,\ \mathcal{P}_{i_+} \leftarrow \mathcal{P}_{i_+} \cup \mathcal{P}_l$
23: $\quad$ Return $(x, \mathcal{P})$

If the algorithm is in state Z_1, the agent $i^+_{\max}$ is picked (Line 9), otherwise a random agent is picked. Note that in a distributed fashion, this could be done for instance by sharing 'done/undone' status and agent IDs, to pick up the 'undone' agent with the smallest ID. Alternatively, agents could probabilistically move (probability α) or not (probability $1 - \alpha$). Agent i retrieves her parent (denoted j) in the current communication tree. The activated agent i then checks whether the combined regions could accommodate another agent (Line 12).

- **Step a:** If $\mathcal{P}_{ij}$ cannot accommodate another agent or $i_{\min}$ belongs to this neighborhood, agent i computes a neighborhood optimum for the pair of agents (i, j) and the allocation is implemented (Line 13).
- **Step b:** If $\mathcal{P}_{ij}$ can accomodate another agent, it computes new locations x' for agents i and j, together with an additional agent l (Line 17). Then, the algorithm computes a new partition for these 3 agents by splitting $\mathcal{P}_{ij}$ based on the new locations x' (Line 18). To allocate the new partition, the algorithm finds the agent $\tilde{i}_{\min} \notin \{i, j\}$ that is the closest to $i_{\min}$ in the communication tree. Then the partition closest to $\tilde{i}_{\min}$ and adjacent to $\mathcal{P}_{\tilde{i}_{\min}}$ is allocated to l first (Line 20). The other partitions are allocated to i and j, for example by an optimal matching algorithm so as to minimize the moves of the agents (Line 21 [4]). Lastly, $\mathcal{P}_l$ is merged to the partition of the agent i_+ that is the closest and adjacent to l (Line 22).

In both cases, agents allocate the partition to avoid collisions after they redevide the neighborhood. Note that the algorithm updates the partition $\mathcal{P}_j$ of only neighbors $(i, j) \in E$ and does not affect the other agents outside of (i, j) (Figure 5). This avoids issues with the convergence of the algorithm in non-convex discrete settings, as discussed in [21].

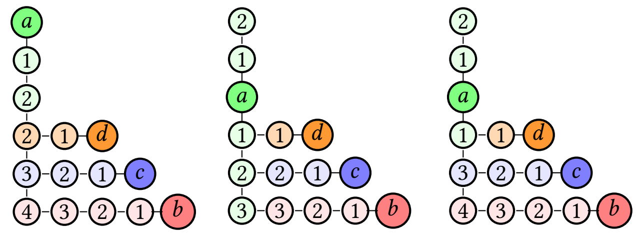

Figure 5: A pathological non-convex discrete example. Different from grid spaces, nodes are connected by edges. Left: Voronoi partition. Middle: A move of agent a changes partitions outside of the neighborhood. Right: Changes in $\mathcal{P}$ are confined inside the neighborhood.

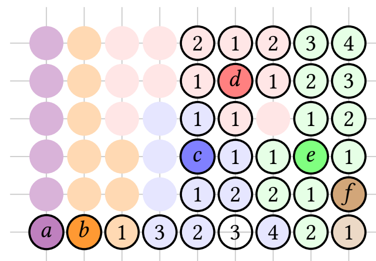

Figure 6: After step b, c and d have reallocated themselves in $\mathcal{P}_c \cup \mathcal{P}_d$ assuming a third agent would join, leaving the empty position (unfilled) as close as possible from a ($i_{\min}$)

Let us illustrate this step on our example (Figure 6). For instance, assuming d is picked as the $i^+_{\max}$ agent, this agent interacts with c, her parent in the communication graph. Upon evaluating the optimal partition with a third agent, they assess that the gain is higher than $u_{\min}$. As a result, they move and make room for a third agent. As discussed before, note that the region is not updated with respect to that of agent b at this stage.

