2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Performance-Aware Self-Configurable Multi-Agent Networks: A Distributed Submodular Approach for Simultaneous Coordination and Network Design

Zirui Xu, Vasileios Tzoumas†

2024 IEEE 63rd Conference on Decision and Control (CDC) | 979-8-3503-1633-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/CDC56724.2024.10886872

***Abstract*— We introduce the first, to our knowledge, rigorous approach that enables multi-agent networks to self-configure their communication topology to balance the trade-off between scalability and optimality during multi-agent planning. We are motivated by the future of ubiquitous collaborative autonomy where numerous distributed agents will be coordinating via agent-to-agent communication to execute complex tasks such as traffic monitoring, event detection, and environmental exploration. But the explosion of information in such large-scale networks currently curtails their deployment due to impractical decision times induced by the computational and communication requirements of the existing near-optimal coordination algorithms. To overcome this challenge, we present the AlterNATing COordination and Network-Design Algorithm (Anaconda), a scalable algorithm that also enjoys near-optimality guarantees. Subject to the agents' bandwidth constraints, Anaconda enables the agents to optimize their local communication neighborhoods such that the action-coordination approximation performance of the network is maximized. Compared to the state of the art, Anaconda is an anytime self-configurable algorithm that quantifies its suboptimality guarantee for any type of network, from fully disconnected to fully centralized, and that, for sparse networks, is one order faster in terms of decision speed. To develop the algorithm, we quantify the suboptimality cost due to decentralization, *i.e.*, due to communication-minimal distributed coordination. We also employ tools inspired by the literature on multi-armed bandits and submodular maximization subject to cardinality constraints. We demonstrate Anaconda in simulated scenarios of area monitoring and compare it with a state-of-the-art algorithm.**

I. Introduction

In the future, distributed teams of agents will be coordinating via agent-to-agent communication to execute tasks such as target tracking [1], environmental mapping [2], and area monitoring [3]. Such multi-agent tasks are modeled in the robotics, control, and machine learning literature via maximization problems of the form

$$\max_{a_i \in \mathcal{V}_i, \forall\, i \in \mathcal{N}} \; f(\{a_i\}_{i\in\mathcal{N}}), \tag{1}$$

where $\mathcal{N}$ is the set of agents, a_i is agent i's action, $\mathcal{V}_i$ is agent i's set of available actions, and $f\colon 2^{\prod_{i\in\mathcal{N}}\mathcal{V}_i} \mapsto \mathbb{R}$ is the objective function that captures the task utility [2]–[14]. Particularly, in information gathering tasks, f is often *submodular* [15]: submodularity is a diminishing returns property, and it emanates due to the possible information overlap among the information gathered by the agents [4]. For example, in target monitoring with multiple cameras at fixed locations, $\mathcal{N}$ is the set of cameras, $\mathcal{V}_i$ is the available directions the camera can point at, and f is the number of targets covered by the collective field of view of the cameras.

†Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109 USA; {ziruixu,vtzoumas}@umich.edu

This work was partially supported by NSF CAREER No. 2337412.

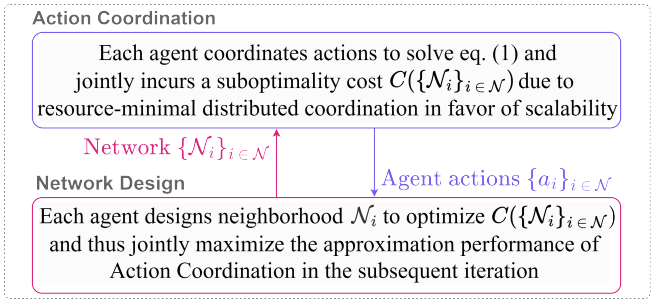


Fig. 1: **Overview of AlterNATing COordination and Network-Design Algorithm (Anaconda).** Starting from an unspecified communication network, and subject to the agents' communication bandwidth and connectivity constraints, Anaconda enables the agents to optimize their local communication neighborhoods such that the action-coordination approximation performance of the whole network is maximized. To this end, Anaconda employs two subroutines, ActionCoordination and NeighborSelection, that alternate optimization. In more detail, given the selected neighborhoods $\{\mathcal{N}_i\}_{i\in\mathcal{N}}$ by NeighborSelection, ActionCoordination instructs the agents to select actions to jointly maximize eq. (1). But ActionCoordination incurs a suboptimality cost $C(\{\mathcal{N}_i\}_{i\in\mathcal{N}})$ due to requiring the agents to coordinate exchanging local information only, prohibiting also multi-hop communication, in favor of decision speed. For this reason, given the agents' bandwidth and connectivity constraints, and the previously selected actions by ActionCoordination, NeighborSelection instructs each agent i to design its neighborhood $\mathcal{N}_i$ to optimize $C(\{\mathcal{N}_i\}_{i\in\mathcal{N}})$ and, thus, maximize the approximation performance of ActionCoordination in the subsequent iteration.

But solving the problem in eq. (1) in real-time is challenging since it is NP-hard [16]. Although polynomial-time algorithms exist that achieve near-optimal solutions for the problem in eq. (1), in the presence of real-world communication delays [17], these algorithms often require high times to terminate —communication delays are caused by the finite speed of real-world communication channels. The reason is that, for an increasing number of agents, the current algorithms require a combination of high number of communication rounds and large inter-agent message lengths, which collectively increase the total delay.

For example, the algorithm in [14], although it achieves the approximation bound $1-1/e$ for eq. (1), which is the best possible [18], can require tenths of minutes to terminate even for 10 agents [11]. This is due to algorithm in [14] requiring near-cubic communication rounds in the number of agents, and inter-agent messages that carry information about all agents, instead of only local information. Similarly, the Sequential Greedy algorithm [15], which is the gold standard in robotics, control, and machine learning [2]–[14], although it sacrifices some approximation performance to enable faster decision speed —achieving the bound $1/2$ instead of the bound $1-1/e$— still requires (i) inter-agent messages that carry information about all the agents and, in the worst case, (ii) a quadratic number of communication

rounds over directed networks [19, Proposition 2], resulting in a communication complexity that is cubic in the number of agents, as we will discuss later in Remark 3.

In a similar vein, the Resource-Aware distributed Greedy (RAG) algorithm [11] aims to sacrifice even further approximation performance in favor of more scalability, by requiring the agents to receive information only from and about neighbors such that each message contains information only about the neighbor that sends it. In more detail, the messages are received directly from each neighbor, via multi-channel communication, assuming a pre-defined directed communication network that respects all agents' communication bandwidths. That way, RAG enables (i) parallel decision-making, instead of only sequential that the Sequential Greedy requires, reducing the total number of communication rounds it (RAG) requires, and (ii) inter-agent messages that contain information about one agent only, instead of multiple agents that the Sequential Greedy requires, thus making the communication of such shorter messages faster in the presence of finite communication speeds. Due to RAG's communication-minimal protocol, RAG has an approximation performance that is the same as the Sequential Greedy when all agents are neighbors with all other —fully centralized coordination— but, when agents coordinate with a few others only, RAG suffers a suboptimality cost as a function of the network topology. Therefore, the following research question arises:

Subject to the agents' bandwidth constraints, how to enable each agent to optimize its coordination neighborhood such that the suboptimality cost due to decentralization is minimized, that is, the action-coordination performance of the multi-agent network is maximized?

