Autonomous Robots manuscript No.
(will be inserted by the editor)

A Penalized Batch-Bayesian Approach to Informative Path Planning for Decentralized Swarm Robotic Search

Payam Ghassemi · Mark Balazon · Souma Chowdhury

Received: date / Accepted: date

Abstract Swarm-robotic approaches to search and target localization, where target sources emit a spatially varying signal, promise unparalleled time efficiency and robustness. With most existing swarm search methods, it remains challenging to simultaneously preserve search efficiency and mathematical insight along with scalability and computational tractability. Our recently developed decentralized method, *Bayes-Swarm-O*, a model-based approach founded on batch Bayesian Optimization, has been shown to outperform state-of-the-art swarm heuristics in terms of search efficiency. However, this original *Bayes-Swarm-O* method did not account for the interactions between robots' decisions (aka samples in a batch) and was found to be sensitive to the prescribed balance between exploration and exploration. These limitations are alleviated in this paper, leading to significantly improved search efficiency and convergence, by respectively using a new marginalization penalization approach to embodied batch sampling and a dynamic adaptation of the exploration/exploitation balance during mission. In addition, this paper presents a systematic set of experiments executed through a new Pybullet-based distributed swarm search simulator, that analyzes the impact of increasing swarm size, partial peer observation, and choice of optimizer, on *Bayes-Swarm-P*. The advanced *Bayes-Swarm-P* method is also found to be clearly superior in terms of search efficiency and robustness when compared to three standard swarm search methods (namely Glowworm search, Levy walk, and exhaustive search) over simulated multi-modal signal distributions and a skier/avalanche search and rescue problem.

P. Ghassemi
Department of Mechanical and Aerospace Engineering,
University at Buffalo,
Buffalo, NY, 14260 USA
E-mail: payamgha@buffalo.edu

M. Balazon
Department of Mechanical and Aerospace Engineering,
University at Buffalo,
Buffalo, NY, 14260 USA

S. Chowdhury
Department of Mechanical and Aerospace Engineering,
University at Buffalo,
Buffalo, NY, 14260 USA
E-mail: soumacho@buffalo.edu

Keywords Avalanche Search and Rescue · Decentralized · Bayesian Optimization · Informative Path Planning · Swarm-Robotic Search

1 Introduction

In time-sensitive search applications pertaining to localizing a signal source or target, a team of robots can broaden the scope of operational capabilities through distributed remote sensing, scalability, fault-tolerance, and parallelism in terms of task execution and information gathering [1,2]. This potential has led to the emergence of *swarm-robotic search* as an important sub-domain of cooperative robotics. In this context, we consider swarm systems comprising mobile robots (e.g., unmanned aerial, ground, or marine-surface vehicles) that are relatively autonomous in operation and require no centralized control; with the term "swarm-robotic" preferred here over "multi-robotic" to highlight the scalability of the proposed search method. Specifically, we are interested in a class of search problems where the goal is to time-efficiently find the source that emits a spatially varying signal.

The societal importance of this class of search problems is evidenced in well-known national and multi-national projects such as the European SHERPA project (focused on avalanche search and rescue operations based on radio-frequency beacons) [2,3], and the ImPACT Tough Robotics Challenge (ImPACT-TRC) in Japan (focused on finding disaster victims based on acoustic localization) [4]. Motivating high-impact uses of swarm-robotic search include finding source of gas leakage [5], finding location of magnetic field/radio source [6,7], target search [8], finding disaster victims [4,9,10], and finding skiers trapped under an avalanche [11,3]. For such applications, decentralized swarm-robotic systems require an informative path planning method that is computationally amenable to online implementation [12], robust across mission scenarios without needing tedious heuristics [13], explainable [14], and allow scalability of the swarm system to enhance search efficiency [1].

With the goal of providing the above-stated capabilities, the planning process for each swarm-robot can be broken down into two main sub-tasks: a) building/updating the spatial model of the signal environment; and b) deciding the next waypoint to move to. For this purpose, we recently introduced a novel informative path planning method called *Bayes-Swarm-O* [10], which is based on the batch Bayesian Optimization (BO) formalism. In this decentralized *Bayes-Swarm-O* algorithm, robots exploit Gaussian Processes (GP) to model the signal distribution over space and independently solve a 2D optimization over a special acquisition function to decide the next waypoints. To account for the embodiment of the search process and communication constraints (unlike in standard BO), the *Bayes-Swarm-O* method seeks to balance exploration/exploitation over robots' paths instead of over points, and allows asynchronous decision-making. In this paper, we build on our original *Bayes-Swarm-O* work [10], by incorporating formulations to preserve convergence properties of batch BO, mitigating the interaction between robots' waypoints for increased search efficiency, analyzing the impact of algorithmic choices within *Bayes-Swarm-P*, and testing its applicability on a simulated avalanche search & rescue problem. The remainder of this section briefly surveys the literature on swarm search algorithms, and summarizes the contributions of this paper.

1.1 Multi- and Swarm-Robotic Search

Different approaches have been proposed in the literature to perform search or signal-source localization using a team of mobile robots; these approaches can be classified into two main paradigms: 1) multi-robot search methods [1]; and 2) nature-inspired swarm intelligence (SI) methods [15]. The first class of methods includes concepts such as hierarchically cooperative control [16], model-driven strategies (e.g., decentralized partially observable Markov decision process (dec-POMDP) [17], partial differential equation aided modeling [18]), Bayesian filter with mutual information [19], and strategies based on local cues [20]. While these methods preserve the mathematical explainability of the search (informative path planning) decisions, they generally require a reasonably representative prior source model; the latter may not be available, or significantly deviate from the observed signal distribution characteristics, in practical mission scenarios. Secondly, scaling these methods from the multi-robotic level ($<$10 agents) to the swarm-robotic level (100 or more agents) becomes prohibitive in terms of online computational tractability [1].

The second class of approaches, based on nature-inspired SI principles, is dedicated to guiding the search behavior for larger teams [21,15], with their favorable scalability with swarm size attributed to the low computational complexity of each decision step at the individual robot level. SI-based heuristics have been used to design algorithms both for search in non-embodied n-D space (e.g., particle swarm optimization), and for search over 2D/3D physical environments [22,23] involving both single source [12,24,25] and multiple source localization [26]. Among these methods, the few that address the problem of localizing the strongest signal-source in the presence of other weaker sources (or deal with multi-modal spatial distributions) resort to limiting assumptions such as distributed starting points [26]. Most of the SI methods also require problem specific heuristics, which limits wider or out-of-the-box deployability of the swarm system. Lastly, with swarm intelligence based methods, the process of inferring the task space (or building a belief model of the signal environment) and selecting a task within it (waypoint planning) are not separable; and as a result the *emergent* behavior raises dependability (due to the use of heuristics) and mathematical explainability concerns [27]).

1.2 Bayesian Approaches to Collective Search

An alternative concept, partly related to model-based multi-robot search approaches, is to simultaneously learn the model of the signal distribution over space and use it to plan informative paths. The former imparts mathematical expainability of the search decisions. Bayesian Optimization or BO is adept at serving in this role [28]. Specifically, *batch* Bayesian inference is the natural choice here, which suits the parallel search characteristics of the swarm system. In such a simultaneous batch sampling process, the algorithm needs to maximize its utility function by considering the effect of unseen samples on each other, often termed as "*interaction among samples*" in a batch [29]. This is needed in order to minimize the optimality gap of the batch sampling approach in comparison to the ideal sequential policy of determining the same number of samples (as in a batch) and obtaining their observations. In the context of swarm-robotic search, we take the mathematical perspective that the samples in a batch refer to the waypoint decisions of the robots in the swarm. We then propose a swarm robotic search approach that is based on the batch Bayesian Optimization (batch-BO) algorithm. Here, each robot's waypoint decision maximizes an utility function that seeks to balance exploration and exploitation based on the latest Gaussian Process model of the signal environment (constructed using the collected signal observations), and mitigate interactions with the decisions of peer robots. Since the waypoint decisions of the robot, and the planned path thereof, embodies the locations at which the search environment will be sampled, "interactions between robots decisions" and "interaction between samples" are directly related. Strictly speaking, "interactions between robots decisions" must account for both "interactions between samples" and conflicts between (i.e., similarity or closeness of) the waypoints that different robots decide to visit. Our proposed approach here is unique in the sense that by accounting for "interaction between samples", it is able to adaptively account for decision conflicts as well, meaning, how close can be robots' waypoint decisions is made adaptive to the estimated variation in the signal strength, with larger variation in signal (sharper gradients) allowing greater closeness and vice versa. Note that, in this paper the term "interaction" is used mainly in the above-stated context, and does not refer to the aspect of communication among robots.