Due to the space limitation, the complete proof of Theorem 2 is in the supplementary material [10]. It follows the line of reasoning described in [12] but adapted to our setting. Briefly, it proves that the following potential function always increases for each iteration of the algorithm. Since the solution space is finite or compact, then the algorithm terminates.

$$\phi(x, \mathcal{P}) = \sum_{i \in \mathcal{N}} u_i(x, \mathcal{P}) + [V(x, \mathcal{P}) - u_{\min}(x, \mathcal{P})]. \tag{18}$$

Also, the proof sketch of the approximation ratio is as follows. Because of the submodularity of G and Equation 16, adding agents in optimal allocation x^* to the same number of agents allocated by the algorithm does not make G double. Formally, $G(x^*) \le G(x, x^*) \le 2G(x)$.

3.3 Special Case

It is interesting to observe that the algorithm provides an optimal solution in the special case where there are exactly as many points of interest as agents. Let $C_+ \subseteq C$ be a set of all important points such that $v_c = 1$, $\forall c \in C_+$. Then other points are less important as $v_{c'} \ll 1$, $\forall c' \in C \setminus C_+$. The algorithm guarantees an optimal solution when agents can cover all the important points as follows.

Theorem 3. *If $N = |C_+|$, Algorithm 1 converges to an optimal solution.*

Proof. Note that each agent is allocated to a point $c \in C_+$ in an optimal solution. Now let us assume that the algorithm converges to a sub-optimal solution $(x, \mathcal{P})$. In this case, some agents including $i_{\min}$ do not have any points $c \in C_+$ in their partitions, due to the neighborhood optimality. Then, there is at least one agent i whose partition $\mathcal{P}_i$ includes more than two points in C_+. This violates the condition (15), which must be satisfied in the convergent state Z_4. This is a contradiction. □

Even though the problem itself is no longer difficult in that case, it is noteworthy that the strategy employed in our algorithm guarantees optimality.

3.4 Communication Requirements

As shown in Theorem 2, the algorithm achieves the approximation ratio of 2, by communicating the minimum degree of information, $u_{\min}$, $i^+_{\max}$ and $x_{i_{\min}}$. The agent with the maximal gain from an additional agent in her region $i^+_{\max}$ is used to focus on the area to be reallocated in Algorithm 1. It is convenient to use it, but we note that it is not strictly required to make the algorithm work (see for instance [12]). The minimum utility, on the other hand, $u_{\min}$ is the key to proving the approximation ratio based on Equation 16. The location $x_{i_{\min}}$ is required to extend the algorithm for 1-dimensional setting in [12] to more general settings.

As for the communication complexity of the algorithm, we start by bounding the convergence rate, which is the number of iterations before convergence. Let $d_{\max}$ be the upper bound of the distance between any two points in the environment C, and Δv be the resolution limit of the weight v_c. In the case of discrete settings, the potential function $\phi(x, \mathcal{P})$ consists of the elemental term $v_c\, g(|x_i - c|)$ and then the improvement in the potential function for each iteration is lower bounded by $\epsilon = \Delta v \cdot g(d_{\max})$. (In the case of continuous settings, we can regard ϵ as the agents' resolution limit of utility). Note that the convergence of the algorithm is guaranteed by Theorem 2. For each iteration, Algorithm 1 checks if the potential function can be improved or not. Before convergence, at least one out of n agents improves the potential function. Then the convergence rate α is bounded as $\alpha \leq n[\phi(x^*, \mathcal{P}^*) - \phi(x, \mathcal{P})]/\epsilon$, where $(x, \mathcal{P})$ and $(x^*, \mathcal{P}^*)$ are the initial allocation and the allocation after convergence, respectively.

Furthermore, for each iteration, the agents build the communication tree, share the global information, share their private information to compute a neighborhood optimum, and finally share the neighborhood optimum solution. Depending on the exact algorithm used to build the communication tree, the communication burden may vary, but it can be done in $O(n^2)$. The communications to share the global information requires at most $O(n^2)$ messages.

4 EXPERIMENTS

In order to validate the practical efficiency and scalability of our approach, we ran several simulations. First, we evaluate the efficiency with small graphs, then we evaluate the scalability with larger graphs.