To our knowledge, no current work provides distributed algorithms that design the network topology to rigorously optimize the coordination's approximation performance. Heuristic methods are proposed for network optimization yet without being rigorously tied to optimizing the coordination performance [20], [21]. Although [11] quantifies the suboptimality cost due to decentralization, it cannot be leveraged to enable the agents to optimize their neighborhoods since it requires the agents to have oracle access to the actions of non-neighbors, which is impossible in practice in favor of scalability. Works also provide fundamental limits in using the Sequential Greedy for distributed submodular optimization where the agents can select actions ignoring the actions of some of the previous agents [7]–[9]. Particularly, [7], [9] assume a Directed Acyclic Graph (DAG) information-passing communication topology. In this context, these works characterize Sequential Greedy's worst-case performance via graph-theoretic properties of the network. For example, [9] characterizes the worst-case approximation bound by considering the worst-case over all submodular functions, and proves a bound that scales inversely proportional to the independence number of the information graph. Intuitively, the bound scales inversely proportionally to the maximum number of groups of agents that can plan independently. In contrast, in this work, we provide algorithms that allow for the distributed co-design of the network topology based on the submodular function f at hand and the agents' bandwidth constraints. Even if all agents plan independently (fully decentralized network), the guaranteed bound can still be $1/2$, depending on f's curvature (Theorem 1), instead of scaling inversely proportional to the number of agents.

Contributions. We provide the first, to our knowledge, rigorous approach that enables multi-agent networks to self-configure their communication topology to balance the trade-off between scalability and optimality during multi-agent coordination. To this end, we initiate the study of the problem *Distributed Simultaneous Coordination and Network Design* (Section II), and present a scalable online algorithm with near-optimality guarantees (Sections III to V).

Subject to the agents' bandwidth and communication constraints, *the algorithm enables the agents to optimize their local communication neighborhoods such that the action-coordination approximation performance of the whole network is maximized*. The optimization occurs over multiple rounds of local information exchange, where information relay via multi-hop communication is prohibited to curtail the explosion of information exchange, and, thus, to keep low delays due to limited communication speeds. We overview the algorithm in more detail in Fig. 1.

To enable the algorithm, we introduce the first, to our knowledge, quantification of the suboptimality cost during distributed coordination as a function of each agent's neighborhood (Section IV). To this end, we capture the action overlap through f between each agent and its neighbors via a mutual-information-like quantity that we term *Mutual Information between an Agent and its Neighbors* (Definition 3).

The algorithm has the following properties:

a) Anytime Self-Configuration: The algorithm enables each agent to select actions and neighbors based on local information only. Therefore, the algorithm enables the multi-agent system to fluidly adapt to new near-optimal actions and communication topology whenever either new agents are included to or existing agents are removed from the network.

b) Decision Speed: For sparse networks, where the size of each agent's neighborhood is not proportional to the size of the whole network, the algorithm is an order faster than the state-of-the-art algorithms when accounting for the impact that the message length has to communication delays (Section V). The result holds true when the communication cost to the decision speed is at least as high as the computational cost. Particularly, we quantify the algorithm's decision speed in terms of the time needed to perform function evaluations and to communicate with finite communication speeds.

c) Approximation Performance: The algorithm enjoys a suboptimality bound against an optimal fully centralized algorithm for eq. (1) (Section IV). Particularly, the bound:

- Captures the suboptimality cost due to decentralization, *i.e.*, due to communication-minimal distributed coordination. For example, if the network returned by the algorithm is fully connected, then the algorithm's approximation bound becomes $1/2$. This is near-optimal since the best approximation bound for eq. (1) is $1 - 1/e \simeq 0.63$ [16].
- Holds true given any network topology, including directed and (partially) disconnected networks. In contrast, the current algorithms, such as the Sequential Greedy algo-

rithm [15], cannot offer suboptimality guarantees when the network is disconnected since they require the agents to be able to relay information about all others.

Numerical Evaluations. We evaluate Anaconda in simulated scenarios of area monitoring with multiple cameras (Section VI). We evaluate the decision speed of and total area covered by Anaconda with different maximum neighborhood sizes and compare it with the state-of-the-art algorithm DFS-SG [19]. The results are presented in Fig. 2.

All proofs can be found in the extended version [22].

II. Distributed Simultaneous Coordination and Network Design

We define the problem *Distributed Simultaneous Coordination and Network Design*. To this end, we use the notation:

- $\mathcal{V}_{\mathcal{N}} \triangleq \prod_{i\in\mathcal{N}} \mathcal{V}_i$ is the cross product of sets $\{\mathcal{V}_i\}_{i\in\mathcal{N}}$.
- $[T] \triangleq \{1,\dots,T\}$ for any positive integer T;
- $f(\,a\,|\,\mathcal{A}\,) \triangleq f(\,\mathcal{A}\cup\{a\}\,) - f(\,\mathcal{A}\,)$ is the marginal gain of set function $f: 2^{\mathcal{V}} \mapsto \mathbb{R}$ for adding $a\in\mathcal{V}$ to $\mathcal{A}\subseteq\mathcal{V}$.
- $|\mathcal{A}|$ is the cardinality of a discrete set $\mathcal{A}$.
- $\mathcal{E}$ is the set of (directed) communication edges among the agents. $\mathcal{E}$ is designed in this paper.

We also lay down the following framework about the agents' communication network, and their function f.

Communication network. The communication network $\mathcal{G}=(\mathcal{N},\mathcal{E})$ among the agents is **unspecified** a priori, with the goal in this paper being that the agents must optimize the network themselves to best execute the given task.

The resulting communication network *can be directed and even disconnected*. When the network is fully connected (all agents receive information from all others), we call it *fully centralized*. In contrast, when the network is fully disconnected (all agents are isolated, receiving information from no other agent), we call it *fully decentralized*.

Communication neighborhood. When a communication channel exists from agent j to agent i, *i.e.*, $(j\to i)\in\mathcal{E}$, then i can receive, store, and process information from j. The set of all agents that i receives information from is denoted by $\mathcal{N}_i$. We refer to $\mathcal{N}_i$ as agent i's *neighborhood*.

Communication constraints. Each agent i can receive information from up to α_i other agents due to onboard bandwidth constraints. Thus, it must be $|\mathcal{N}_i|\leq\alpha_i$.

Also, we denote by $\mathcal{M}_i$ the set of agents than have agent i within communication reach —not all agents may have agent i within communication reach because of distance or obstacles. Therefore, agent i can pick its neighbors by choosing at most α_i agents from $\mathcal{M}_i$. Evidently, $\mathcal{N}_i\subseteq\mathcal{M}_i$.

Definition 1 (Normalized and Non-Decreasing Submodular Set Function [15])**.** *A set function $f: 2^{\mathcal{V}}\mapsto\mathbb{R}$ is* normalized and non-decreasing submodular *if and only if*

- *(Normalization)* $f(\,\emptyset\,)=0$*;*
- *(Monotonicity)* $f(\,\mathcal{A}\,)\leq f(\,\mathcal{B}\,)$*,* $\forall\,\mathcal{A}\subseteq\mathcal{B}\subseteq\mathcal{V}$*;*
- *(Submodularity)* $f(\,s\,|\,\mathcal{A}\,)\geq f(\,s\,|\,\mathcal{B}\,)$*,* $\forall\,\mathcal{A}\subseteq\mathcal{B}\subseteq\mathcal{V}$ *and* $s\in\mathcal{V}$*.*

Intuitively, if $f(\,\mathcal{A}\,)$ captures the number of targets tracked by a set $\mathcal{A}$ of sensors, then the more sensors are deployed, more or the same targets are covered; this is the non-decreasing property. Also, the marginal gain of tracked targets caused by deploying a sensor s *drops* when *more* sensors are already deployed; this is the submodularity property.