Implementing this concept of capturing the interactions and mitigating them when jointly selecting the next locations to sample (by different robots) can be done in the most optimum manner if the decisions-making process is synchronized across the swarm, i.e., where all robots take decisions at the same fixed time-intervals. However, such an implementation would cause undue delays as robots' travel times between waypoints vary (making some robots wait while others are still moving), as well as increase the dependency on perfect communication. Thus, to study the swarm's distributed search performance under with low dependency on communication [30, 31] (which tends to be imperfect in practice [32]), an asynchronous implementation is preferred and used for the case studies in this paper. However, the proposed search algorithm can be implemented in both a synchronized and an asynchronized manner. Notable examples of methods that use a modeling/planning concept in an asynchronous manner include the work by Sujit and Ghose [33] on negotiation schemes for cooperative search, and our own recent work on using batch-BO for collective search [10]. However, these existing methods do not consider the interactions between robots' paths or waypoints, leading to suboptimal decisions.

In a batch-BO setting, the interactions between waypoints can be computed using predictive distributions of the underlying Gaussian process, and marginalizing this predictive distribution over all previous batch samples [34]. However, the ensuing *optimization-marginalization* loop becomes intractable for large batches or for real-time planning, since the cost of evaluating the function and its derivative scales poorly with increasing batch size [35]. To address this issue in non-embodied search, various approximations (of the batch creation process) have been suggested [36, 29, 37, 38]. Some of these approximate techniques use a *marginalization-penalization* approach that decomposes the process into a sequence of inexpensive individual optimizations to identify the samples in a batch, prior to evaluating any of them. The interaction among the sequentially-planned samples is minimized by multiplying the acquisition function with a penalty factor. Therein lies an yet-untapped opportunity for exploring if an analogical approximation technique can be useful in an embodied (collective) search process.

1.3 Contributions of this Paper

The primary contribution of this paper is **1)** to fundamentally extend our prior work on batch BO-based swarm search (*Bayes-Swarm-O*) [10], by formulating and using a *marginalization-penalization* technique [29, 39] to enhance its convergence and search efficiency. To the best of our knowledge, we present here the first exploration of the viability and benefits of interaction-aware approximated (thus computationally efficient) batch creation in an **embodied** collective search process. In light of this primary contribution, this extended swarm search method is called *Bayes-Swarm-Penalized* or *Bayes-Swarm-P*. Additional contributions [1] of this paper can be summarized as follows: **2)** We present an automated dynamic variation (instead of prescription) of the *Bayes-Swarm-P* hyper-parameter that balances exploration/exploitation, thus seeking to enhance search adaptability; **3)** We provide statistical analyses of the performance and scalability of *Bayes-Swarm-P* under different solver choices (that act upon the special acquisition function for optimum waypoint planning) and communication constraints; **4)** We develop and test a new event-based medium-fidelity environment using Pybullet [41] to simulate asynchronous decentralized swarm search planning, with significant potential to serve as a first-of-its-kind benchmarking environment for swarm or multi-robotic search [2]. This simulator provides both standard nonlinear test functions representing multi-modal signal environments and the avalanche search & rescue environment, over which the performance

[1] Note that portion of this paper has been presented in the 2019 IEEE International Symposium on Multi-Robot and Multi-Agent Systems, and published in the subsequent proceedings [40]. Contributions 3-4 laid out here represent newer developments and numerical experiments beyond those covered in the conference version of this work.

[2] To aid the replication of results, benchmarking and further adoption of the proposed method, the implementations of *Bayes-Swarm-O* and *Bayes-Swarm-P* and the comparative methods, and the PyBullet simulation have been made available at the following repository: https://github.com/adamslab-ub/Bayes-Swarm-P.

of *Bayes-Swarm-P* is evaluated, and compared with that of the original *Bayes-Swarm-O* method, a well-known SI method (Glowworm Swarm [42]), a Levy Walk search [43], and an exhaustive search baseline.

The remaining portion of the paper is organized as follows: Mathematical background of embodied collective search with batch BO and our proposed advancements to *Bayes-Swarm-P* are described in Section 2. Sections 3 and 4 then respectively describe the numerical experiments and results, which investigate the comparative performance of *Bayes-Swarm-P* over different case studies (with test functions) and analyze its characteristics w.r.t. scalability, impact of the penalty factor, and impact of partial observance and solver choice. Description of the avalanche problem, and results of applying *Bayes-Swarm-P* on it are presented in Section 5. Then, in Section 6, we briefly discuss how the presented method can be transitioned to more realistic environments and physical demonstration in the future, and challenges expected thereof. The paper ends with concluding remarks. A summary background of GP modeling and various problem and solver settings are provided as Appendix at the end of the paper, for ease of reference.

2 The Bayes-Swarm-P Method

2.1 Bayes-Swarm-P: Overview

To develop and implement *Bayes-Swarm-P*, in this paper we make the following assumptions: i) All robots are equipped with precise localization. ii) Movement occurs in an obstacle-free 2D or 3D environment. iii) Each robot executes the *Bayes-Swarm-P* algorithm after reaching a waypoint, to update its knowledge and identify the next waypoint; the timing of which could differ among robots, thus leading to an asynchronous implementation. It should be noted that Bayes-swarm works in both synchronous and asynchronous modes, and because of some of the practical advantages of the asynchronous mode of swarm operation [30,31], we simulate Bayes-Swarm under asynchronous operation in this paper for all case studies. Figure 1 illustrates the sequence of processes and associated flow of information, encapsulating the behavior of each swarm robot. The pseudocode of our proposed *Bayes-Swarm-P* algorithm is given in Alg. 1. The primary steps within *Bayes-Swarm-P* (blocks in Fig. 1) include updating the GP model of the signal environment through recently recorded own and peer observations, using this GP model to construct a special acquisition function which is then used to perform the optimal waypoint planning, and broadcasting the waypoint decision and the recently collected own data.

Before we describe these computational and information parts of *Bayes-Swarm-P* in the following subsections, let us define the key parameters used in *Bayes-Swarm-P*:

- m: number of robots in the swarm
- $\mathcal{D}_r^{k_r} = [\mathbf{X}_r^{k_r}, \mathbf{y}_r^{k_r}]$: the observation locations ($\mathbf{X}_r^{k_r}$) and signal measurements ($\mathbf{y}_r^{k_r}$) of robot-r over its path connecting waypoints $(k_r - 1)$ and k_r, termed as "recent observations";
- $\mathcal{D}_r^{1:k_r}$: the cumulative information of robot-r up to its arrival at the k_r-th waypoint, including all self-recorded and peer-reported observations; thus we get $\mathcal{D}_r^{1:k_r} = \bigcup_{r=1}^{m}\bigcup_{i=1}^{k_r}\mathcal{D}_r^i$;
- $\hat{\mathbf{x}}_p^{k_r}$: the next planned waypoint of robot-p, known to robot-r at the time when it's at its k_r-th waypoint;
- $\hat{\mathbf{X}}_{-r}^{k_r} = \bigcup_{p=1 \wedge p \neq r}^{m} \hat{\mathbf{x}}_p^{k_r}$: the reported next waypoints of robot-r's peers by that time.