In what follows, the nodes in an environment graph are classified into two groups, which are $c \in C_+$ with $v_c = 1$ and $c' \in C \setminus C_+$ with $v_{c'} \ll 1$. Nodes in C_+ and the initial position of agents are allocated uniformly at random in the environment graph. We run 32 simulations for each experiment. All the error bars in the figures show 95% confidence intervals. Note that the environment can be any dimensional space, even though all the graphs are projected into 2D figures. All the numerical results are summarized in Table 1. As for the implementation, we use Python 3.8.12, Red Hat Enterprise Linux Server release 7.9 and Intel Xeon CPU E5-2670 (2.60 GHz), 192 GB memory to run the experiments. The random number generator is initialized by numpy.random.seed at the beginning of the main code, with different seeds for each run of the simulation.

Comparison. The neighborhood optimal approach proposed in Section 3.1, coined in the following as NBO, is compared to the following algorithms:

- (VVP) the vanilla distributed covering algorithm based on Voronoi partitioning, as described in Section 2.5.
- (SOTA) the algorithm of Sadeghi *et al.* [16]. In a nutshell, for a given agent i, the algorithm first tries to perform individual moves to maximize the (local) social welfare of her neighborhood, as defined by the Voronoi partitioning. If no such move is improving, it considers coordinated moves with a single other agent j, in the sense that i would move and j would take the place of agent i. The algorithm first considers neighbors, and then (via neighborhood communication), may consider agents further away. However, these coordinated moves only involve two agents at most.
- (CGR) the centralized greedy algorithm that allocates agents one by one starting from the empty environment. Recall that this algorithm guarantees a performance of $1 - 1/e \approx 63\%$ of the optimal solution.

We evaluate the performance of the algorithms above with different shapes of the environments (Figure 7). In addition to these shapes, we also use the small bridge setting and OR library dataset shown in [21]. More details about the shapes and the way instances are generated are available in the supplementary material.

4.1 Efficiency

First, we evaluate the efficiency of the proposed neighborhood optimum algorithm, by comparing its solution with the optimal one and with solutions returned by VVP, SOTA and CGR. The efficiency is measured as the ratio $G(x)/G(x^*)$ where x is a solution of the algorithm and x^* is an optimal solution. Since finding an optimal solution is NP-hard, the simulation is conducted on a 1D chain with $|C| = 20$, $|C_+| = 10$, and $N = 5$.

In that case, the proposed algorithm outperforms both distributed algorithms (VVP and SOTA) and improves the efficiency by about 21.7% compared to SOTA (Figure 8). The efficiency ratio of NBO is 90% which is much better than the theoretical approximation

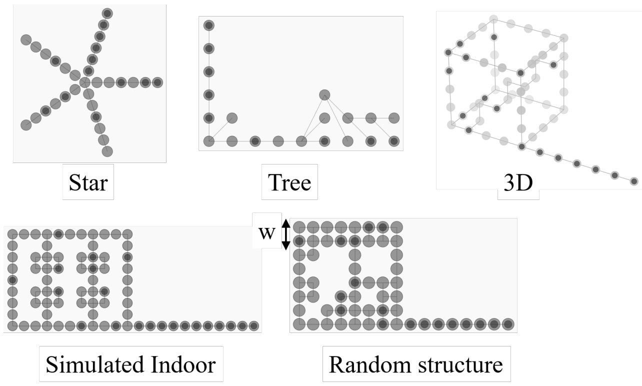

Figure 7: Different shapes of the environment.

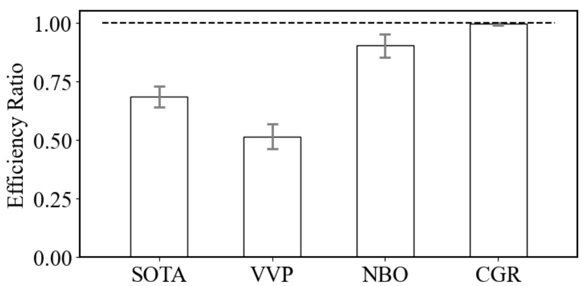

Figure 8: Efficiency ratio $G(x)/G(x^*)$ where x is a solution of each algorithm and x^* is an optimal solution, with $N = 5$.

ratio of 2. We examine the detail with an illustrative example in the supplementary material.