Definition 2 (2nd-order Submodular Set Function [23], [24])**.** *$f: 2^{\mathcal{V}}\mapsto\mathbb{R}$ is* 2nd-order submodular *if and only if*

$$f(s\,|\,\mathcal{C}) - f(s\,|\,\mathcal{A}\cup\mathcal{C}) \geq f(s\,|\,\mathcal{B}\cup\mathcal{C}) - f(s\,|\,\mathcal{A}\cup\mathcal{B}\cup\mathcal{C}), \quad (2)$$

for any disjoint $\mathcal{A},\mathcal{B},\mathcal{C}\subseteq\mathcal{V}$ *($\mathcal{A}\cap\mathcal{B}\cap\mathcal{C}=\emptyset$) and $s\in\mathcal{V}$.*

Intuitively, if $f(\,\mathcal{A}\,)$ captures the number of targets tracked by a set $\mathcal{A}$ of sensors, then *marginal gain of the marginal gains* drops when more sensors are already deployed.

Problem 1 (Distributed Simultaneous Coordination and Network Design)**.** *Each agent $i\in\mathcal{N}$ needs to select a neighborhood $\mathcal{N}_i$ of size at most α_i, and an action a_i such that the agents jointly solve the optimization problem*

$$\max_{\substack{\mathcal{N}_i\subseteq\mathcal{M}_i\\ \forall\, i\in\mathcal{N}}} \;\; \max_{\substack{a_i\in\mathcal{V}_i\\ \forall\, i\in\mathcal{N}}} \; f(\,\{a_i\}_{i\in\mathcal{N}}\,) \;\; \textit{s.t.} \;\; |\mathcal{N}_i|\leq\alpha_i, \quad (3)$$

where each agent i selects their action a_i after coordinating actions with its neighbors only, without having access to information about non-neighbors*, and where $f: 2^{\mathcal{V}_{\mathcal{N}}}\mapsto\mathbb{R}$ is a normalized, non-decreasing submodular, and 2nd-order submodular set function.*

Problem 1 implies that the network and action optimizations are coupled: when the network is fully decentralized (all agents coordinate with no other), the achieved value of f can be lower compared to the value that can be achieved when the network is instead fully centralized (all agents coordinate with all others). For example, consider the target monitoring scenario in Section I: in the fully decentralized setting, all cameras may end up covering the same targets, thus f will equal the number of targets covered by one camera only. In contrast, in the fully centralized setting, the cameras can coordinate and end up covering different targets, thus maximizing the total number of covered targets f.

Remark 1 (Decision speed vs. Optimality)**.** *As demonstrated by the above example, the more centralized a network is, the higher the value of f that can be achieved. But a more centralized network can also lead to lower decision speeds due to an explosion of information passing among all the agents since all agents will coordinate with all others. The goal of this paper is to develop a communication-efficient distributed algorithm that not only requires just a few coordination rounds for convergence; it also needs only short messages to be communicated among agents, thus, accounting for real-world communication delays due to limited communication speeds [25]. For this reason, in particular, Problem 1 requires each agent to (i) coordinate actions only with its neighbors, and (ii) receive information only about them, instead of also about non-neighbors. This is in contrast to standard distributed methods that allow information about the whole network to travel to all agents via multi-hop communication, hence often not reducing the amount of information flowing in the network compared to*

Algorithm 1: AlterNAting COordination and Network-Design Algorithm (Anaconda)

Input: Number of time steps T; agent i's neighbor candidate set $\mathcal{M}_i$; agent i's neighborhood size α_i; objective set function $f : 2^{\mathcal{V}_\mathcal{N}} \mapsto \mathbb{R}$.

Output: Agent i's action $a_{i,t}$ and neighbor set $\mathcal{N}_{i,t}$ at each $t \in [T]$.

1: $\mathcal{N}_{i,0} \leftarrow \emptyset, \forall i \in \mathcal{N}$;
2: **for** each time step $t \in [T]$ **do**
3: $\quad a_{i,t} \leftarrow$ ActionCoordination$([T], \mathcal{V}_i, f)$;
4: $\quad \mathcal{N}_{i,t} \leftarrow$ NeighborSelection$(a_{i,t}, [T], \mathcal{M}_i, \alpha_i, f)$;
5: $\quad$ **receive** neighbors' actions $\{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}$ and **update** ActionCoordination (per lines 6-8) and NeighborSelection (per lines 6-11);
6: **end for**

Algorithm 2: ActionCoordination

Input: Number of time steps T; agent i's action set $\mathcal{V}_i$; objective set function $f : 2^{\mathcal{V}_\mathcal{N}} \mapsto \mathbb{R}$.

Output: Agent i's action $a_{i,t}$ at each $t \in [T]$.

1: $\eta_1 \leftarrow \sqrt{8 \log |\mathcal{V}_i| \,/\, T}$;
2: $w_1 \leftarrow \left[w_{1,1}, \ldots, w_{|\mathcal{V}_i|,1}\right]^\top$ with $w_{v,1} = 1, \forall a \in \mathcal{V}_i$;
3: **for** each time step $t \in [T]$ **do**
4: $\quad$ **get** distribution $p_t \leftarrow w_t \,/\, \|w_t\|_1$;
5: $\quad$ **draw** action $a_{i,t} \in \mathcal{V}_i$ **from** p_t;
6: $\quad$ **input** $a_{i,t}$ **to** NeighborSelection and **receive** neighbors' actions $\{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}$;
7: $\quad r_{a,t} \leftarrow f(\, a \,|\, \{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}\,)$ and **normalize** $r_{a,t}$ **to** $[0,1]$, $\forall a \in \mathcal{V}_i$;
8: $\quad w_{a,t+1} \leftarrow w_{a,t} \exp\left(\eta_1 \, r_{a,t}\right), \forall a \in \mathcal{V}_i$;
9: **end for**

fully centralized coordination, and hence often introducing impractical communication delays [11], [17].

III. Alternating Coordination and Network-Design Algorithm (Anaconda)

We present the AlterNAting COordination and Network-Design Algorithm (Anaconda) for Problem 1. Anaconda aims to approximate a solution to Problem 1 by alternating the optimization for action coordination and neighbor selection. A description of the algorithm is given in Fig. 1.

Both action coordination and neighborhood selection take the form of Multi-Armed Bandit (MAB) problems, therefore, in the following, we first present the MAB problem (Section III-A). Then, we will present the algorithms ActionCoordination (Section III-B) and NeighborSelection (Section III-C).

A. Multi-Armed Bandit Problem

The adversarial *Multi-Armed Bandit* (MAB) problem [26] involves an agent selecting a sequence of actions to maximize the total reward over a given number of time steps [26]. The challenge is that, at each time step, the reward associated with each action is unknown to the agent a priori, becoming known only after the action has been selected. To rigorously present the MAB problem, we use the notation:

- $\mathcal{V}$ denotes the available action set;
- $v_t \in \mathcal{V}$ denotes the agent's selected action at time t;
- $r_{v_t,t} \in [0,1]$ denotes the reward that the agent receives by selecting action v_t at t.

