Non-Monotone Energy-Aware Information Gathering for Heterogeneous Robot Teams

Xiaoyi Cai[1,*], Brent Schlotfeldt[2,*], Kasra Khosoussi[1], Nikolay Atanasov[3], George J. Pappas[2], and Jonathan P. How[1]

***Abstract*— This paper considers the problem of planning trajectories for a team of sensor-equipped robots to reduce uncertainty about a dynamical process. Optimizing the trade-off between information gain and energy cost (e.g., control effort, distance travelled) is desirable but leads to a non-monotone objective function in the set of robot trajectories. Therefore, common multi-robot planning algorithms based on techniques such as coordinate descent lose their performance guarantees. Methods based on local search provide performance guarantees for optimizing a non-monotone submodular function, but require access to all robots' trajectories, making it not suitable for distributed execution. This work proposes a distributed planning approach based on local search and shows how lazy/greedy methods can be adopted to reduce the computation and communication of the approach. We demonstrate the efficacy of the proposed method by coordinating robot teams composed of both ground and aerial vehicles with different sensing/control profiles and evaluate the algorithm's performance in two target tracking scenarios. Compared to the naive distributed execution of local search, our approach saves up to 60% communication and 80–92% computation on average when coordinating up to 10 robots, while outperforming the coordinate descent based algorithm in achieving a desirable trade-off between sensing and energy cost.**

SUPPLEMENTARY MATERIAL

`https://www.youtube.com/watch?v=xWgFi6fwex0`

I. INTRODUCTION

Developments in sensing and mobility have enabled effective utilization of robot systems in autonomous mapping [1]–[4], search and rescue [5]–[7], and environmental monitoring [8]–[11]. These tasks require spatiotemporal information collection which can be achieved more efficiently and accurately by larger robot teams, rather than relying on individual robots. Robot teams may take advantage of heterogeneous capabilities, require less storage and computation per robot, and may achieve better environment coverage in shorter time [12]–[15]. Task-level performance is usually quantified by a measure of information gain, where typically the marginal improvements diminish given additional measurements (*submodularity*), and adding new measurements does not reduce the objective (*monotonicity*). Although planning optimally for multi-robot sensing trajectories is generally intractable, these two properties allow for *near-optimal* approximation algorithms that scale to large robot teams, while providing worst-case guarantees. Additionally, practical implementations often need to consider various measures for energy expenditure, such as control effort or distance travelled. A common approach is to impose fixed budgets, which preserves submodularity and monotonicity of the objective, so that existing algorithms may still be used [16]–[18].

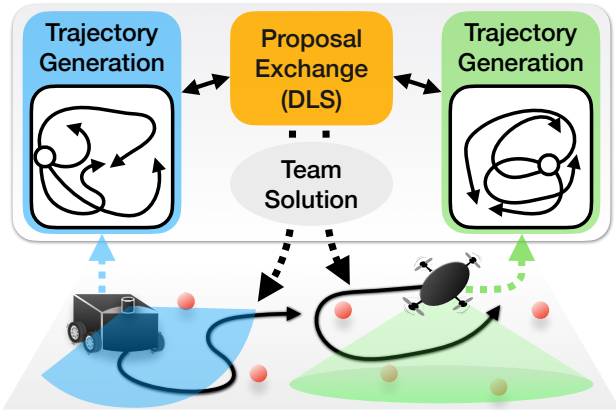

Fig. 1. Overview of the proposed distributed planning approach for non-monotone information gathering (see Sec. IV). Robots generate individual candidate trajectories and jointly build a team plan via distributed local search (`DLS`), by repeatedly proposing changes to the collective trajectories.

In this paper, we are motivated by scenarios where robots, with potentially different sensing and control capabilities, seek a desired trade-off between sensing and energy cost. Specifically, we formulate an *energy-aware active information acquisition* problem, where the goal is to plan trajectories for a team of heterogeneous robots to maximize a weighted sum of information gain and energy cost. One key observation is that adding the energy cost breaks the *monotonicity* of the objective, violating an assumption held by existing approximation algorithms. Thus, we propose a new distributed planning algorithm based on local search [19] (see Fig. 1) that has a worst-case guarantee for the non-monotone objective. We also show how to reduce the method's computation and communication to improve scalability.

Related Work. Our work belongs to the category of multi-robot informative path planning, where robots plan sensing trajectories to reduce uncertainty about a dynamic process (e.g., [2], [4], [16], [18], [20]–[25]). To alleviate the computational complexity, which is exponential in the number of robots, approximation methods have been developed to pro-

*indicates equal contribution.

[1]X. Cai, K. Khosoussi, and J. P. How are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. {`xyc, kasra, jhow`}`@mit.edu`.

[2]B. Schlotfeldt, and G. J. Pappas are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA. {`brentsc, pappasg`}`@seas.upenn.edu`.

[3]N. Atanasov is with the Electrical and Computer Engineering Department, University of California, San Diego, La Jolla, CA 92093, USA. `natanasov@ucsd.edu`.

duce near-optimal solutions for a submodular and monotone objective (e.g., mutual information). A common technique is coordinate descent, where robots plan successively while incorporating the plans of previous robots. Ref. [16] showed that coordinate descent extends the near-optimality of a single-robot planner to the multi-robot scenario. This result was extend to dynamic targets by [26], achieving at least 50% of the optimal performance regardless of the planning order. Refs. [18], [22] decentralized the greedy method [27] by adding the best single-robot trajectory to the team solution in every round. Ref. [4] proposed distributed sequential greedy algorithm to alleviate the inefficiency in sequential planning.