Due to the NP-hardness of the problem, it is difficult to compare with optimal solutions in larger settings. For that reason, we instead compute the efficiency ratio with the centralized greedy (CGR) as a practical baseline. Table 1 summarizes all the results obtained. The quantitative comparison of the efficiency shows similar results as in Figure 8 that the proposed method outperforms the benchmarks in all cases. Note that NBO even sometimes outperforms CGR in some instances, despite the following two disadvantages of NBO compared to CGR: (i) NBO uses only limited information while CGR uses full information, (ii) NBO starts from a random initial state, which is less favorable compared to CGR's iterative approach from the empty environment.

4.2 Scalability

We finally evaluate the scalability of the proposed algorithm by varying the size of the space $|C|$ and the number of agents n (we could run experiments with more than 150 agents, the details can be found in the supplementary material). The results show that:

- The runtime grows with the size $|C|$, as expected. On the largest instances of size $|C| = 192$, with 20 agents, the runtime is above 3 hours.
- The runtime decreases when n increases because the size of partitions $|\mathcal{P}_{ij}|$ also decreases, and thus the opportunities of improvement are more severely constrained.

Shapes	SOTA	VVP	NBO
1D Chains	0.686 ± 0.045	0.515 ± 0.052	0.903 ± 0.050
Stars	0.983 ± 0.008	0.962 ± 0.010	1.017 ± 0.006
Trees	0.952 ± 0.009	0.935 ± 0.009	0.980 ± 0.004
Simulaed indoor	0.895 ± 0.007	0.841 ± 0.010	0.917 ± 0.013
Random ($w = 1$)	0.932 ± 0.010	0.904 ± 0.008	0.963 ± 0.009
Random ($w = 2$)	0.936 ± 0.008	0.901 ± 0.007	0.976 ± 0.009
Small bridge	0.975 ± 0.007	0.955 ± 0.012	1.000 ± 0.002
3D structure	0.960 ± 0.006	0.937 ± 0.007	0.982 ± 0.010
OR library	0.843 ± 0.126	0.963 ± 0.014	0.978 ± 0.006

Table 1: Comparison on different shapes: Efficiency ratio $G(x)/G(x^{CGR})$ where x is a solution of each algorithm and x^{CGR} is a solution of CGR. Each cell shows mean ± standard deviation. Larger values mean better results.

These results demonstrate the applicability of the proposed algorithm to real-world medium-scale problems. For larger-scale settings, further improvements will be required to make the approach viable. It should be emphasized though that the anytime nature of the algorithm makes it relevant even with a limited time budget. Further details about the experiments are deferred to the supplementary material.

5 CONCLUSION

This paper extends the approach of [12] to general dimensional settings for discrete environments, ensuring convergence to a neighborhood optimum solution, even in challenging non-convex scenarios, with an approximation ratio of 2. The communication requirements involve disseminating the value and position of the minimum utility agent, capturing an approximation ratio that surpasses other methods presented in [6, 16]. Interestingly, this minimal level of informational dissemination recaptures the best achievable approximation ratio of 2 for Nash equilibria, which requires all agents to have full information about the environment (e.g., choices of all agents, utility associated with all feasible choices, etc.). While more communication demanding than simple best responses based on Voronoi partitioning, the informational dissemination requirements are manageable, enabling local decision-making rules. Furthermore, our algorithm guarantees optimality in a special subclass of the coverage problem and outperforms state-of-the-art benchmark algorithms in experiments. Future research will focus on improving the algorithm's efficiency by minimizing information transmission or enhancing experimental results. Implementing this algorithm on real robotic systems is another important avenue for further exploration.