Problem 2 (Multi-Armed Bandit [26])**.** *Assume an operation horizon of T time steps. At each time step $t \in [T]$, the agent must select an action v_t such that the regret*

$$\text{MAB-Reg}_T \triangleq \max_{v \in \mathcal{V}} \sum_{t=1}^{T} r_{v,t} \;-\; \sum_{t=1}^{T} r_{v_t,t}, \qquad (4)$$

*is sublinear in T, where for **full-information feedback**, the rewards $r_{v,t} \in [0,1]$ for all $v \in \mathcal{V}$ become known to the agent* after *v has been executed at each t; whereas for **bandit feedback**, only the reward $r_{v_t,t} \in [0,1]$ becomes known to the agent* after *v has been executed.*

Problem 2 asks for MAB-Reg$_T$ to be sublinear, *i.e.*, MAB-Reg$_T/T \to 0$ for $T \to +\infty$, since this implies that the agent asymptotically chooses optimal actions even though the rewards are unknown a priori [26].

Problem 2 presents two versions of MAB, one with full-information feedback, and one with bandit feedback. The difference between them is that, at each $t \in [T]$, in full-information feedback the rewards of all $v \in \mathcal{V}$ are revealed, even though only one action is selected; while in bandit feedback, only the reward of the selected action is revealed.

B. Action Coordination

We present ActionCoordination and its performance guarantee. To this end, we introduce the coordination problem that ActionCoordination instructs the agents to simultaneously solve and show that it takes the form of Problem 2 with full-information feedback. We use the definitions:

- $\mathcal{A}_t \triangleq \{a_{i,t}\}_{i \in \mathcal{N}}$ is the set of all agents' actions at t;
- $\mathcal{A}^{\text{OPT}} \in \arg\max_{a_i \in \mathcal{V}_i, \forall\, i \in \mathcal{N}} f(\{a_i\}_{i \in \mathcal{N}})$ is the optimal actions for agents $\mathcal{N}$ that solve eq. (1);
- $\mathcal{N}_i^\star$ is the optimal neighborhood corresponding to $\{a_{i,t}\}_{t \in [T]}$ that solves eq. (8);
- $\kappa_f \triangleq 1 - \min_{v \in \mathcal{V}} [f(\mathcal{V}) - f(\mathcal{V} \setminus \{v\})]/f(v)$ is the curvature of f [27]. κ_f measures how far f is from modularity: if $\kappa_f = 0$, then $f(\mathcal{V}) - f(\mathcal{V} \setminus \{v\}) = f(v)$, $\forall v \in \mathcal{V}$, *i.e.*, f is modular. In contrast, $\kappa_f = 1$ in the extreme case where there exist $v \in \mathcal{V}$ such that $f(\mathcal{V}) = f(\mathcal{V} \setminus \{v\})$, *i.e.*, v has no contribution in the presence of $\mathcal{V} \setminus \{v\}$.

The intuition is that the agents should *select actions simultaneously* such that each agent i selects an action $a_{i,t}$ that maximizes the marginal gain $f(\, a \,|\, \{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}\,)$. But since the agents select actions simultaneously, $\{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}$ become known only after agent i selects $a_{i,t}$ and communicates with $\mathcal{N}_{i,t}$, *i.e.*, computing $f(\, a \,|\, \{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}\,)$ becomes feasible for all $a \in \mathcal{V}_i$ only in hindsight. To this end, ActionCoordination instructs each agent i to select actions $\{a_{i,t}\}_{t \in [T]}$ such that the action regret

$$\max_{a \in \mathcal{V}_i} \sum_{t=1}^{T} f(\, a \,|\, \{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}\,) - \sum_{t=1}^{T} f(\, a_{i,t} \,|\, \{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}\,), \quad (5)$$

is sublinear in T. Thus, the action coordination problem takes the form of Problem 2 with full-information feedback, where

the reward of each action $a \in \mathcal{V}_i$ is the marginal gain, *i.e.*, $r_{a,t} \triangleq f(a \,|\, \{a_{j,t}\}_{j \in \mathcal{N}_{i,t}})$.

ActionCoordination implements a Multiplicative Weights Update (MWU) procedure to converge to an optimal solution to eq. (5) —the MWU procedure has been introduced to solve Problem 2 with full-information feedback [28]. In more detail, ActionCoordination starts by initializing a learning rate η_1 and a weight vector w_t for all available actions $a \in \mathcal{V}_i$ (Algorithm 2's lines 1-2). Then, at each time step $t \in [T]$, ActionCoordination executes in sequence the steps:

- Compute probability distribution p_t using w_t (lines 3-4);
- Select action $a_{i,t} \in \mathcal{V}_i$ by sampling from p_t (line 5);
- Send $a_{i,t}$ to NeighborSelection and receive neighbors' actions $\{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}$ (line 6);
- Compute marginal gain of selecting $a \in \mathcal{V}_i$ as reward $r_{a,t}$ associated with each $a \in \mathcal{V}_i$, and update weight $w_{a,t+1}$ for each $a \in \mathcal{V}_i$ (lines 7-9).[1]

Proposition 1 (Approximation Performance)**.** *The agents select actions via* ActionCoordination *such that:*

$$\sum_{t=1}^{T} f(\mathcal{A}_t) \geq \frac{1-\kappa_f}{1+\kappa_f-\kappa_f^2} \sum_{t=1}^{T} \Bigg[f(\mathcal{A}^{\mathsf{OPT}}) + \sum_{i \in \mathcal{N}} \underbrace{[f(a_{i,t}) - f(a_{i,t} \,|\, \{a_{j,t}\}_{j \in \mathcal{N}_{i,t}})]}_{I_{f,t}(a_{i,t}; \mathcal{N}_{i,t})} \Bigg] - \underbrace{\tilde{O}\left(|\mathcal{N}|\sqrt{T}\right)}_{\textit{sublinear regret}}. \tag{6}$$

where $\tilde{O}(\cdot)$ *hides* log *terms.*

Proposition 1 implies that the approximation performance of ActionCoordination increases for network designs $\{\mathcal{N}_{i,t}\}_{i \in \mathcal{N}}$ with higher value $I_{f,t}(a_{i,t}; \mathcal{N}_{i,t})$. Particularly, the $-\sum_{i \in \mathcal{N}} I_{f,t}(a_{i,t}; \mathcal{N}_{i,t})$ plays the role of $C(\{\mathcal{N}_i\}_{i \in \mathcal{N}})$ in Introduction. Intuitively, $I_{f,t}(a_{i,t}; \mathcal{N}_{i,t})$ captures the utility overlap between agent i's action and the actions of its neighbors: for example, when the network is fully disconnected ($\mathcal{N}_{i,t} = \emptyset$, $\forall i \in \mathcal{N}$), then $I_{f,t}(a_{i,t}; \mathcal{N}_{i,t}) = 0$.

Definition 3 (Mutual Information between an Agent and Its Neighbors)**.** *At any $t \in [T]$, given an agent $i \in \mathcal{N}$ with an action $a_{i,t}$ and neighbors $\mathcal{N}_{i,t}$, the mutual information between the agent and its neighbors is denoted by*[2]

$$I_{f,t}(a_{i,t}; \mathcal{N}_{i,t}) \triangleq f(a_{i,t}) - f(a_{i,t} \,|\, \{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}). \tag{7}$$

NeighborSelection will next leverage Lemma 1 to enable the agents to distributively select a network topology that optimizes the approximation bound of ActionCoordination.