Our problem can be seen as non-monotone submodular maximization subject to a partition matroid constraint (see Sec. III), for which approximation algorithms already exist. The first such algorithm was developed by [19] based on local search, which can handle multiple matroid constraints. Extending [19], ref. [28] proposed a greedy-based approach that can handle multiple independence systems (more general than matroids), but has a worse approximation ratio given a single matroid. Other methods use multilinear relaxation such as [29], [30] for better approximation ratios, but require significant computation. Applying some of these ideas in robotics, ref. [31] used the continuous greedy method by [29] for decentralized multi-robot task assignment. In the same domain, ref. [32] combined sampling, greedy method, and lazy evaluation [33] to achieve fast computation. We decided to build upon [19] for its simplicity and guarantees. We also attempt to incorporate well-known techniques like greedy method and lazy evaluation, but they are specialized in the context of local search, as detailed in Sec. IV-B.

Contributions. The main limitation of the prior works is the assumption of monotonicity of the objective function. Problems without monotonicity, such as the energy-aware problem we propose, cannot be solved by the above methods while retaining their near-optimality properties. In contrast, our proposed algorithm provides a theoretical performance guarantee even for non-monotone objectives. In this work:

- We propose a distributed algorithm based on local search where robots collaboratively build a team plan by proposing modifications to the collective trajectories;
- We reduce its computation and communication requirements by prioritizing search orders of local search and warm starting with greedy solutions, respectively;
- We show that the proposed algorithm outperforms a state-of-the-art algorithm for multi-robot target tracking in coordinating a team of heterogeneous robots, while trading off sensing performance and energy expenditure.

II. Preliminaries

We review some useful definitions. Let $g : 2^{\mathcal{M}} \to \mathbb{R}$ be a set function defined on the ground set $\mathcal{M}$ consisting of finite elements. Let $g(a|S) := g(S \cup \{a\}) - g(S)$ be the discrete derivative, or the marginal gain, of g at S with respect to a.

Definition 1 (Submodularity)**.** Function g is submodular if for any $S_1 \subseteq S_2 \subseteq \mathcal{M}$ and $a \in \mathcal{M} \backslash S_2$, $g(a|S_1) \geq g(a|S_2)$.

Definition 2 (Monotonicity)**.** Function g is monotone if for any $S_1 \subseteq S_2 \subseteq \mathcal{M}$, $g(S_1) \leq g(S_2)$.

III. Problem Formulation

Consider robots indexed by $i \in \mathcal{R} := \{1, \dots, n\}$, whose states are $x_{i,t} \in \mathcal{X}_i$ at time $t = 0, \dots, T$, and dynamics are:

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t}), \tag{1}$$

where $u_{i,t} \in \mathcal{U}_i$ is the control input and $\mathcal{U}_i$ is a finite set. We denote a control sequence as $\sigma_i = u_{i,0}, \dots, u_{i,T-1} \in \mathcal{U}_i^T$.

The robots' goal is to track targets with state $y \in \mathbb{R}^{d_y}$ that have the following linear-Gaussian motion model:

$$y_{t+1} = A_t y_t + w_t, \quad w_t \sim \mathcal{N}(0, W_t), \tag{2}$$

where $A_t \in \mathbb{R}^{d_y \times d_y}$ and w_t is a zero-mean Gaussian noise with covariance $W_t \succeq 0$. The robots have sensors that measure the target state subject to an observation model:

$$z_{i,t} = H_{i,t}(x_{i,t}) y_t + v_{i,t}(x_{i,t}), \quad v_{i,t} \sim \mathcal{N}(0, V_{i,t}(x_{i,t})), \tag{3}$$

where $z_{i,t} \in \mathbb{R}^{d_{z_i}}$ is the measurement taken by robot i in state $x_{i,t}$, $H_{i,t}(x_{i,t}) \in \mathbb{R}^{d_{z_i} \times d_y}$, and $v_{i,t}(x_{i,t})$ is a state-dependent Gaussian noise, whose values are independent at any pair of times and across sensors. The observation model is linear in target states but can be nonlinear in robot states. If it depends nonlinearly on target states, we can linearize it around an estimate of target states to get a linear model.

We assume every robot i has access to N_i control trajectories $\mathcal{M}_i = \{\sigma_i^k\}_{k=1}^{N_i}$ to choose from. Denote the set of all control trajectories as $\mathcal{M} = \cup_{i=1}^n \mathcal{M}_i$ and its size as $N = |\mathcal{M}|$. Potential control trajectories can be generated by various single-robot information gathering algorithms such as [24], [34]–[36]. The fact that every robot cannot execute more than one trajectory can be encoded as a partition matroid $(\mathcal{M}, \mathcal{I})$, where $\mathcal{M}$ is the ground set, and $\mathcal{I} = \{S \subseteq \mathcal{M} \mid |S \cap \mathcal{M}_i| \leq 1 \; \forall i \in \mathcal{R}\}$ consists of all admissible subsets of trajectories. Given $S \in \mathcal{I}$, we denote the joint state of robots that have been assigned trajectories as $x_{S,t}$ at time t, and their indices as $\mathcal{R}_S := \{i \mid |\mathcal{M}_i \cap S| = 1 \; \forall \; i \in \mathcal{R}\}$. Also, denote the measurements up to time $t \leq T$ collected by robots $i \in \mathcal{R}_S$ who follow the trajectories in S by $z_{S,1:t}$.