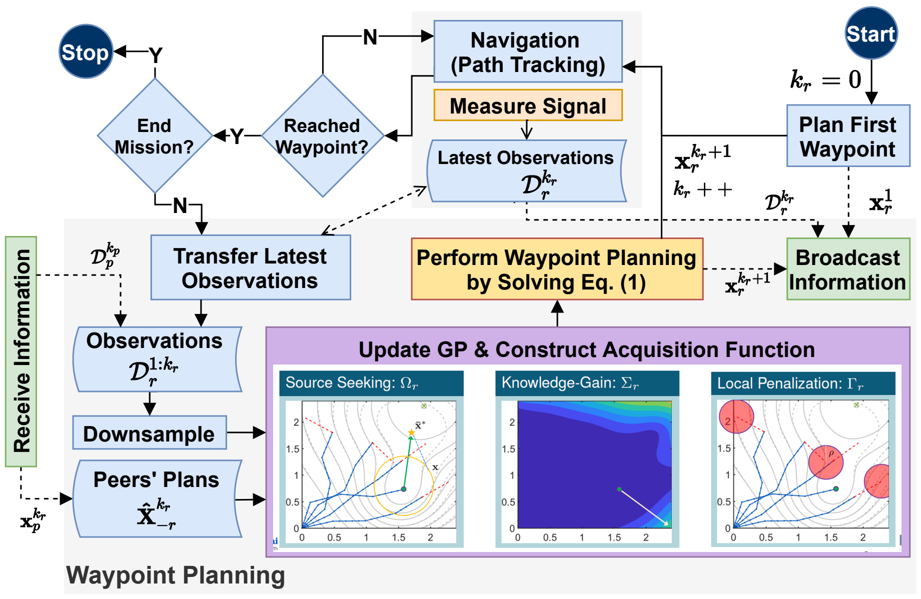
2.2 Acquisition Function

Each robot-r takes an action, i.e., plans and travels-to the next waypoint ($x_r^{k_r+1}$), that maximizes an acquisition function. Given the swarm's objective is to collectively explore a search area to find the strongest signal source among multiple sources in a robust optimal manner, the acquisition function must balance exploration and exploitation, analogical to the goals in BO, but with specific changes to account for

Algorithm 1 *Bayes-Swarm-P* Algorithm

Input: $\mathbf{x}_r^{k_r}$, $\mathbf{X}_r^{k_r}$: current location and recent observations of robot-r; $\hat{\mathbf{X}}_{-r}^{k_r}$: planned waypoints of peers.
Output: $\mathbf{x}_r^{k_r+1}$: the next waypoint of robot-r.

```
1:  procedure TAKEDECISION($r, k_r, m, \Delta\theta$)
2:      if $k_r = 0$ then
3:          $\mathbf{x}_r^{k_r+1} \leftarrow$ TAKEFIRSTDECISION($r, k_r, \mathbf{x}_r^{k_r}, m, \Delta\theta$)
4:      else
5:          if Size of $\mathcal{D}_r^{1:k_r} > N_{\text{max}}$ then
6:              Down-sample $\mathcal{D}_r^{1:k_r}$ to $N_{\text{max}}$ observations
7:          $\alpha \leftarrow$ Update the exploitation weight using Alg. 2
8:          $GP_r \leftarrow$ Update Gaussian Model using $\mathbf{x}_r^{k_r}$ and $\hat{\mathbf{X}}_{-r}^{k_r}$
9:          $\bar{\mathbf{x}}_r^* \leftarrow$ Solve optimization problem ($GP_r$), Eq.(5)
10:         $\mathbf{x}_r^{k_r+1} \leftarrow$ Solve optimization problem ($GP_r$), Eq.(1)
11:     if $\|\mathbf{x}_r^{k_r+1} - \mathbf{x}_r^{k_r}\| < V/f_{\text{Hz}}$ then
12:         if $\|\bar{\mathbf{x}}_r^* - \mathbf{x}_r^{k_r}\| < 0.9\epsilon$ then      ▷ $\epsilon$: Detection range
13:             $\alpha \leftarrow 0$      ▷ Purely explorative search
14:             $\mathbf{x}_r^{k_r+1} \leftarrow$ Solve optimization problem, Eq.(1)
15:         else
16:             $r_1 \leftarrow \mathcal{U}(0,1)$
17:             $r_2 \leftarrow \mathcal{U}(0.1,0.9)$
18:             $\mathbf{x}_r^{k_r+1} \leftarrow \mathbf{x}_r^{k_r} + r_2 VT[\cos(2r_1\pi), \sin(2r_1\pi)]$
19:     $k_r \leftarrow k_r + 1$
20:     return $\mathbf{x}_r^{k_r}, k_r$
21: procedure TAKEFIRSTDECISION($r, m, \Delta\theta, V, T$)
22:     $l \leftarrow VT$
23:     if $\Delta\theta = 360$ then      ▷ $\Delta\theta$: Initial feasible dir. range
24:         $\theta \leftarrow r\Delta\theta/m$      ▷ $m$: Number of robots
25:     else
26:         $\theta \leftarrow r\Delta\theta/(m+1)$
27:     $\mathbf{x}_r^1 \leftarrow [l\cos\theta, l\sin\theta]$
28:     return $\mathbf{x}_r^1$,
```



Stop
Y
End Mission?
Y
N
Reached Waypoint?
N
Navigation (Path Tracking)
Measure Signal
Latest Observations $\mathcal{D}_r^{k_r}$
Start
$k_r = 0$
Plan First Waypoint
$\mathbf{x}_r^{k_r+1}$
k_r++
$\mathcal{D}_r^{k_r}$
$\mathbf{x}_r^1$
Receive Information
$\mathcal{D}_p^{k_p}$
Transfer Latest Observations
Perform Waypoint Planning by Solving Eq. (1)
$\mathbf{x}_r^{k_r+1}$
Broadcast Information
Observations $\mathcal{D}_r^{1:k_r}$
Downsample
Peers' Plans $\hat{\mathbf{X}}_{-r}^{k_r}$
$\mathbf{x}_p^{k_p}$
Update GP & Construct Acquisition Function
Source Seeking: Ω_r
Knowledge-Gain: Σ_r
Local Penalization: Γ_r
Waypoint Planning

Fig. 1 Flowchart of processes executed by each swarm robot running *Bayes-Swarm-P*: Here, computing processes are depicted by rectangular blocks, and data artifacts are depicted by curved-sided blocks; solid and dashed arrows respectively depict information flow between (and thus the sequence of) computing process, and the flow of recorded/received data and decisions.

the embodiment of the search process. To this end, the acquisition function must first consider balancing the following: i) explorative behavior – that reduces uncertainty in the swarm's knowledge of the signal environment; and ii) exploitative behavior – greedy behavior that gets the swarm robots closer to the expected signal source based on the current (albeit evolving) knowledge of the signal environment. In addition, due to the parallel nature of the search, the acquisition function must also mitigate interactions between the expected samples to be collected by the different robots as a result of their waypoint decisions. The proposed acquisition function thus contains an explorative and a scaled exploitative term, with their aggregate multiplied by a local penalizing factor, as given below:

$$\mathbf{x}_r^{k_r+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} (\alpha \cdot \Omega_r + (1-\alpha)\Sigma_r)\ \Gamma_r \tag{1}$$

s.t.

$$0 \le l = \|\mathbf{x} - \mathbf{x}_r^{k_r}\| \le VT \tag{2}$$

This acquisition function includes three key terms, which are defined as follows: *1) source seeking term,* $\Omega_r = \Omega_r(\mathbf{x}, \mathcal{D}^{1:k_r})$*:* leads robot-$r$ towards the location of the maximum signal strength expectation (promotes exploitation); *2) knowledge-uncertainty reducing term,* $\Sigma_r = \Sigma_r(\mathbf{x}, \mathcal{D}^{1:k_r}, \hat{\mathbf{X}}_{-r}^{k_r})$*:* minimizes the knowledge uncertainty of robot-r w.r.t. the signal's spatial distribution (promotes exploration); and *3) local penalty factor,* $\Gamma_r(\mathbf{x}, \bar{\mathbf{X}}_{-r}^{k_r})$*:* allows the mitigation of the interactions among the samples planned to be collected by robot-r and its peers; smaller value depicts larger penalty, since the objective function is defined in maximization terms. Sections 2.3-2.5 provide further description of how these terms are formulated differently from the standard acquisition functions used in BO, to account for the unique characteristics of ***embodied*** search.