Lemma 1 (Monotonicity and Submodularity of $I_{f,t}$)**.** *Given an $a \in \mathcal{V}_i$ and a non-decreasing and 2nd-order submodular function $f: 2^{\mathcal{V}_{\mathcal{N}}} \mapsto \mathbb{R}$, then $I_{f,t}(a; \cdot)$ is non-decreasing and submodular in the second argument.*

C. Neighbor Selection

[1]The coordination algorithms in [12]–[14] instruct the agents to select actions simultaneously at each time step as ActionCoordination, but they lift the coordination problem to the continuous domain and require each agent to know/estimate the gradient of the multilinear extension of f, which leads to a decision time at least one order higher than ActionCoordination [11].

[2]The quantity in eq. (7) extends the definition of *submodular mutual information* [29] to the multi-agent setting introduced herein.

Algorithm 3: NeighborSelection

Input: Number of time steps T; agent i's neighbor candidate set $\mathcal{M}_i \subseteq \mathcal{N} \setminus \{i\}$; agent i's neighborhood size α_i; objective set function $f: 2^{\mathcal{V}_{\mathcal{N}}} \mapsto \mathbb{R}$.

Output: Agent i's neighbors $\mathcal{N}_{i,t}$ at each $t \in [T]$.

1: $\eta_2 \leftarrow \sqrt{2 \log |\mathcal{M}_i| \,/\, (|\mathcal{M}_i| T)}$; $\gamma = \eta_2/2$;
2: $z_1^{(k)} \leftarrow \left[z_{1,1}^{(k)}, \ldots, z_{\alpha_i,1}^{(k)}\right]^\top$ with $z_{j,1}^{(k)} = 1$, $\forall v \in \mathcal{M}_i, \forall k \in [\alpha_i]$;
3: **for** each time step $t \in [T]$ **do**
4: **receive** action $a_{i,t}$ **from** ActionCoordination;
5: **for** $k = 1, \ldots, \alpha_i$ **do**
6: **get** distribution $q_t^{(k)} \leftarrow z_t^{(k)} \,/\, \|z_t^{(k)}\|_1$;
7: **draw** agent $j_t^{(k)} \in \mathcal{M}_i$ **from** $q_t^{(k)}$;
8: **receive** action $a_{j_t^{(k)},t}$ **from** $j_t^{(k)}$;
9: $r_{j_t^{(k)},t} \leftarrow I_{f,t}(a_{i,t}; \{a_{j_t^{(1)},t}, \ldots, a_{j_t^{(k)},t}\}) - I_{f,t}(a_{i,t}; \{a_{j_t^{(1)},t}, \ldots, a_{j_t^{(k-1)},t}\})$ and **normalize** $r_{j_t^{(k)},t}$ **to** $[0,1]$;
10: $\tilde{r}_{j,t}^{(k)} \leftarrow 1 - \frac{\mathbf{1}(j_t^{(k)}=j)}{q_{j,t}^{(k)}+\gamma}\left(1 - r_{j_t^{(k)},t}\right)$, $\forall j \in \mathcal{M}_i$;
11: $z_{j,t+1}^{(k)} \leftarrow z_{j,t}^{(k)} \exp\left(\eta_2 \tilde{r}_{j,t}^{(k)}\right)$, $\forall j \in \mathcal{M}_i$;
12: **end for**
13: $\mathcal{N}_{i,t} \leftarrow \{j_{k,t}\}_{k \in [\alpha_i]}$;
14: **end for**

We present NeighborSelection. To this end, we introduce the neighbor-selection problem that NeighborSelection instructs the agents to simultaneously solve and show that it takes the form of Problem 2 with bandit feedback.

Since ActionCoordination's suboptimality bound improves when $I_{f,t}(a_{i,t}; \mathcal{N}_{i,t})$ increases, NeighborSelection instructs each agent i to select its neighbors by solving the cardinality-constrained maximization problem:

$$\max_{\mathcal{N}_{i,t} \subseteq \mathcal{M}_i, |\mathcal{N}_{i,t}| \leq \alpha_i} \sum_{t=1}^{T} I_{f,t}(a_{i,t}; \mathcal{N}_{i,t}), \tag{8}$$

where $a_{i,t}$ is given by ActionCoordination (Fig. 1). The problem in eq. (8) is a submodular optimization problem since we prove that $I_{f,t}(a_{i,t}; \mathcal{N}_i)$ is submodular in $\mathcal{N}_i$.

But $I_{f,t}(a_{i,t}; \mathcal{N}_{i,t})$ is computable in hindsight only: the $\{a_{j,t}\}_{j \in \mathcal{N}_{i,t}}$ become known only after agent i has selected and communicated with $\mathcal{N}_{i,t}$. Therefore, eq. (8) takes the form of cardinality-constrained bandit submodular maximization [1], [30], [31], which is an extension of Problem 2 to the submodular multi-agent setting.

Solving eq. (8) using algorithms for Problem 2 with bandit feedback will lead to exponential-running-time algorithms due to an exponentially large $\mathcal{V}$ per eq. (4) [31]. Therefore, NeighborSelection instead extends [31, Algorithm 2], which can solve eq. (8) in the full-information setting, to the bandit setting [1]. Specifically, NeighborSelection decomposes eq. (8) to α_i instances of Problem 2 with bandit feedback, and separately solves each of them using the EXP3-IX algorithm [32], which can handle bandit feedback.

NeighborSelection starts by initializing a learning rate η_2 and α_i weight vectors $z_t^{(k)}, \forall k \in [\alpha_i]$, each determining the

k-th selection in $\mathcal{N}_{i,t}$ (Algorithm 3's lines 1-2). Then, at each $t \in [T]$, NeighborSelection executes the steps:

- Receive action $a_{i,t}$ by ActionCoordination (lines 3-4);
- Compute distribution $q_t^{(k)}$ using $z_t^{(k)}, \forall k \in [\alpha_i]$ (lines 5-6);
- Select agent $j_t^{(k)} \in \mathcal{M}_i$ as neighbor by sampling from $q_t^{(k)}$, and receive its action $a_{j_t^{(k)},t}, \forall k \in [\alpha_i]$ (lines 7-8);
- For each $k \in [\alpha_i]$, compute reward $r_{j_t^{(k)},t}$ associated with each $j_t^{(k)}$, estimate reward $\tilde{r}_{j,t}^{(k)}$ for each $j \in \mathcal{M}_i$, and update weight $z_{j,t+1}^{(k)}$ for each $j \in \mathcal{M}_i$ (lines 9-12).

IV. Approximation Guarantee

We present the suboptimality bound of Anaconda. Thus, *the bound compares* Anaconda *with an optimal fully centralized algorithm that maximizes f per eq. (1).*

Theorem 1 (Approximation Performance)**.** Anaconda *instructs over $t \in [T]$ each agent $i \in \mathcal{N}$ to select actions $\{a_{i,t}\}_{t\in[T]}$ and neighborhoods $\{\mathcal{N}_{i,t}\}_{t\in[T]}$ that guarantee:*

- *If the network is* ***fully centralized****,* i.e.*, $\mathcal{N}_{i,t} \equiv \mathcal{N} \setminus \{i\}$,*

$$\mathbb{E}\left[f(\mathcal{A}_t)\right] \geq \frac{1}{1+\kappa_f} f(\mathcal{A}^{\mathsf{OPT}}) - \underbrace{\tilde{O}\left(|\mathcal{N}|\sqrt{1/T}\right)}_{\phi(T)}. \quad (9)$$

- *If the network is* ***fully decentralized****,* i.e.*, $\mathcal{N}_{i,t} \equiv \emptyset$,*

$$\mathbb{E}\left[f(\mathcal{A}_t)\right] \geq \frac{1-\kappa_f}{1+\kappa_f-\kappa_f^2} f(\mathcal{A}^{\mathsf{OPT}}) - \underbrace{\tilde{O}\left(|\mathcal{N}|\sqrt{1/T}\right)}_{\chi(T)}. \quad (10)$$