Due to the linear-Gaussian assumptions in (2) and (3), the optimal estimator for the target states is a Kalman filter. The target estimate covariance $\Sigma_{S,t}$ at time t resulting from robots $\mathcal{R}_S$ following trajectories in S obeys:

$$\Sigma_{S,t+1} = \rho^{\mathrm{e}}_{S,t+1}(\rho^{\mathrm{p}}_t(\Sigma_{S,t}), x_{S,t+1}), \tag{4}$$

where $\rho^{\mathrm{p}}_t(\cdot)$ and $\rho^{\mathrm{e}}_{S,t}(\cdot,\cdot)$ are the Kalman filter prediction and measurement updates, respectively:

$$\textbf{Predict:} \qquad \rho^{\mathrm{p}}_t(\Sigma) := A_t \Sigma A_t^\top + W_t,$$

$$\textbf{Update:} \quad \rho^{\mathrm{e}}_{S,t}(\Sigma, x_{S,t}) := \left(\Sigma^{-1} + \sum_{i \in \mathcal{R}_S} M_{i,t}(x_{i,t})\right)^{-1},$$

$$M_{i,t}(x_{i,t}) := H_{i,t}(x_{i,t}) V_{i,t}(x_{i,t})^{-1} H_{i,t}(x_{i,t})^\top.$$

When choosing sensing trajectories, we want to capture the trade-off between sensing performance and energy expenditure, which is formalized below.

Problem 1 (Energy-Aware Active Information Acquisition)**.** Given initial states $x_{i,0} \in \mathcal{X}_i$ for every robot $i \in \mathcal{R}$, a prior distribution of target state y_0, and a finite planning horizon T, find a set of trajectories $S \in \mathcal{M}$ to optimize the following:

$$\max_{S \in \mathcal{I}} \; J(S) := \mathbb{I}(y_{1:T}; z_{S,1:T}) - C(S), \tag{5}$$

where $\mathbb{I}(y_{1:T}; z_{S,1:T}) = \frac{1}{2}\sum_{t=1}^{T} \left[\log\det\left(\rho^{\mathrm{p}}_{t-1}(\Sigma_{S,t-1})\right) - \log\det(\Sigma_{S,t})\right] \geq 0$ is the mutual information between target states and observations[1], and $C : 2^{\mathcal{M}} \to \mathbb{R}$ is defined as:

$$C(S) := \sum_{\sigma_i \in S} r_i \, C_i(\sigma_i), \tag{6}$$

where $0 \leq C_i(\cdot) \leq c^{\max}$ is a non-negative, bounded energy cost for robot i to apply controls σ_i weighted by $r_i \geq 0$.

Remark 1. Robots are assumed to know others' motion models (1) and observation models (3) before the mission, so that any robot can evaluate (5) given a set of trajectories.

Remark 2. The optimization problem (5) is non-monotone, because adding extra trajectories may worsen the objective by incurring high energy cost $C(S)$. Thus, the constraint $S \in \mathcal{I}$ may not be tight, i.e., some robots may not get assigned trajectories. This property is useful when a large repository of heterogeneous robots is available but only a subset is necessary for the given tasks.

Remark 3. The choice of (5) is motivated by the energy-aware target tracking application. However, the proposed algorithm in Sec. IV is applicable to any scenario where $J(S)$ is a submodular set function that is not necessarily monotone, but can be made non-negative with a proper offset.

Solving Problem 1 is challenging because adding energy cost $C(S)$ breaks the monotonicity of the objective, a property required for approximation methods (e.g., coordinate descent [2] and greedy algorithm [27]) to maintain performance guarantees. This is because these methods only add elements to the solution set, which always improves a monotone objective, but can worsen the objective in our setting, and may yield arbitrarily poor performance. We now propose a new distributed algorithm based on local search [19].

IV. Multi-Robot Planning

We first present how local search [19] can be used to solve Problem 1 with near-optimal performance guarantee. Despite the guarantee, local search is not suitable for distributed robot teams, because it assumes access to all locally planned robot control trajectories which can be communication-expensive to gather. To address this problem, we propose a new distributed algorithm that exploits the structure of a partition matroid to allow robots to collaboratively build a team plan by repeatedly proposing changes to the collective trajectories. Moreover, we develop techniques to reduce its computation and communication to improve scalability.

In the following subsections, we denote $g : 2^{\mathcal{M}} \to \mathbb{R}$ as the non-negative, submodular oracle function used by local search, where the ground set $\mathcal{M}$ contains robot trajectories.

[1]Our problem differs from sensor placement problems that consider the mutual information between selected and not selected sensing locations.

Algorithm 1 Centralized Local Search [19] (`CLS`)

1: **require** $\alpha > 0$, ground set $\mathcal{M}$, admissible subsets $\mathcal{I}$, oracle g
2: $N \leftarrow |\mathcal{M}|$
3: $S_1, S_2 \leftarrow \emptyset$
4: **for** $k = 1, 2$ **do**
5: $\quad S_k \leftarrow \{\arg\max_{a \in \mathcal{M}} g(\{a\})\}$ // Initialize with best traj.
6: $\quad$ **while** resultant S'_k from ①, ② or ③ satisfies $S'_k \in \mathcal{I}$ and $g(S'_k) \geq (1 + \frac{\alpha}{N^4}) g(S_k)$ **do** $S_k \leftarrow S'_k$ // Repeat local operations
7: $\quad\quad$ ① **Delete:** $S'_k \leftarrow S_k \backslash \{d\}$, where $d \in S_k$
8: $\quad\quad$ ② **Add:** $S'_k \leftarrow S_k \cup \{a\}$, where $a \in \mathcal{M} \backslash S_k$
9: $\quad\quad$ ③ **Swap:** $S'_k \leftarrow S_k \backslash \{d\} \cup \{a\}$, where $d \in S_k$, $a \in \mathcal{M} \backslash S_k$
10: $\quad \mathcal{M} \leftarrow \mathcal{M} \backslash S_k$
11: **return** $\arg\max_{S \in \{S_1, S_2\}} g(S)$