Note that the key advancements here over the acquisition function in our original *Bayes-Swarm-O* algorithm [10], include the local penalty factor and an adaptive weighting to balance the contributions of exploration and exploitation. Comparison of this advanced *Bayes-Swarm-P* algorithm to the original *Bayes-Swarm-O* algorithm in our numerical experiments demonstrate the favorable impact of these key advancements to the acquisition function of Bayes-Swarm.

In Eq. (2), $\mathbf{x}_r^{k_r}$ represents the current location (waypoint) of robot-r at its k_r-th decision-step. Here, Eq. (2) constrains the length of robot-r's planned path (l) based on a set time-horizon (T) for reaching the next waypoint, and the nominal velocity (V) of robots. Through numerical trials it was found that for the case studies in this paper a time-horizon setting that allows a maximum separation between consecutive waypoints equal to roughly half the arena length worked well. However, the actual setting could also be treated as a user-defined parameter that is regulated to suit the problem at hand. In this regard note that, too high a value for time horizon could make the asynchronous parallel search less efficient (e.g., when new measurement knowledge becomes available from peers but remains unused for a robot traveling overly long distance before it can act on it); on the other hand, too low a value leads to wastage of computing resources by triggering waypoint planning instances too frequently. As a best practice, the time horizon should be set at a value between four times the inverse of the measurement frequency and half the arena length.

The coefficient $\alpha \in [0, 1]$ in Eq. (1) is the exploitation weight, i.e., $\alpha = 1$ leads to purely exploitative search behavior. Here, we design α to be adaptive in a way such that the swarm behavior is strongly explorative at the start and becomes more exploitative over waypoint iterations, i.e., as the mission progresses. This is achieved by setting $\alpha = 1/\big(1+\exp(-10(\frac{t}{T_{\text{max}}} - \frac{1}{3}))\big)$, where t is the time elapsed at that point in the mission and T_{max} is a user-prescribed maximum allowed mission time. This formula for examples varies α from ≈0.5 to ≈0.97 when the elapsed time is between 33% and 70% of the maximum allowed mission time. Through numerical experiments we also identified the need for further adjustments to varying α. Examples include setting α to low values to recover from an anticipated local minimum, and promoting greedy choices if a robot taking decision is notably closer (than a peer robot) to an expected source location reported and chosen as waypoint by that peer robot. The overall adaptive strategy to set the exploitation weight α is summarized in Algorithm 2.

Algorithm 2 Adaptive Exploitation Weight Strategy

Inputs: $\bar{\mathbf{x}}^*$: Expected source location of robot-r; $\hat{\mathbf{X}}^{k_r}_{-r}$: planned waypoints of peers; $\mathbf{P}_{\text{weaker}}$: Expected weaker source locations; l_{robot}: Size of robot (here set at 0.1m).

```
 1: procedure UPDATEEXPLOITATIONWEIGHT($\hat{\mathbf{X}}^{k_r}_{-r}, \mathbf{P}_{\text{weaker}}, l_{\text{robot}}$)
 2:     $d_{\text{weaker}} \leftarrow \min_{\mathbf{x}\in\mathbf{P}_{\text{weaker}}} \|\mathbf{x} - \bar{\mathbf{x}}^*\|^2$
 3:     if $d_{\text{weaker}} < 2l_{\text{robot}}$ then
 4:         $\alpha \leftarrow 0.1$
 5:     else
 6:         $d_{\text{best}} \leftarrow \min_{\mathbf{x}\in\hat{\mathbf{X}}^{k_r}_{-r}} \|\mathbf{x} - \bar{\mathbf{x}}^*\|^2$
 7:         if $d_{\text{best}} \geq 2l_{\text{robot}}$ then
 8:             $\alpha \leftarrow 0.8$
 9:         else
10:             $\alpha \leftarrow 1/\big(1+\exp(-10(\frac{t}{T_{\text{max}}} - \frac{1}{3}))\big)$
11:     return $\alpha$
```

2.3 Source Seeking (Exploitative) Term

Each robot models the signal's spatial distribution using a GP with squared exponential kernel (further description of this GP modeling is given in Appendix A). The GP model is updated based on the robot's own recent observations and those communicated by its peers, thereby leading to the following mean function:

$$\mu_r(\mathbf{x}) = \mu(\mathbf{x}, \mathbf{X}, \mathbf{y}) \tag{3}$$

Here $\mathbf{X}$ and $\mathbf{y}$ are respectively the observation locations and their corresponding signal measurements given in $\mathcal{D}^{1:k_r}_r$. Due to motion constraints, Eq. (2), a robot may not be able to reach the location, $\bar{\mathbf{x}}^*$, with the maximum expected signal strength (estimated using their GP model), within the time horizon. Therefore, the exploitative term (Ω_r) in Eq. (1) is re-defined to represent nearness to $\bar{\mathbf{x}}^*$, i.e., closer the better, and saturating at a maximum value of 1, as given by:

$$\Omega_r(\mathbf{x}, \mathcal{D}) = \frac{1}{1 + (\mathbf{x} - \bar{\mathbf{x}}^*)^T(\mathbf{x} - \bar{\mathbf{x}}^*)} \tag{4}$$

where

$$\bar{\mathbf{x}}^*_r = \arg\max_{\bar{\mathbf{x}}} \mu_r(\bar{\mathbf{x}}) \tag{5}$$

2.4 Knowledge-Uncertainty Reducing (Explorative) Term

Unlike in optimization, in robotic search, sampling is performed over the path of each agent. This concept is known as informative path planning, where robots decide their path such that useful information is collected. The explorative term in Eq. (1) (Σ_r) is designed to model the reduction in uncertainty in the robots' knowledge, thus facilitating informative path planning. To this end, the path of the robot is written in a parametric form as given below:

$$s(u) = u\mathbf{x} + (1-u)\mathbf{x}^{k_r}_r;\ u \in [0, 1] \tag{6}$$

where $\mathbf{x}^{k_r}_r$ is the current location of robot-r. While in this paper we consider obstacle-free search arenas, the above formulation can be readily extended to consider more complex paths in the presence of apriori-known obstacles.

In computing the self-reducible uncertainty in the posterior of robot-r, we account for the locations of both the past observations made by the robot and its peers, and the future observations to be made by robot-r's peers over the paths to their planned (immediate next) waypoints ($\bar{\mathbf{X}}^{k_r}_{-r}$) – both of these only consider what's currently known to robot-r via communication from its peers. The explorative term can thus be expressed as:

$$\Sigma_r(\mathbf{x}, \mathcal{D}^{1:k_r}, \bar{\mathbf{X}}^{k_r}_{-r}) = \int_{s(\mathbf{x})} \sigma_r(s(u))du \tag{7}$$

where $\sigma_r(\mathbf{x}) = \sigma(\mathbf{x}, \mathbf{X}^e_r)$ and $\mathbf{X}^e_r = \mathbf{X} \cup \bar{\mathbf{X}}^{k_r}_{-r}$. For further details on computing the mean (Eq. (4)) and the variance (Eq. (7)) of the GP, refer to Appendix A.