- *If the network is* ***anything in between*** *fully centralized and fully decentralized,* i.e.*, $\mathcal{N}_{i,t} \subseteq \mathcal{M}_i \subseteq \mathcal{N} \setminus \{i\}$,*

$$\mathbb{E}\left[f(\mathcal{A}_t)\right] \geq \frac{1-\kappa_f}{1+\kappa_f-\kappa_f^2} f(\mathcal{A}^{\mathsf{OPT}}) + \frac{1-\kappa_f}{1+\kappa_f-\kappa_f^2}\frac{1}{\kappa_f}(1-e^{-\kappa_f})\,\mathbb{E}\Bigg[\underbrace{\sum_{i\in\mathcal{N}} I_{f,t}(a_{i,t};\mathcal{N}_i^\star)}_{I^\star}\Bigg]$$
$$\underbrace{-\tilde{O}\left(\bar{\alpha}|\mathcal{N}|\sqrt{|\bar{\mathcal{M}}|/T}\right) - \log(2/\delta)\tilde{O}\left(\bar{\alpha}|\mathcal{N}|\sqrt{|\bar{\mathcal{M}}|/T}\right)}_{\psi(T)}, \quad (11)$$

where $\bar{\alpha} \triangleq \max_{i\in\mathcal{N}} \alpha_i$, and $|\bar{\mathcal{M}}| \triangleq \max_{i\in\mathcal{N}} |\bar{\mathcal{M}}_i|$.

Particularly, each case above holds with probability at least $1-\delta$, for any $\delta \in (0,1)$, and the expectation is due to Anaconda*'s internal randomness. $\tilde{O}(\cdot)$ hides* log *terms.*

Theorem 1 quantifies both the suboptimality of Anaconda due to decentralization, and the convergence speed of Anaconda. Both are affected as follows:

- *Effect of online co-design of network topology and agent actions*: ψ in eq. (11) captures the convergence speed of the network and action selection and its impact to the suboptimality bound —similarly in eqs. (9) and (10) for ϕ and χ. Particularly, ψ vanishes as $T \to \infty$, having no impact on the suboptimality bound anymore, and its vanishing speed captures how fast the agents converge to near-optimal actions and neighborhoods.
- *Effect of resource-minimal distributed communication and action coordination*: After ψ vanishes as $T \to \infty$, the bound in eq. (11) depends on $I^\star$ that captures the suboptimality due to decentralization such that the higher $I^\star$ is the better is Anaconda's approximation performance. Particularly, $I^\star$ depends on the neighborhoods of each agent i, and the larger agent i's neighborhood can be, the higher $I^\star$ can be since $I_{f,t}(a_{i,t};\mathcal{N}_i)$ is non-decreasing in $\mathcal{N}_i$. Then, the better Anaconda's suboptimality bound can be per eq. (11). In contrast, if $\alpha_i = 0$ for all agents $i \in \mathcal{N}$, then $I_{f,t}(a_{i,t};\emptyset) = 0$, and as $T \to \infty$, eq. (10) and eq. (11) become the same.

Overall, eqs. (9) to (11) imply that Anaconda's global suboptimality bound will improve if the agents can communicate and coordinate over a more centralized network, with the bound covering the range $[(1-\kappa_f)/(1+\kappa_f-\kappa_f^2), 1/(1+\kappa_f)]$ as the network covers the spectrum from being fully decentralized (eq. (10)) to fully centralized (eq. (9)). Importantly, the $1/(1+\kappa_f)$ suboptimality bound in the fully centralized case recovers the bound in [27] and is near-optimal since the best possible bound for in (3) is $1-\kappa_f/e$ [33].[3]

In all, asymptotically (as $T \to \infty$), Anaconda *enables the agents to individually select near-optimal actions and communication neighborhoods.*

Remark 2 (On the Tightness of Approximation Bounds)**.** *The approximation bound in Theorem 1 are not necessarily tight. Particularly, the bound in eq. (11) does not converge to $1/(1+\kappa_f)$ when the network becomes fully centralized. As part of our future work, we will investigate tight bounds.*

V. Runtime Analysis

We present the runtime of Anaconda by analyzing its computation and communication complexity (accounting for message length). We use the notation and observations:

- τ_f is the time required for one evaluation of f;
- τ_c is the time for transmitting the information about one action from an agent directly to another agent;
- ϵ is Anaconda's convergence error after T iterations: if the network is fully centralized or fully decentralized, then Anaconda's convergence error after T iterations is ϵ only if, per eqs. (9) and (10), $\phi(T) \leq \epsilon$ or $\chi(T) \leq \epsilon$, *i.e.*, $T \geq |\mathcal{N}|^2/\epsilon$. Similarly, if the network is anything in between, per eq. (11), only if $\psi(T) \leq \epsilon$, *i.e.*, $T \geq |\mathcal{M}_i|\,|\mathcal{N}|^2/\epsilon$.

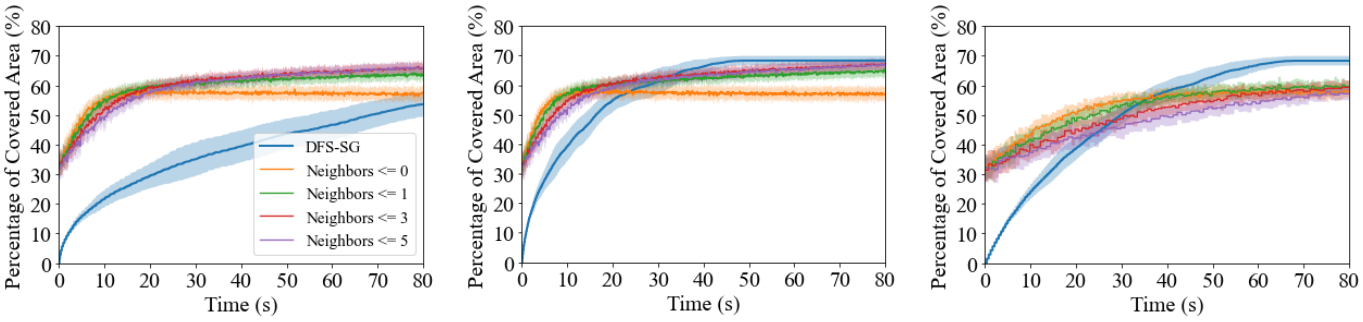
Proposition 2 (Computational Complexity)**.** *At each $t \in [T]$,* Anaconda *requires each agent i to execute $|\mathcal{V}_i|+2\alpha_i+1$ evaluations of f and $O(|\mathcal{V}_i|+\alpha_i|\mathcal{M}_i|)$ additions/multiplications.*

Proposition 3 (Communication Complexity)**.** *At each $t \in [T]$,* Anaconda *requires one communication round where each agent i only transmits its own action to other agents.*

Theorem 2 (Decision Speed)**.** Anaconda *terminates in $O\left\{\left[\tau_f \max_{i\in\mathcal{N}}(|\mathcal{V}_i|+\alpha_i) + \tau_c\right](\max_{i\in\mathcal{N}}|\mathcal{M}_i|)\,|\mathcal{N}|^2/\epsilon\right\}$.*

Corollary 1 (Decision Speed for Sparse Networks)**.** *In sparse networks, where $|\mathcal{M}_i| = o(|\mathcal{N}|)$,* Anaconda *terminates in $O\left\{\left[\tau_f \max_{i\in\mathcal{N}}(|\mathcal{V}_i|+\alpha_i) + \tau_c\right]|\mathcal{N}|^2/\epsilon\right\}$ time.*

[3]The bounds $1/(1+\kappa_f)$ and $1-\kappa_f/e$ become $1/2$ and $1-1/e$ when, in the worst case, $\kappa_f = 1$.