A. Centralized Local Search (`CLS`)

We present the original local search [19] in our setting with a single partition matroid constraint. We refer to it as centralized local search (`CLS`, Alg. 1) because it requires access to trajectories $\mathcal{M}$ from all robots. The algorithm proceeds in two rounds to find two candidate solutions $S_1, S_2 \in \mathcal{I}$. In each round $k = 1, 2$, solution S_k is initialized with a single-robot trajectory maximizing the objective (Line 5). Repeatedly, S_k is modified by executing one of the **Delete**, **Add** or **Swap** operations, if it improves the objective by at least $(1 + \frac{\alpha}{N^4})$ of its original value (Lines 6–9), where $\alpha > 0$ controls run-time and performance guarantee. This procedure continues until S_k is no longer updated, and the next round begins without considering S_k in the ground set $\mathcal{M}$ (Line 10). Lastly, the better of S_1 and S_2 is returned.

One important requirement of `CLS` is that the objective function g is non-negative. With the objective from Problem 1, this may not be true, so we add an offset O. The next proposition provides a worst-case performance guarantee for applying Alg. 1 to Problem 1 after properly offsetting the objective to be non-negative.

Proposition 1. *Consider that we solve Problem 1 whose objective is made non-negative by adding a constant offset:*

$$\max_{S \in \mathcal{I}} \; g(S) := J(S) + O, \tag{7}$$

where $O := \sum_{i=1}^{n} r_i c^{\max}$. *Denote* S^* *and* S^{ls} *as the optimal solution and solution obtained by* `CLS` *(Alg. 1) for* (7)*, by using* $g(\cdot)$ *as the oracle. We have the following worst-case performance guarantee for the objective:*

$$0 \leq g(S^*) \leq 4(1 + \alpha) g(S^{\mathrm{ls}}). \tag{8}$$

Proof. In (5), mutual information is a submodular set function defined on measurements provided by selected trajectories [2]. Moreover, $C(S)$ is modular given its additive nature:

$$C(S) = \sum_{\sigma_i \in S} r_i C_i(\sigma_i) \geq 0. \tag{9}$$

Since mutual information is non-negative, (7) is a submodular non-monotone maximization problem with a partition matroid constraint. Setting $k = 1$ and $\epsilon = \alpha$ in [19, Thm. 4], the proposition follows directly after rearranging terms. □

Remark 4. Having the constant O term in (7) does not change the optimization in Problem 1, but ensures that the oracle used by CLS (Alg. 1) is non-negative so that the ratio $(1+\frac{\alpha}{N^4})$ correctly reflects the sufficient improvement condition.

Besides the communication aspect that CLS requires access to all robot trajectories, running it naively can incur significant computation. In the worst case, CLS requires $\mathcal{O}(\frac{1}{\alpha}N^6\log(N))$ oracle calls[2], where N is the total number of trajectories [19]. Even on a central server, run-time may be greatly reduced by using our proposed method (see Sec. V).

B. Distributed Local Search (DLS)

This section proposes a distributed implementation of local search (see Algs. 2 and 3 written for robot i). Exploiting the structure of the partition matroid, DLS enables each robot to propose local operations based on its own trajectory set, while guaranteeing that the team solution never contains more than one trajectory for every robot. All steps executed by CLS can be distributedly proposed, so DLS provides the same performance guarantee in Theorem 1. By prioritizing search orders and starting with greedy solutions, we reduce computation and communication of DLS, respectively.

1) Distributed Proposal: Every proposal consists of two trajectories (d, a), where d is to be deleted from and a is to be added to the solution set. We also define a special symbol "NOP" that leads to no set operation, i.e., $S_k \cup \{\texttt{NOP}\} = S_k \backslash \{\texttt{NOP}\} = S_k$. Note that $(d, \texttt{NOP})$, $(\texttt{NOP}, a)$ and (d, a) are equivalent to the **Delete**, **Add** and **Swap** steps in CLS.

Every robot i starts by sharing the size of its trajectory set $|\mathcal{M}_i|$ and its best trajectory $a_i^* \in \mathcal{M}_i$ in order to initialize S_k and N collaboratively (Alg. 2 Lines 5–7). Repeatedly, every robot i executes the subroutine FindProposal (Alg. 3) in parallel, in order to propose changes to S_k (Alg. 2 Lines 8–13). Since any valid proposal shared by robots improves the objective, the first $(d, a) \neq (\texttt{NOP}, \texttt{NOP})$ will be used by all robots to update S_k in every round (Alg. 2 Lines 10–12). We assume instantaneous communication, so robots always use a common proposal to update their copies of S_k. Otherwise, if delay leads to multiple valid proposals, a resolution scheme is required to ensure robots pick the same proposal.

In FindProposal (Alg. 3), an outer loop looks for potential deletion $d \in S_k$ (Alg. 3 Lines 2–6). Otherwise, further adding $a \in \mathcal{M}_i$ is considered, as long as the partition matroid constraint is not violated (Alg. 3 Lines 7–8). Next, we discuss how to efficiently search for trajectories to add.