2.5 Local Penalizing Factor

For a batch-BO implementation, it is necessary to account for (and mitigate) the interaction between the batch of future samples, to preserve convergence [29]. Additionally, in swarm-robotic search, mitigating this interaction allows explicitly reducing the overlap in the planned knowledge gain by robots in the swarm – thereby promoting a more efficient search process. Modeling the interaction explicitly via predictive distribution, namely the *optimization-marginalization* approach, carries a significant computational overhead of $O(n^3)$ [29]. Simultaneous optimization of future candidate samples in the batch [35] also demands a strictly synchronized-distributed (or centralized) deployment of the swarm robots decision-making process, which as discussed earlier could be inefficient in an embodied search setting. An asynchronous distributed deployment is instead preferred here.

In the literature, there exist computationally tractable approximations to model the interactions. Specifically, we adopt the *marginalization-penalization* approach by González et al. [29]. They have shown that if the source signal is Lipschitz continuous, one can define a penalisation-based policy to collect a batch of points multiple steps ahead without having their (function) observations, such that the policy replicates results close to a sequential policy. Such a policy was observed to perform well in terms of the convergence to the optimum and rate of information gain [29].

In our adaptation of the penalization approach, we design a penalty factor, $\mathit{\Gamma}_r(\mathbf{x}, \hat{\mathbf{X}}^{p}_{-i})$ (smaller value depicts greater penalty), that enables local exclusion zones based on the Lipschitz properties of the signal's spatial function ($f(x)$). The multiplicative penalty factor (in Eq. (1)) thus tends to smoothly reduce the acquisition function in the neighborhood of the existing batch samples, i.e., the known planned waypoints of robot-r's peers ($\hat{\mathbf{X}}^{k_r}_{-r}$), with the signal observations at those points being not yet reported. To compute the penalty factor w.r.t. a given peer of robot-r, we define a ball $\mathcal{B}_r$ with radius ρ around that peer's planned waypoints, as given by:

$$\mathcal{B}_r(\mathbf{x}, \hat{\mathbf{x}}^{k_p}_p) = \{\mathbf{x} \in \mathcal{X} : \|\hat{\mathbf{x}}^{k_p}_p - \mathbf{x}\| \leq \rho\};\ \hat{\mathbf{x}}^{k_p}_p \in \hat{\mathbf{X}}^{k_r}_{-r} \quad (8)$$

The local penalty associated with a point $\mathbf{x}$ is then defined as the probability that $\mathbf{x}$ does not belong to the ball $\mathcal{B}_r$, i.e.,:

$$\gamma(\mathbf{x}, \hat{\mathbf{x}}^{k_p}_p) = 1 - P(\mathbf{x} \in \mathcal{B}_r(\mathbf{x}, \hat{\mathbf{x}}^{k_p}_p)) \quad (9)$$

We assume that the distribution of the ball radius ρ is Gaussian with mean $(M - \mu_r(\hat{\mathbf{x}}^{k_p}_p))/L$ and variance $\sigma^2_r(\hat{\mathbf{x}}^{k_p}_p)/L^2$. Here, $M = \max_{\mathbf{x}} f(\mathbf{x})$ is the maximum strength of the source signal and L is a valid Lipschitz constant ($\|f(\mathbf{x}_1) - f(\mathbf{x}_2)\| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|$). Both M and L can be in general set based on the knowledge of the application, and for batch-BO, Gonzalez et al. [29] showed that approximate values work quite well. The term M is defined as: $M = \max_x f(x)$, which can be reasonably assumed in many practical search applications based on the knowledge of the maximum possible source signal strength and/or the signal sensor's saturation level. Practically speaking, underestimation of "M" is quite unlikely, and in the event of overestimation of M, we will experience a larger ball radii and hence over-penalization of interactions, likely slowing down the search progress to some extent. The Lipschitz constant L can be estimated as follows: $L = \max_x \|\nabla f(x)\|$. Various numerical techniques [44,45] have been proposed in the literature to estimate the Lipschitz constant"L" based on (function) signal measurements. If the estimation of "L" is difficult or time-inefficient, it can be set to a large number. If the value is much larger than the (latent) actual value of L, then the local penalizing factor will tend to 1, and the behavior of *Bayes-Swarm-P* will tend to that of the *Bayes-Swarm-O* method. More detailed mathematical proof on the calculation of L and empirical results on estimating of L are provided in [29]. Here, we use a fixed value of L, since estimating L insitu would have added to the online computing burden with little impact on performance compared to the fixed value (which we checked on few trial runs) especially for larger swarm sizes.

Note that in most practical applications, the probability ($P(\mathbf{x} \in \mathcal{B}_r)$) will likely be very small w.r.t. peers that are far away from robot-r, and can thus be set to zero by using a tolerance threshold. With the respective assumed and estimated values of M and L, we can derive the following expression for the local penalty factor:

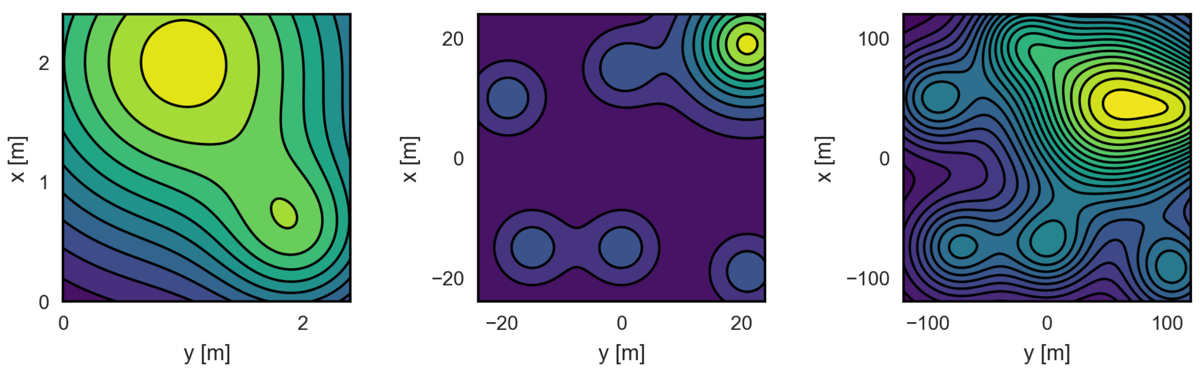
$$\begin{aligned}\gamma(\mathbf{x}, \hat{\mathbf{x}}^{k_p}_p) &= 1 - P(\|\hat{\mathbf{x}}^{k_p}_p - \mathbf{x}\| \leq \rho) \\ &= P(\mathcal{N}(0,1) \leq \frac{L\|\mathbf{x} - \hat{\mathbf{x}}^{k_p}_p\| - M + \mu_r(\hat{\mathbf{x}}^{k_p}_p)}{\sigma_r(\hat{\mathbf{x}}^{k_p}_p)}) \\ &= \frac{1}{2}\text{erfc}\left(-\frac{L\|\mathbf{x} - \hat{\mathbf{x}}^{k_p}_p\| - M + \mu_r(\hat{\mathbf{x}}^{k_p}_p)}{\sqrt{2\sigma^2_r(\hat{\mathbf{x}}^{k_p}_p)}}\right)\end{aligned} \quad (10)$$