ACKNOWLEDGMENTS

This work is partially supported by ONR grant #N00014-20-1-2359 and AFOSR grants #FA9550-20-1-0054 and #FA9550-21-1-0203.

REFERENCES

[1] Gurdal Arslan, Jason R Marden, and Jeff S Shamma. 2007. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control* 129, 5 (2007), 584–596.

[2] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. 2011. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.* 40, 6 (2011), 1740–1766.

[3] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. 2004. Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation* 20, 2 (2004), 243–255.

[4] David F Crouse. 2016. On implementing 2D rectangular assignment algorithms. *IEEE Trans. Aerospace Electron. Systems* 52, 4 (2016), 1679–1696.

[5] Yinon Douchan, Ran Wolf, and Gal A Kaminka. 2019. Swarms Can be Rational. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems.* 149–157.

[6] Joseph W Durham, Ruggero Carli, Paolo Frasca, and Francesco Bullo. 2011. Discrete partitioning and coverage control for gossiping robots. *IEEE Transactions on Robotics* 28, 2 (2011), 364–378.

[7] Enric Galceran and Marc Carreras. 2013. A survey on coverage path planning for robotics. *Robotics and Autonomous systems* 61, 12 (2013), 1258–1276.

[8] Chunkai Gao, Jorge Cortés, and Francesco Bullo. 2008. Notes on averaging over acyclic digraphs and discrete coverage control. *Automatica* 44, 8 (2008), 2120–2127.

[9] Dominik Haumann, Andreas Breitenmoser, Volker Willert, Kim Listmann, and Roland Siegwart. 2011. DisCoverage for non-convex environments with arbitrary obstacles. In *2011 IEEE International Conference on Robotics and Automation.* IEEE, 4486–4491.

[10] Tatsuya Iwase, Aurélie Beynier, Nicolas Bredeche, Nicolas Maudet, and Jason Marden. 2024. Is Limited Information Enough? An Approximate Multi-agent Coverage Control in Non-Convex Discrete Environments. *arXiv preprint arXiv:2401.03752* (2024).

[11] Elias Koutsoupias and Christos Papadimitriou. 1999. Worst-Case Equilibria. In *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science, STACS*, Christoph Meinel and Sophie Tison (Eds.). 404–413.

[12] Jason R Marden. 2016. The role of information in distributed resource allocation. *IEEE Transactions on Control of Network Systems* 4, 3 (2016), 654–664.

[13] Nimrod Megiddo and Kenneth J Supowit. 1984. On the complexity of some common geometric location problems. *SIAM journal on computing* 13, 1 (1984), 182–196.

[14] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. 2007. Algorithmic game theory, 2007. *Book available for free online* (2007).

[15] Keisuke Okumura and Xavier Défago. 2022. Solving Simultaneous Target Assignment and Path Planning Efficiently with Time-Independent Execution. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 32. 270–278.

[16] Armin Sadeghi, Ahmad Bilal Asghar, and Stephen L Smith. 2020. Approximation algorithms for distributed multi-robot coverage in non-convex environments. *arXiv preprint arXiv:2005.02471* (2020).

[17] Yoav Shoham and Kevin Leyton-Brown. 2008. *Multiagent systems: Algorithmic, Game-Theoretic, and logical foundations.* Cambridge University Press. https://doi.org/10.1017/CBO9780511811654

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

[19] Roger Wattenhofer. 2016. Principles of Distributed Computing. (2016). ETZ Zurich.

[20] Yair Weiss and William T Freeman. 2001. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory* 47, 2 (2001), 736–744.

[21] Seung-kook Yun and Daniela Rus. 2014. Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning. *Robotica* 32, 2 (2014), 257–277. https://doi.org/10.1017/S0263574713001148

[22] Roie Zivan, Harel Yedidsion, Steven Okamoto, Robin Glinton, and Katia Sycara. 2015. Distributed constraint optimization for teams of mobile sensing agents. *Autonomous Agents and Multi-Agent Systems* 29, 3 (may 2015), 495–536. https://doi.org/10.1007/s10458-014-9255-3