2) Lazy Search: Instead of searching over trajectories in an arbitrary order, we can prioritize the ones that already perform well by themselves, based on $g(a|\emptyset)$ for all $a \in \mathcal{M}_i$ (Alg. 2 Line 2). In this fashion, we are more likely to find trajectories that provide sufficient improvement earlier (Alg. 3 Lines 12–13). Note that $g(a|\emptyset)$ is typically a byproduct of the trajectory generation process, so it can be saved and reused.

This ordering also allows us to prune unpromising trajectories. Given the team solution after deletion $S_k^- := S \backslash \{d\}$, the required marginal gain for later adding trajectory a is

$$g(a|S_k^-) \geq \Delta := (1+\frac{\alpha}{N^4})g(S_k) - g(S_k^-). \tag{10}$$

We can prune any $a \in \mathcal{M}_i$, if $g(a|\emptyset) < \Delta$ based on the diminishing return property: because $\emptyset \subseteq S_k^-$, we know that $\Delta > g(a|\emptyset) \geq g(a|S_k^-)$, violating condition (10). Similarly, all subsequent trajectories a' can be ignored, because their marginal gains $g(a'|\emptyset) \leq g(a|\emptyset) < \Delta$ due to ordering (Alg. 3 Lines 10–11). Lastly, if an addition improves S_k^- sufficiently, the proposal is broadcasted (Alg. 3 Lines 12–13).

3) Greedy Warm Start: We observe empirically that a robot tends to swap its own trajectories consecutively for small growth in the objective, increasing communication unnecessarily. This can be mitigated by a simple technique: when finding local operations initially, we force robots to only propose additions to greedily maximize the objective, until doing so does not lead to enough improvement or violates the matroid constraint. Then robots resume Alg. 3 and allow all local operations. By warm starting the team solution greedily, every robot aggregates numerous proposals with smaller increase in the objective into a greedy addition with larger increase, thus effectively reducing communication.

[2]For 2 solution candidates, each requires $\mathcal{O}(\frac{1}{\alpha}N^4\log(N))$ local operations, and N^2 oracle calls to find each local operation in the worst case.

Algorithm 2 Distributed Local Search (DLS)

1: **require** $\alpha > 0$, trajectories $\mathcal{M}_i$, oracle g
2: Sort $\mathcal{M}_i$ in descending order based on $g(a|\emptyset)$ for all $a \in \mathcal{M}_i$
3: $S_1, S_2 \leftarrow \emptyset$
4: **for** $k = 1, 2$ **do**
5: Broadcast $|\mathcal{M}_i|$ and $a_i^* \in \mathcal{M}_i$ that maximizes $g(\{a_i^*\})$
6: $S_k \leftarrow \{a^*\}$, where $a^* \in \{a_i^*\}_{i=1}^n$ maximizes $g(\{a^*\})$
7: $N \leftarrow \sum_{i=1}^n |\mathcal{M}_i|$
8: **repeat**
9: Run FindProposal$(S_k, \mathcal{M}_i, \alpha, N, g)$ in background
10: **if** Receive $(d, a) \neq (\texttt{NOP}, \texttt{NOP})$ **then**
11: Terminate FindProposal if it has not finished
12: $S_k \leftarrow S_k \backslash \{d\} \cup \{a\}$
13: **until** Receive $(d, a) = (\texttt{NOP}, \texttt{NOP})$ from all robots
14: $\mathcal{M}_i \leftarrow \mathcal{M}_i \backslash S_k$
15: **return** $\arg\max_{S \in \{S_1, S_2\}} g(S)$

Algorithm 3 Find Proposal (FindProposal)

1: **require** S_k, $\mathcal{M}_i$, $\alpha > 0$, N, g
2: **for** $d \in S_k$ or $d = \texttt{NOP}$ **do** // Delete d, or no deletion
3: $S_k^- \leftarrow S_k \backslash \{d\}$
4: $\Delta \leftarrow (1+\frac{\alpha}{N^4})g(S_k) - g(S_k^-)$ // Δ: deficiency of S_k^-
5: **if** $\Delta \leq 0$ **then**
6: **broadcast** $(d, \texttt{NOP})$
7: **if** $\exists\ a \in S_k^-$ planned by robot i **then**
8: **continue** // Cannot add due to partition matroid
9: **for** $a \in \mathcal{M}_i$ in sorted order **do** // Add a
10: **if** $g(a|\emptyset) < \Delta$ **then**
11: **break** // No $a \in \mathcal{M}_i$ will improve S_k^- enough
12: **if** $g(a|S_k^-) \geq \Delta$ **then**
13: **broadcast** (d, a)
14: **broadcast** $(\texttt{NOP}, \texttt{NOP})$

V. Simulation Results

We evaluate DLS in two target tracking scenarios based on objective values, computation, communication, and ability to handle heterogeneous robots. Its performance is compared against coordinate descent (CD [2]), a state-of-the-art algorithm for multi-robot target tracking that, however, assumes monotonicity of the objective. Planning for robots sequentially, CD allows every robot to incorporate the plans of previous robots. We also allow CD to not assign anything to a robot if it worsens the objective. Reduced value iteration [34] is used to generate trajectories for both algorithms. Comparisons between CLS and DLS are omitted because the two algorithms empirically achieve the same average performance. We set $\alpha = 1$ arbitrarily, because tuning it was not effective due to the large number of trajectories N.