Here, erfc(.) is the complementary error function, which is a differentiable function with an output bounded between 0 and 2. The effective penalty factor for the acquisition function of robot-r (accounting for interactions with all peers) is then determined from the net probability that the corresponding candidate waypoint $\mathbf{x}$ does not belong to the ball of any peers of robot-r, as given by:

$$\mathit{\Gamma}_r(\mathbf{x}, \hat{\mathbf{X}}^{k_r}_{-r}) = \prod_{p=1 \wedge p \neq r}^{m} \gamma(\mathbf{x}, \hat{\mathbf{x}}^{k_p}_p) \quad (11)$$

2.6 Information Sharing and Downsampling

While in swarm-robotic systems, a reliable inter-robot communication is desirable for sharing information, real-world applications present imperfect communication capabilities in terms of range, uptime and/or latency. To investigate the the performance of *Bayes-Swarm-P* under some degree of communication realism, here we consider the range limitation [46]. Specifically, we set the communication range at 2 m and 20 m for Case 1 and Case 2, respectively, with the cases being described in Section 3.2. This specification is akin to the simple disk model of wireless communication. In addition, considering the bandwidth limitations of ad-hoc



(a) Case 1: small arena, non-convex bimodal signal distribution (b) Case 2: large arena, multi-modal signal distribution (c) Case 3: very large arena, multi-modal signal distribution

Fig. 2 Three environment cases with different signal distributions.

wireless communication [47], we design our information sharing policy such that, along with asynchronous planning with fixed-time repetition, robots share only a compact set of observations. Table 9 provides a quick overview of the type and frequency of the information shared by each robot with all its peers across the swarm. Algorithm 3 lists two procedures that each robot uses to share or receive information. The term $\hat{X}^{k_r}_{-rp}$ is a 4-element array representing the current path of the p^{th} peer of robot-r, where the first two elements contain its current location coordinates, and the last two elements show the coordinates of the next (decided) waypoint that the p^{th} peer robot is moving towards.

Updating the GP models presents a cubic time complexity ($O(n^3)$) with respect to the size (n) of the data set. In order to keep the *Bayes-Swarm-P* algorithm computational tractable with increasing swarm size and mission duration, we need to downsample the collective data set of the swarm. Here we use a simple approach of sample rate compression by an integer factor d [48]. This approach reduces the data set by keeping the first sample and then every d-th sample after the first, where $d = \lceil \text{size}(\mathcal{D}^{1:k_r})/N_{\text{max}} \rceil$, where N_{max} is the active set size.

Algorithm 3 Information Packaging & Communication

```
1:  procedure RECEIVEINFORMATION($r, p, \mathbf{x}^{k_p}_p, \mathcal{D}^{k_p}_p$)
2:      $\mathcal{D}^{1:k_r}_r \leftarrow \mathcal{D}^{1:k_r}_r \bigcup \mathcal{D}^{k_p}_p$
3:      $\hat{X}^{k_r}_{-rp}(1:2) \leftarrow \hat{X}^{k_r}_{-rp}(3:4)$
4:      $\hat{X}^{k_r}_{-rp}(3:4) \leftarrow \mathbf{x}^{k_p}_p$
5:      return $\mathcal{D}^{1:k_r}_r, \hat{X}^{k_r}_{-rp}$
6:  procedure SENDINFORMATION($r, x^{k_r}_r, \mathcal{D}^{k_r}_r$)
7:      if $k_r = 0$ then
8:          Broadcast $\mathbf{x}^{k_r}_r$                         ▷ 4 bytes
9:      else
10:         Broadcast $\{\mathbf{x}^{k_r}_r; \mathcal{D}^{k_r}_r\}$          ▷ $4 + 6T$ bytes
```

3 Numerical Experiments & Case Studies

3.1 Distributed Virtual Implementation

In order to evaluate the *Bayes-Swarm-P* algorithm, we develop a discrete event-based swarm search simulation using Python 3 and PyBullet, and deployed this environment in a 4 core workstation (Intel® Xeon E5-1620 3.50 GHz, 4 cores processor). Each robot performs its behavior, as shown in Fig. 1, in parallel with respect to the rest of the swarm; i.e., updating its own knowledge model after each waypoint and deciding its next waypoint based on its own information and that received from its peers till that point. The robots share information with each other via a network class that assumed a limited range communication between robots. In addition, the developed simulator provides a graphical mode to show the mission based on PyBullet (a python based physics simulator) and supports both aerial and ground robots. The observation frequency (f_{Hz}) is set at 1 Hz. In order to maintain the computational cost of GP refitting tractable, the active set size (N_{max}) is set at 1,000.

3.2 Case Studies

To investigate the performance of the *Bayes-Swarm-P* method and perform comparative analysis, we design a set of numerical experiments and based on three distinct synthetic signal environments (Case 1, Case 2, and Case 3), which are discussed later. In addition, we demonstrate the effectiveness of the *Bayes-Swarm-P* method for potential real-world applications by applying it to a simulated avalanche search & rescue problem. This problem is described in Section 5.

Environments & Simulation Criteria: The two signal environments are shown in Fig. 2, and mathematically described in Appendix C. These cases respectively provide a bimodal spatial distribution over a small arena, and a complex multimodal spatial distribution over a larger arena. In order to evaluate and compare the results of the experiments in terms of a measure that is insensitive to the robots' speed and the arena size, we define a *relative completion time* (τ) metric. The relative completion time represents the search completion time (t_{achieved}) relative to the idealized completion time ($t_{\text{idealized}}$), as given by:

$$\tau = (t_{\text{achieved}} - t_{\text{idealized}}) / t_{\text{idealized}} \qquad (12)$$

Here, $t_{\text{idealized}}$ represents the time that a swarm robot would hypothetically take to directly traverse the straight-line path connecting the starting point and the signal source location.

For simulation termination purposes, two criteria are used. The first criterion terminates the search if any robot arrives within ϵ-vicinity of the signal source location. The second criterion terminates the simulated mission, if a maximum allowed search time (T_{max}) is reached.

Experiments: In total, we conduct six experiments. In *Experiment 1*, we benchmark various candidate optimizers to solve Eqs. (1)–(2) and Eq. (5) in *Bayes-Swarm-P*. In *Experiment 2*, we conduct a comparative analysis of *Bayes-Swarm-P*'s performance w.r.t. three other methods. In *Experiment 3*, we perform a scalability analysis to investigate the performance of *Bayes-Swarm-P* across multiple swarm sizes. In addition, we provide a time profiling for further insights on the computational complexity of *Bayes-Swarm-P*. In *Experiment 4*, we analyze the impact of the penalty term on search performance and computational cost. In *Experiment 5*, we analyze how partial information from peers impacts the performance of *Bayes-Swarm-P*. Finally, in *Experiment 6*, we study how the measurement noise and exploitation level of affect the performance of *Bayes-Swarm-P*. Detailed settings used for *Bayes-Swarm-P*, and the comparative algorithms are tabulated under Appendix D.

3.3 Demonstrating Bayes-Swarm-P: Case 1

Before delving deeper into the results of the five experiments (Section 4), here we provide an insightful illustration of how the proposed method works. Specifically, we show snapshots of the location of a team of 5 robots, as well as their traversed and planned paths, at different time points (Fig. 3) in the Case 1 environment (Fig. 2(a)). Alongside, we present how the knowledge (i.e., the GP-estimated expected value) of the signal's spatial distribution (μ_r) for a given robot-r differ from the actual signal distribution, and how this difference evolves over the mission. From Fig. 3(a), we can see that early on the accuracy of the robots' (belief) knowledge of the signal environment is limited (e.g., robot-1's $\mu_r(x)$ at time $t = 10^-$) leading to an expectation of the source location that is far away from the actual source. This is expected since at this point robot-1 has access to only 11 self observations. By $t = 20$ s, as seen from Fig. 3(c), robot-4 is able to construct a (GP-based) knowledge model of the signal environment that is fairly close to the actual signal distribution. Finally, by $t = 28$ s, majority of the robots get very close to the dominant signal source, with robot-4 getting within the set threshold (of 5 cm) that marks "target reached" (Figs. 3(d)).