(a) $\tau_f = 0.01s, \tau_c = 0.05s$. (b) $\tau_f = 0.01s, \tau_c = 0.01s$. (c) $\tau_f = 0.05s, \tau_c = 0.01s$.

Fig. 2: **Area Monitoring with Multiple Cameras:** Anaconda **vs.** DFS-SG. The cameras select the locations of their FOVs either per Anaconda with different maximum neighborhood sizes ranging among $\{0, 1, 3, 5\}$, or per the DFS-SG. (a)-(c) are averaged over 30 Monte-Carlo trials. From (a) to (b) to (c), the time τ_c of communicating an action decreases compared to the time τ_f of completing a function evaluation, with $\tau_c/\tau_f = \{5, 1, 0.2\}$.

Remark 3 (Anaconda vs. Sequential Greedy [15])**.** *In sparse networks,* Anaconda *can be one order faster than the Sequential Greedy algorithm, which requires* $O(\tau_f \sum_{i\in\mathcal{N}} |\mathcal{V}_i| + \tau_c |\mathcal{N}|^3)$ *time to terminate for directed networks. The decision time results from each agent* i *in the sequence needing to first evaluate* f *for* $|\mathcal{V}_i|$ *times and then transmit all actions selected so far by agents* $[i]$ *to agent* $(i+1)$*. The proof appears in the full version [22], following the steps of [19, Proposition 2] accounting for the size of each inter-agent communication message. In contrast,* Anaconda *requires* $O\left\{[\tau_f \max_{i\in\mathcal{N}}(|\mathcal{V}_i| + \alpha_i) + \tau_c]\,|\mathcal{N}|^2/\epsilon\right\}$ *time to terminate. For example, if* $\tau_f = \tau_c = \tau$ *and* $|\mathcal{V}_i| = v, \forall i \in \mathcal{N}$*, with* τ *and* v *being constant, then Sequential Greedy takes* $O(|\mathcal{N}|^3)$ *time while* Anaconda *takes* $O(|\mathcal{N}|^2/\epsilon)$ *time.*

VI. Numerical Evaluation in Sensor Scheduling for Area Monitoring

We evaluate Anaconda in a simulated scenario of 2D area monitoring. The results are summarized in Fig. 2. We next elaborate on the simulated area monitoring setup, the compared algorithms, and the used performance metrics.

Area Monitoring Setup. The setup is as follows:

- *Environment:* The environment is a static 100×100 map.
- *Agents:* There exist 60 downward-facing cameras. Each camera $i \in \mathcal{N}$ is located at $x_i \in [0, 100]^2$ and can point its limited field of view (FOV) to different directions.
- *Actions:* Each camera i has a circular FOV of radius $r = 7$ and can point it to 8 cardinal directions. Particularly, at each time step t, camera i can locate the center of its FOV at $a_{i,t} \in \mathcal{V}_i$ where $\mathcal{V}_i \triangleq x_i + [r\cos\theta_t, r\sin\theta_t]$ and θ_t is one of the 8 cardinal directions. Each camera i is unaware of $\mathcal{V}_j, j \in \mathcal{N}\backslash\{i\}$. Thus, the cameras have to communicate to know about one another's action information.
- *Communication Network:* The emergent communication network $\mathcal{G}_t$ can be directed and time-varying. At each time step t, each camera i first finds its neighbor candidate set $\mathcal{M}_i \triangleq \{j\}_{\|x_j - x_i\| \le c_i, j\in\mathcal{N}\backslash\{i\}}$, where c_i is i's communication range. Then, it uses NeighborSelection to select neighborhood $\mathcal{N}_{i,t}$ from $\mathcal{M}_i$. Once $\mathcal{N}_{i,t}$ is determined by all cameras $i \in \mathcal{N}$, $\mathcal{G}_t$ is defined.
- *Objective Function:* $f(\{a_{i,t}\}_{i\in\mathcal{N}})$ is the total area covered by the cameras $\mathcal{N}$ when they select $\{a_{i,t}\}_{i\in\mathcal{N}}$ as the centers of their FOVs. f is proved to be submodular [3].

Compared Algorithms. We evaluate Anaconda against the state-of-the-art algorithm DFS-SG [19] across 30 Monte-Carlo scenarios where at each trial (i) each camera location x_i is uniformly sampled from $[0, 100]^2$, $\forall i \in \mathcal{N}$, and (ii) the communication ranges c_i, $\forall i \in \mathcal{N}$ are uniformly sampled from $[15, 20]$. We repeat the 30 trials over three sets of possible values of τ_f and τ_c, $(\tau_f, \tau_c) = (0.01s, 0.05s), (0.01s, 0.01s)$, or $(0.05s, 0.01s)$, that capture scenarios with different computational and communication loads.

In more detail, we evaluate the following algorithms:

a) Anaconda *with different neighborhood sizes:* To evaluate the impact of decentralization on the trade-off between decision speed and total area coverage, at each Monte-Carlo trial, we ran multiple Anaconda varying the maximum neighborhood size $\alpha_i \le n_{max}$, where $n_{max} = \{0, 1, 3, 5\}$.

b) DFS-SG [19]*:* We also compare Anaconda with the state-of-the-art algorithm DFS-SG. DFS-SG requires a given connected communication network to run. To this end, at each trial, we first sample the same communication ranges c_i from $[15, 20]$ and construct $\mathcal{M}_i, \forall i \in \mathcal{N}$ for both Anaconda and DFS-SG. Then, while Anaconda actively selects neighbors $\mathcal{N}_{i,t} \subseteq \mathcal{M}_i, \forall i \in \mathcal{N}$, to enable scalability, DFS-SG will directly take $\mathcal{N}_{i,t} \equiv \mathcal{M}_i, \forall t$. We set the range c_i to be not too small to ensure the communication network for DFS-SG is always connected.

Performance Metrics. We evaluate the performance of the algorithms in terms of their (i) decision speed —time to convergence— and (ii) achieved objective value.

System Specifications. We ran all simulations using Python 3.11.7 on a Windows PC with the Intel Core i9-14900KF CPU @ 3.20 GHz and 64 GB RAM.

Code. The code is available at https://github.com/UM-iRaL/Self-configurable-network.

Results. The simulation results are presented in Fig. 2. From Fig. 2, we observe the following:

- *Trade-off of centralization and decentralization:* As the neighborhood size n_{max} increases, *i.e.*, the coordination becomes more centralized, the convergence speed of Anaconda decreases, while the total covered area upon convergence increases. These observations agree with the proven theory: (i) per Proposition 2, NeighborSelection runs faster when each agent can select fewer neighbors; and (ii) per Theorem 1, as the cameras' neighborhood sizes increase,

the approximation performance of Anaconda increases.