Both DLS and CD are implemented in C++ and evaluated in simulation on a laptop with an Intel Core i7 CPU. For DLS, every robot owns separate threads, and executes Alg. 3 over 4 extra threads to exploit its parallel structure. Similarly, CD allows every robot to use 4 threads and additionally incorporates accelerated greedy [33] for extra speed-up.

A. Characteristics of Robots

Given initial state $x_{i,0} \in \mathcal{X}_i$ for robot $i \in \mathcal{R}_S$ who follows the control sequence $u_{i,0}, \dots, u_{i,T-1} = \sigma_i \in S$, the resultant states are $x_{i,1}, \dots, x_{i,T}$ based on dynamics (1). The energy cost $C(S)$ may also be state-dependent. We define it as:

$$C(S) := \sum_{i \in \mathcal{R}_S} r_i \sum_{t=0}^{T-1} \left(c_i^{\text{ctrl}}(u_{i,t}) + c_i^{\text{state}}(x_{i,t}) \right), \qquad (11)$$

where the state-dependent cost $c_i^{\text{state}}(\cdot)$ and control-dependent cost $c_i^{\text{ctrl}}(\cdot)$ are defined based on robot types—in our case, robot i is either an unmanned ground vehicle (UGV) or an unmanned aerial vehicle (UAV). Note that decomposition between state and control is not required for our framework to work. The setup for robots are summarized in Table I. For simplicity, all robots follow differential-drive dynamics[3] with sampling period $\tau = 0.5$ and motion primitives consisting of linear and angular velocities $\{u = (\nu, \omega) \mid \nu \in \{0, 8\}$ m/s, $\omega \in \{0, \pm\frac{\pi}{2}\}$ rad/s$\}$. We consider muddy and windy regions that incur state-dependent costs for UGVs and UAVs, respectively. The robots have range and bearing sensors, whose measurement noise covariances grow linearly with target distance. Within limited ranges and field of views (FOVs), the maximum noise standard deviations are 0.1 m and 5° for range and bearing measurements, respectively. Outside the ranges or field of views, measurement noise becomes infinite. Please refer to [20] for more details.

B. Scenario 1: Multi-Robot Dynamic Target Tracking

Here we show the computation and communication savings for DLS, and compare the performance of DLS and CD (see Figs. 2 and 3). The scenario involves $2, \dots, 10$ UGVs trying to estimate the positions and velocities of the same number of dynamic targets. The targets follow discretized double integrator models corrupted by Gaussian noise, with a top speed of 2 m/s. Robots and targets are spawned in a square arena whose sides grow from 40 m to 60 m, and 50 random trials are run for each number of robots.

Non-monotonicity in the problem is accentuated by an increasing penalty for control effort of additional robots, by setting $r_i = i$ for each robot i as defined in (11) (i.e., the

[3]We note that any dynamically feasible model can be used for the specific robot which is being planned for. We use the same kinematic model for the quadrotor and ground vehicle for implementation convenience, and because the quadrotors are restricted to a plane to avoid collisions.

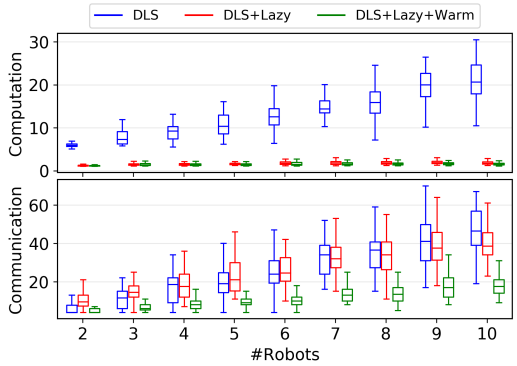

Fig. 2. Computation and communication savings afforded by lazy search (*Lazy*) and greedy warm start (*Warm*) for DLS. Computation is measured by total oracle calls divided by the number of trajectories N, where N reaches around 12500 for 10 robots. Communication is measured by the number of proposal exchanges. Combining lazy search and greedy warm start (green) leads to 80–92% computation reduction, and up to 60% communication reduction compared to the naive implementation (blue) on average.

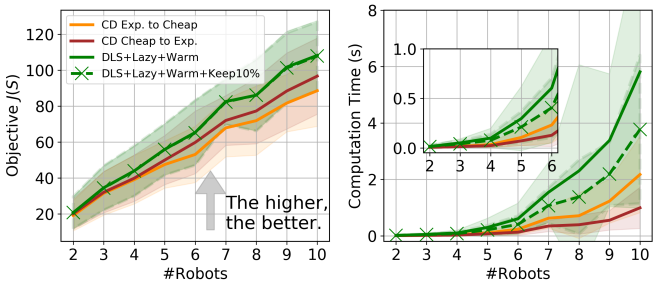

Fig. 3. Objective values and computation time (s) for variants of DLS and CD, where the lines and shaded areas show the mean and standard deviation, respectively. The time excludes the trajectory generation time (< 2 s), which is the same for every algorithm. DLS (solid green) consistently outperforms CD in optimizing the objective, where it is better for CD to plan from cheaper to more expensive robots (brown), rather than the reverse order (orange). The performance gap between DLS and CD widens as more costly robots increase non-monotonicity of the problem. However, DLS requires longer run-time, which in practice can be alleviated by using a portion of all trajectories. This invalidates the worst-case guarantee, but DLS solution based on the best 10% of each robot's trajectories (green crosses) still outperforms CD.

TABLE I

Robot setup in two experimental scenarios.