Note that in our asynchronous implementation, communication occurs in a sequence among robots, and only at waypoints (and different robots might reach their k-th waypoint at a different time). This approach introduces non-homogeneity in the models of the environment across the swarm at any given time point. An example of this non-homogeneity is seen from the differences in the belief model of robot 1 (Fig. 3(a)) and robot 4 (Fig. 3(b)) around the 10s timepoint. In the future this issue could be addressed, while retaining the asynchronous benefits, by designing the communication schedule to be independent of the planning process, or by using a mixture of local GP models.

4 Results and Discussion

4.1 Experiment 1: Selection of Optimizers for *Bayes-Swarm-P*

In our proposed method, we have two optimization problems, i.e., Eqs. (1)–(2) and Eq. (5), that need to be solved in a nested manner. To choose a suitable optimization solver for each problem, we evaluate the performance of candidate solvers in terms of computing time (speed) and the quality of solutions, by running them over a set of scenarios. In this study, we consider a team of 5 robots in the Case 1 and Case 2 environments. In addition, to capture how each optimization solver performs as the belief or knowledge model ($\mu_r(.)$) evolves, we report the performance of the solver at two different decision-making instances over the same mission. In order to have a fair comparison, all conditions or instances of the problem should remain the same across the solvers being compared. Hence, we randomly chose two specific (decision-making) instances across which we will study the performance of the solvers. These instances represent the second waypoint-planning decision of robot-1 and the second waypoint-planning decision of robot-5. We run each solver using their suggested default settings.

The optimization problem to find the expected source location (Eq. (5)) is a maximization problem with a nonlinear objective function and side constraints (based on the given boundaries of the search domain). For this problem, we consider three optimization methods that are suitable for this type of optimization problems and has publicly available Python implementations, namely: (1) *L-BFGS-B* [49]: a Limited memory quasi-Newton method of Broyden, Fletcher, Goldfarb, and Shanno with Bound-Constraints (SciPy's implementation [50]); (2) *TNC* [51]: a truncated Newton method (SciPy's implementation); and

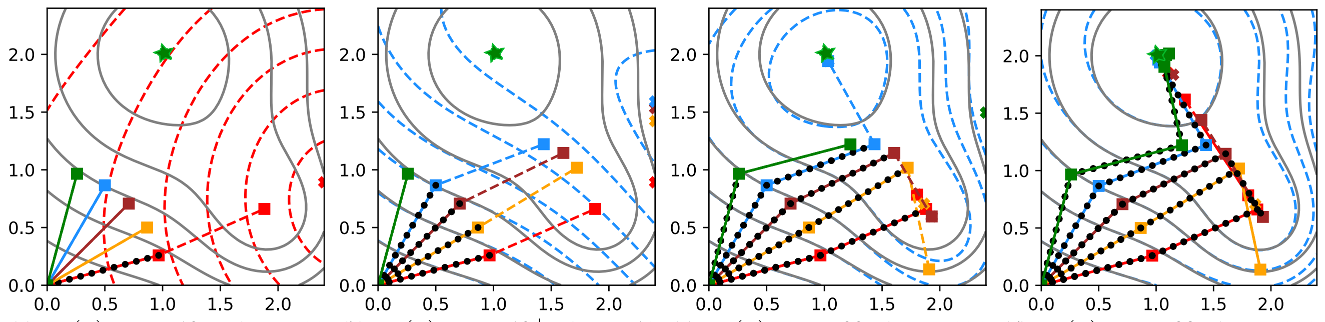

(a) $\mu_1(\mathbf{x})$ at $t = 10^{-}$s ($n = 11$ samples) (b) $\mu_4(\mathbf{x})$ at $t = 10^{+}$s ($n = 41$ samples) (c) $\mu_4(\mathbf{x})$ at $t = 20$s ($n = 136$ samples) (d) $\mu_4(\mathbf{x})$ at $t = 28$s ($n = 220$ samples)

Fig. 3 Experiment 1, Case 1: knowledge state and robot path snapshots. These figures show robot paths. The squares show robots' waypoints. Straight solid lines depict travelled paths. Straight dashed lines depict planned paths. Each robot has been assigned a unique color to distinguish its trajectory, which is respectively red, orange, brown, blue, and green for robots 1-5. The black dots show the observations (samples) that have been used by the robot to build its own belief model. Gray solid contours represent the actual signal distribution and dashed contours represent the current signal distribution model of the stated robot. The green star shows the source location.

Table 1 Performance of optimizers in *Bayes-Swarm-P* for solving the inner problem given by Eq. (5) for a 5-robot scenario on Case 1 and Case 2.

#	Env.	Method	Performance		
			t_{OPT} [sec]	$\mathbf{x}^*$	f^*
1	Case 1	L-BFGS-B	0.004	[2.4, 0.89]	1.07
		TNC	0.029	[2.4, 0.89]	1.07
		PSO	0.286	[2.4, 0.89]	1.07
2	Case 1	L-BFGS-B	0.004	[2.4, 1.5]	1.21
		TNC	0.024	[2.4, 1.5]	1.21
		PSO	0.198	[2.4, 1.5]	1.21
3	Case 2	L-BFGS-B	0.014	[6.3, 19.39]	0.53
		TNC	0.075	[6.3, 19.39]	0.53
		PSO	0.467	[6.3, 19.39]	0.53
4	Case 2	L-BFGS-B	0.006	[6.41, 18.85]	0.53
		TNC	0.036	[6.41, 18.85]	0.53
		PSO	0.362	[6.41, 18.85]	0.53

t_{OPT}: time taken to solve the optimization problem at the given instance.

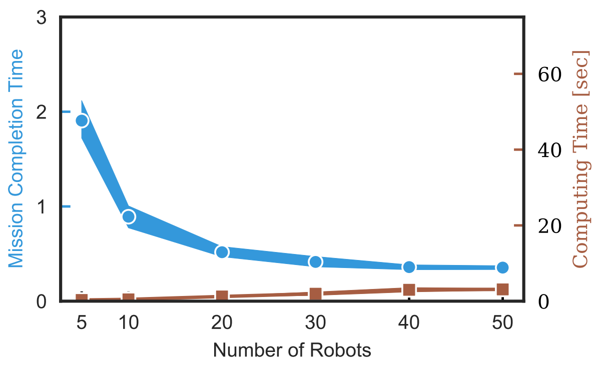
(3) *PSO* [52]: the gradient-free Particle Swarm Optimization method (the pyswarm library [53]). Note that, a randomized initial guess (or population if it's PSO) is used with each solver when solving Eq. (5), which was found to fare better than simply using the current point of the concerned robot as the initial guess for that optimization. Hence, the results are stochastic (across multiple runs of *Bayes-Swarm-P*) even with the solver choices 1 and 2. Table 1 shows that while all three methods find the same optimum point, L-BFGS-B is up to 7 and 71 times faster than TNC and PSO, respectively. Based on these results, we selected the L-BFGS-B method as the default optimizer for the inner optimization problem (Eq. (5)) of *Bayes-Swarm-P*.

The outer optimization problem, Eqs. (1)–(2), involves a nonlinear objective function to be maximized, an inequality constraint and side constraints. For this type of problem, we compare and contrast the following four optimization solvers: (1) *COBYLA* [54]: a direct search method; (2) *SLSQP* [55]: a sequential square programming method; (3) *Trust-Cons* [56]: a trust-region interior point method; and (4) *PSO*: same one as described earlier. For the first three methods, we use the implementations that are provided in the SciPy library. As can be seen from Table 2, PSO and COBYLA are the top choices in terms of finding the optimal waypoint with the highest acquisition function value, which is likely due to the multimodal nature of the acquisition function in this case. The COBYLA method is however up to 170 times faster than the PSO method.