- Anaconda *vs.* DFS-SG*:* Upon convergence, both algorithms cover a comparable total area. As expected, Anaconda converges faster when the communication cost to the decision speed is no less than the computational cost ($\tau_c \geq \tau_f$). In more detail, we observe the following:
  - *Total covered area*: Anaconda starts with a non-zero covered area (30% of the total area), whereas DFS-SG starts from near-zero covered area. The reason is that Anaconda instructs all cameras to execute an action from the start of time, whereas DFS-SG instructs the cameras to execute actions sequentially. Upon convergence, the two algorithms cover a comparable total area.
  - *Convergence speed*: When the communication cost to the decision speed is no less than the computation cost ($\tau_c \geq \tau_f$), then Anaconda converges faster. For example, for $\tau_f = 0.01s$ and $\tau_c = 0.05s$ (Fig. 2(a)), Anaconda converges within 10s to 20s, whereas DFS-SG achieves comparable performance at 80s. Anaconda converges slower when $\tau_f = 0.05s$ and $\tau_c = 0.01s$ (Fig. 2(c)) since Anaconda requires more computations per step. But still, Anaconda covers a comparable total area to DFS-SG across all time steps in Fig. 2(c).

VII. Conclusion

We introduced a rigorous approach, Anaconda, that enables multi-agent networks to self-configure their communication topology to balance the trade-off between decision speed and approximation performance during multi-agent coordination. Compared to the state of the art, Anaconda is an anytime self-configuration algorithm that quantifies its suboptimality guarantee for any type of network, from fully disconnected to fully centralized, and that, for sparse networks, is one order faster. We demonstrated Anaconda in simulated scenarios of area monitoring with multiple cameras.

Future work. We will extend this work to reduce the number of function evaluations needed by Anaconda, and investigate tighter suboptimality bounds.

References

[1] Z. Xu, X. Lin, and V. Tzoumas, "Bandit submodular maximization for multi-robot coordination in unpredictable and partially observable environments," in *Robotics: Science and Systems (RSS)*, 2023.

[2] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *IEEE Inter. Conf. Rob. Auto. (ICRA)*, 2015, pp. 4775–4782.

[3] M. Corah and N. Michael, "Distributed submodular maximization on partition matroids for planning on large sensor networks," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6792–6799.

[4] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Jour. Mach. Learn. Res. (JMLR)*, vol. 9, pp. 235–284, 2008.

[5] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research (JAIR)*, vol. 34, pp. 707–755, 2009.

[6] P. Tokekar, V. Isler, and A. Franchi, "Multi-target visual tracking with aerial robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 3067–3072.

[7] B. Gharesifard and S. L. Smith, "Distributed submodular maximization with limited information," *IEEE Transactions on Control of Network Systems (TCNS)*, vol. 5, no. 4, pp. 1635–1645, 2017.

[8] J. R. Marden, "The role of information in distributed resource allocation," *IEEE Transactions on Control of Network Systems (TCNS)*, vol. 4, no. 3, pp. 654–664, 2017.

[9] D. Grimsman, M. S. Ali, J. P. Hespanha, and J. R. Marden, "The impact of information in distributed submodular maximization," *IEEE Trans. Ctrl. Netw. Sys. (TCNS)*, vol. 6, no. 4, pp. 1334–1343, 2019.

[10] B. Schlotfeldt, V. Tzoumas, and G. J. Pappas, "Resilient active information acquisition with teams of robots," *IEEE Transactions on Robotics (TRO)*, vol. 38, no. 1, pp. 244–261, 2021.

[11] Z. Xu and V. Tzoumas, "Resource-aware distributed submodular maximization: A paradigm for multi-robot decision-making," in *IEEE Conference on Decision and Control (CDC)*, 2022, pp. 5959–5966.

[12] B. Du, K. Qian, C. Claudel, and D. Sun, "Jacobi-style iteration for distributed submodular maximization," *IEEE Transactions on Automatic Control (TAC)*, vol. 67, no. 9, pp. 4687–4702, 2022.

[13] N. Rezazadeh and S. S. Kia, "Distributed strategy selection: A submodular set function maximization approach," *Automatica*, vol. 153, p. 111000, 2023.

[14] A. Robey, A. Adibi, B. Schlotfeldt, H. Hassani, and G. J. Pappas, "Optimal algorithms for submodular maximization with distributed constraints," in *Learn. for Dyn. & Cont. (L4DC)*, 2021, pp. 150–162.

[15] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions–II," in *Polyhedral combinatorics*, 1978, pp. 73–87.

[16] U. Feige, "A threshold of $ln(n)$ for approximating set cover," *Journal of the ACM (JACM)*, vol. 45, no. 4, pp. 634–652, 1998.

[17] Q. Wu, J. Xu, Y. Zeng, D. W. K. Ng, N. Al-Dhahir, R. Schober, and A. L. Swindlehurst, "A comprehensive overview on 5G-and-beyond networks with UAVs: From communications to sensing and intelligence," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 2912–2945, 2021.

[18] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[19] R. Konda, D. Grimsman, and J. R. Marden, "Execution order matters in greedy algorithms with limited information," in *American Control Conference (ACC)*, 2022, pp. 1305–1310.

[20] Y.-C. Liu, J. Tian, C.-Y. Ma, N. Glaser, C.-W. Kuo, and Z. Kira, "Who2com: Collaborative perception via learnable handshake communication," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6876–6883.

[21] Y. Niu, R. Paleja, and M. Gombolay, "Multi-agent graph-attention communication and teaming," in *International Conference on Autonomous Agents and MultiAgent Systems*, 2021, pp. 964–973.

[22] Z. Xu and V. Tzoumas, "Performance-aware self-configurable multi-agent networks: A distributed submodular approach for simultaneous coordination and network design," *arXiv preprint:2409.01411*, 2024.

[23] Y. Crama, P. L. Hammer, and R. Holzman, "A characterization of a cone of pseudo-boolean functions via supermodularity-type inequalities," in *Quantitative Methoden in den Wirtschaftswissenschaften*. Springer, 1989, pp. 53–55.

[24] S. Foldes and P. L. Hammer, "Submodularity, supermodularity, and higher-order monotonicities of pseudo-boolean functions," *Mathematics of Operations Research*, vol. 30, no. 2, pp. 453–461, 2005.

[25] O. S. Oubbati, M. Atiquzzaman, P. Lorenz, M. H. Tareque, and M. S. Hossain, "Routing in flying ad hoc networks: Survey, constraints, and future challenge perspectives," *IEEE Access*, pp. 81 057–81 105, 2019.

[26] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge University Press, 2020.

[27] M. Conforti and G. Cornuéjols, "Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem," *Discrete Applied Mathematics*, vol. 7, no. 3, pp. 251–274, 1984.

[28] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[29] R. Iyer, N. Khargonkar, J. Bilmes, and H. Asnani, "Generalized submodular information measures: Theoretical properties, examples, optimization algorithms, and applications," *IEEE Transactions on Information Theory*, 2021.

[30] M. Zhang, L. Chen, H. Hassani, and A. Karbasi, "Online continuous submodular maximization: From full-information to bandit feedback," *Adv. Neu. Info. Proc. Sys. (NeurIPS)*, vol. 32, 2019.

[31] T. Matsuoka, S. Ito, and N. Ohsaka, "Tracking regret bounds for online submodular optimization," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2021, pp. 3421–3429.

[32] G. Neu, "Explore no more: Improved high-probability regret bounds for non-stochastic bandits," *Adv. Neu. Info. Proc. Sys.*, vol. 28, 2015.

[33] M. Sviridenko, J. Vondrák, and J. Ward, "Optimal approximation for submodular and supermodular optimization with bounded curvature," *Math. of Operations Research*, vol. 42, no. 4, pp. 1197–1218, 2017.