	$c^{\text{ctrl}}(u)$, u given as			$c^{\text{state}}(x)$, x in		FOV (°)	Range (m)
	$0, 0$	$0, \frac{\pm\pi}{2}$	$8, \frac{\pm\pi}{2}$	Mud	Wind	Exp.1&2	Exp.1&2
UGV	0	1	2	3	/	160	6 & 15
UAV	2	2	4	/	3	360	/ & 20

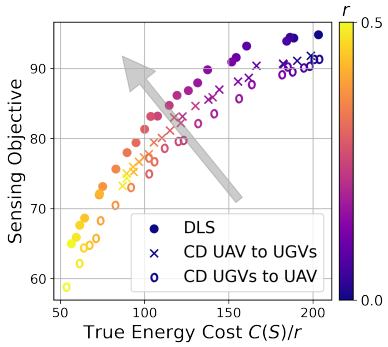

Fig. 4. Trade-off between sensing performance (mutual information (5)) and the true energy expenditure $C(S)/r$ in heterogeneous robot experiments produced by DLS and CD, where it is better to be in the upper left. Each point is an average obtained over 50 trials for a fixed r, where we set $r_i = r$ for each robot i to penalize the team energy expenditure per (11).

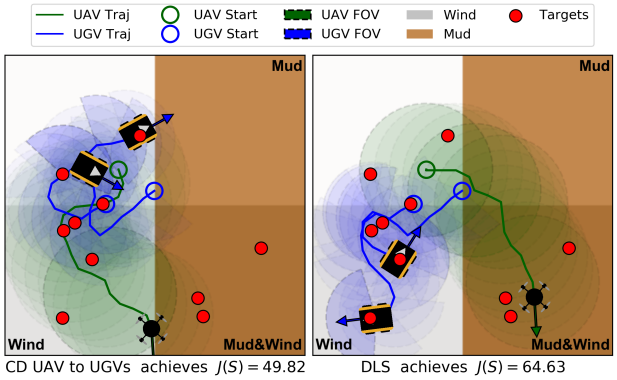

Fig. 5. Example solutions from CD (left) and DLS (right) for 2 UGVs and 1 UAV with $r = 0.2$ that penalizes energy cost $C(S)$ in (11). The arena is both windy and muddy, which is costly for the UAV and UGVs, respectively. (Left) CD performs poorly due to its fixed planning order: the UAV plans first to hover near the targets on the left, rather than venturing over the mud. Thus, the UGVs are under-utilized because they are unwilling to go into the mud to observe the targets on the bottom right. For similar reasons, CD with reversed order under-utilizes the UAV, which is not visualized due to limited space. (Right) In contrast, DLS deploys the UAV over the muddy regions, leading to a better value of $J(S)$ in (5).

10-th added robot is 10 times more expensive to move than the first). Note that state-dependent cost is set to 0 only for this experiment. Trajectory generation has parameters $\epsilon = 1$ and $\delta = 2$ for horizon $T = 10$. As the planning order is arbitrary for CD, we investigate two planning orders: first from cheaper to more expensive robots, and then the reverse. Intuitively and shown in Fig. 3, the former should perform better, because the same amount of information can be gathered while spending less energy. While other orderings are possible (e.g., [18], [22]), we only use two to show CD's susceptibility to poor planning order. For a fair comparison between DLS and CD, we use a fixed set of trajectories generated offline, but ideally trajectories should be replanned online for adaptive dynamic target tracking.

Proposed methods for improving naive distributed execution of local search, namely lazy search (*Lazy*) and greedy warm start (*Warm*), are shown to reduce computation by 80–92% and communication by up to 60% on average, as shown in Fig. 2. As expected, when there are few robots with similar control penalties, the objective is still close to being monotone, and DLS and CD perform similarly as seen in Fig. 3. However, as more costly robots are added, their contributions in information gain are offset by high control penalty, so the problem becomes more non-monotone. Therefore, the performance gap between DLS and CD widens, because CD requires monotonicity to maintain its performance guarantee, but DLS does not. From Fig. 3, we can see that planning order is critical for CD to perform well, but a good ordering is often unknown a priori. Compared to CD which requires only $n-1$ communication rounds for n robots, DLS requires more for its performance. For practical concerns to save more time, DLS with down-sampled trajectories (e.g., keeping the best 10% of each robot's trajectories) still produces better solution than CD, but the guarantee of DLS no longer holds.

C. Scenario 2: Heterogeneous Sensing and Control

Now consider a heterogeneous team with 2 UGVs and 1 UAV with different sensing and control profiles (Table I) tracking 10 static targets in a 100 m $\times$ 100 m arena over a longer horizon $T = 20$ (see Fig. 5). The UAV has better sensing range and field of view compared to UGVs, but consumes more energy. The arena has overlapping muddy and windy regions, so robots must collaboratively decide which should venture into the costly regions. To explore the trade-off between sensing and energy objectives as a team, we set $r_i = r,\ \forall i$ and then, as we vary r from 0 to 0.5, we run 50 trials for each value. Robots are spawned in the non-muddy, non-windy region, but targets may appear anywhere. We set $\delta = 4$ to handle the longer horizon, and evaluate two CD planning orders: from UAV to UGVs, and the reverse.

As shown in Fig. 4, DLS consistently achieves better sensing and energy trade-off than CD on average. To gain intuitions on why CD under-performs, a particular trial given $r = 0.2$ is shown in Fig. 5. Due to the non-monotone objective, the robot who plans first to maximize its own objective can hinder robots who plan later, thus negatively affecting team performance.

VI. Conclusion

This work considered a multi-robot information gathering problem with non-monotone objective that captures the trade-off between sensing benefits and energy expenditure. We proposed a distributed algorithm based on local search and reduced its computation and communication requirements by using lazy and greedy methods. The proposed algorithm was evaluated in two target tracking scenarios and outperformed the state-of-the-art coordinate descent method. Future work will focus on scaling the algorithm to larger robot teams by exploiting spatial separation, formalizing heterogeneity, and carrying out hardware experiments.

Acknowledgment

This research was supported in part by Boeing Research & Technology and ARL DCIST CRA W911NF-17-2-0181.

REFERENCES

[1] L. Carlone, J. Du, M. K. Ng, B. Bona, and M. Indri, "Active slam and exploration with particle filters using kullback-leibler divergence," *Journal of Intelligent & Robotic Systems*, vol. 75, no. 2, pp. 291–311, 2014.

[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 International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4775–4782.

[3] E. López, S. García, R. Barea, L. M. Bergasa, E. J. Molinos, R. Arroyo, E. Romera, and S. Pardo, "A multi-sensorial simultaneous localization and mapping (slam) system for low-cost micro aerial vehicles in gps-denied environments," *Sensors*, vol. 17, no. 4, p. 802, 2017.

[4] M. Corah and N. Michael, "Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice," *Autonomous Robots*, vol. 43, no. 2, pp. 485–501, 2019.

[5] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," *IEEE Pervasive computing*, vol. 3, no. 4, pp. 24–33, 2004.

[6] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple uas," in *International Symposium on Experimental Robotics*. Springer, 2018, pp. 140–152.

[7] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot systems for search and rescue: Coordination and perception," *arXiv preprint arXiv:2008.12610*, 2020.

[8] F. Shkurti, A. Xu, M. Meghjani, J. C. G. Higuera, Y. Girdhar, P. Giguere, B. B. Dey, J. Li, A. Kalmbach, C. Prahacs *et al.*, "Multi-domain monitoring of marine environments using a heterogeneous robot team," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1747–1753.

[9] X. Lan and M. Schwager, "Rapidly exploring random cycles: Persistent estimation of spatiotemporal fields with multiple sensing robots," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1230–1244, 2016.

[10] G. Notomista and M. Egerstedt, "Persistification of robotic tasks," *IEEE Transactions on Control Systems Technology*, 2020.

[11] M. Popović, T. Vidal-Calleja, G. Hitz, J. J. Chung, I. Sa, R. Siegwart, and J. Nieto, "An informative path planning framework for uav-based terrain monitoring," *Autonomous Robots*, pp. 1–23, 2020.

[12] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–25, 2006.

[13] G. S. Sukhatme, A. Dhariwal, B. Zhang, C. Oberg, B. Stauffer, and D. A. Caron, "Design and development of a wireless robotic networked aquatic microbial observing system," *Environmental Engineering Science*, vol. 24, no. 2, pp. 205–215, 2007.

[14] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[15] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: a survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–31, 2019.

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

[17] A. Singh, A. Krause, and W. Kaiser, "Nonmyopic adaptive informative path planning for multiple robots," in *IJCAI*, 2009.

[18] S. Jorgensen, R. H. Chen, M. B. Milam, and M. Pavone, "The team surviving orienteers problem: routing teams of robots in uncertain environments with survival constraints," *Autonomous Robots*, vol. 42, no. 4, pp. 927–952, 2018.

[19] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, "Non-monotone submodular maximization under matroid and knapsack constraints," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 323–332.

[20] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas, "Anytime planning for decentralized multirobot active information gathering," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1025–1032, 2018.

[21] Y. Kantaros, B. Schlotfeldt, N. Atanasov, and G. J. Pappas, "Asymptotically optimal planning for non-myopic multi-robot information gathering." in *Robotics: Science and Systems*, 2019.

[22] P. Dames, P. Tokekar, and V. Kumar, "Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1540–1553, 2017.

[23] A. Viseras, Z. Xu, and L. Merino, "Distributed multi-robot information gathering under spatio-temporal inter-robot constraints," *Sensors*, vol. 20, no. 2, p. 484, 2020.

[24] D. Levine, B. Luders, and J. How, "Information-rich path planning with general constraints using rapidly-exploring random trees," in *AIAA Infotech@ Aerospace 2010*, 2010, p. 3360.

[25] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-mcts: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[26] N. A. Atanasov, "Active information acquisition with mobile robots," Ph.D. dissertation, University of Pennsylvania, 2015.

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

[28] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar, "Constrained non-monotone submodular maximization: Offline and secretary algorithms," in *International Workshop on Internet and Network Economics*. Springer, 2010, pp. 246–257.

[29] M. Feldman, J. Naor, and R. Schwartz, "A unified continuous greedy algorithm for submodular maximization," in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE, 2011, pp. 570–579.

[30] S. O. Gharan and J. Vondrák, "Submodular maximization by simulated annealing," in *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2011, pp. 1098–1116.

[31] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, and V. Segui, "Decentralised submodular multi-robot task allocation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2829–2834.

[32] H.-S. Shin, T. Li, and P. Segui-Gasco, "Sample greedy based task allocation for multiple robot systems," *arXiv preprint arXiv:1901.03258*, 2019.

[33] M. Minoux, "Accelerated greedy algorithms for maximizing submodular set functions," in *Optimization techniques*. Springer, 1978, pp. 234–243.

[34] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Information acquisition with sensing robots: Algorithms and error bounds," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6447–6454.

[35] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[36] X. Lan and M. Schwager, "Planning periodic persistent monitoring trajectories for sensing robots in gaussian random fields," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2415–2420.