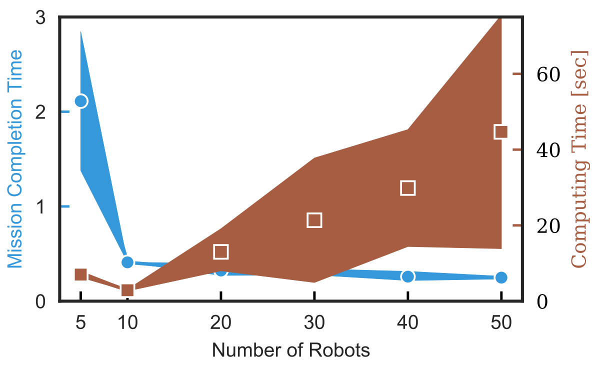
In summary, for solving the inner and outer optimizations in *Bayes-Swarm-P*, PSO performs the best in terms of finding the optimum, while lagging significantly behind the COBYLA + L-BFGS-B choice in terms of computational efficiency. Both of these choices are thus applied in the remaining experiments.

Table 2 Performance of optimizers in *Bayes-Swarm-P* for solving the outer problem given by Eq. (1)–(2) on Case 1 and 2.

Inst.	Env.	Method	Performance		
			t_{OPT} [sec]	$\mathbf{x}^*$	f^*
1	Case 1	COBYLA	0.054	[1.88, 0.66]	0.61
		SLSQP	0.064	[1.89, 0.64]	0.61
		Trust-Constr	0.807	[1.87, 0.62]	0.59
		PSO	6.652	[1.88, 0.66]	0.61
2	Case 1	COBYLA	0.045	[1.23, 1.21]	0.33
		SLSQP	0.070	[1.26, 1.00]	0.31
		Trust-Constr	0.694	[0.43, 0.94]	0.15
		PSO	5.355	[1.23, 1.19]	0.33
3	Case 2	COBYLA	0.062	[6.18, 19.02]	0.70
		SLSQP	0.039	[3.45, 10.61]	0.01
		Trust-Constr	0.419	[6.18, 19.02]	0.70
		PSO	10.539	[6.17, 19.02]	0.70
4	Case 2	COBYLA	0.091	[6.47, 8.84]	0.00
		SLSQP	0.006	[0.55, 16.89]	0.00
		Trust-Constr	0.703	[0.07, 9.47]	0.00
		PSO	10.613	[-9.45, 16.78]	0.01



(a) *Bayes-Swarm-P* with COBYLA and L-BFGS-B for solving Eq. (1) and Eq. (5), respectively.



(b) *Bayes-Swarm-P* with PSO for solving both Eq. (1) and Eq. (5).

Fig. 4 Scalability analysis of *Bayes-Swarm-P*: Variation in performance metrics (completion time (τ) and computing time) with swarm sizes changing from 5 to 50 for Case 2. The computing times are in terms of the median of computing time per waypoint planning of the robot that reached the target. The line and colored region in both figures represent the mean and 95% confidence interval, respectively.

4.2 Experiment 2: Comparative Analysis of Bayes-Swarm-P

Table 3 summarizes the comparison of *Bayes-Swarm-P* to three other methods in terms of completion time (τ). The three other methods include: ***1)*** *Glowworm Swarm Search:* a state-of-the-art search method based on swarm intelligence [26,42], which has been successfully used for robotic search applications [42,57,58]; We use an implementation of the Glowworm algorithm available at [42]; ***2)*** *Levy Walk Search:* the Levy walk search method [59,43] is a bio-inspired search strategy that has been used for the localization of sources, such as chemical plume; and ***3)*** *Exhaustive Search:* a baseline exhaustive search method [60], which divides the search area into equal partitions, and sends a robot to each partition to perform a simple sweeping pattern on the area. It should be noted that the Glowworm algorithm assumes the robots to be initially distributed in the search arena. To address this requirement and have a fair comparison, we run the Glowworm algorithm after executing the `takeFirstDecision` procedure in Algorithm 1. In this experiment, *Bayes-Swarm-P* is run 55 times and other comparative methods are run 5 times on each environment case with the same number of robots (i.e., 5 and 10 robots for Case 1, and Case 2, respectively), and the performance is reported in Table 3 in terms of (mean $\pm$ std-dev of) completion time.

Table 3 Comparative analysis of *Bayes-Swarm-P* on Case 1 and Case 2 with a swarm of 5 and 10 robots, respectively. The PSO solver option is used for *Bayes-Swarm-P*.

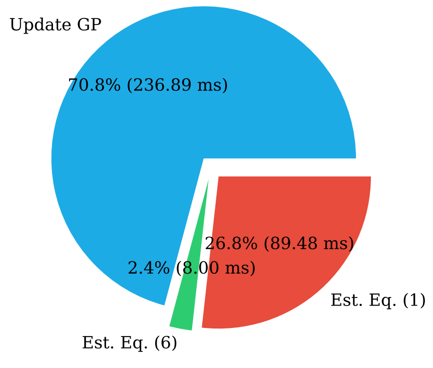
Case	Algorithm	Completion Time (τ)
1	Bayes-Swarm-P	0.34 ± 0.02
	Glowworm Swarm Search	1.92 ± 0.75
	Levy Walk Search	$^*10.44 \pm 9.39$
	Exhaustive Search	$^\dagger 11.16 \pm 0.00$
2	Bayes-Swarm-P	0.39 ± 0.00
	Glowworm Swarm Search	1.47 ± 0.18
	Levy Walk Search	$^*9.16 \pm 11.90$
	Exhaustive Search	$^\dagger 42.14 \pm 0.00$

The maximum allowed search time, the idealized time, the robot velocity, and the Lipschitz constant are set for both cases as follows: Case 1: $T_{\text{max}} = 100$s, $T_{\text{idealized}} = 22.36$s, $T = 10$s, $V = 0.1$m/s, $M = 1.2$, $L = 2$, $\epsilon = 0.05$m; Case 2: $T_{\text{max}} = 1,000$s, $T_{\text{idealized}} = 28.3$s, $V = 1$m/s, $T = 20$s, $M = 1.2$, $L = 100$, $\epsilon = 0.2$m. * Levy walk search fails to find the target in several sample runs, and the results here represent that of 5 best runs among roughly a total of 9-13 runs. † Exhaustive search is conducted in parallel by 10 robots (3 separately in the 1st and 3rd quarters, and 2 separately in the other 2 quarters of the arena).

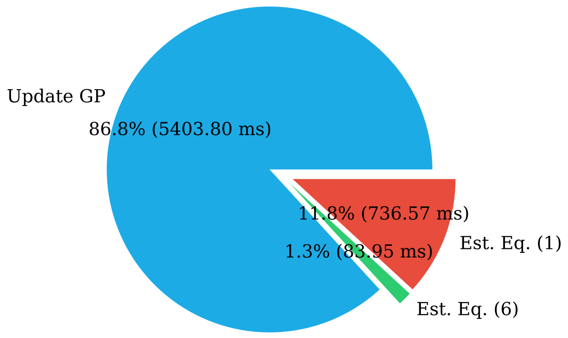
It can be seen from Table 3 that *Bayes-Swarm-P* outperforms Levy walk and exhaustive search by almost two orders of magnitude better search efficiency (i.e., smaller completion time) in both cases. *Bayes-Swarm-P* also provides roughly five and four times better search efficiency compared to the Glowworm swarm search method on Case 1 and 2, respectively. For the Glowworm method, we set the observation frequency at 100. Hence, the Glowworm method uses 100 times more samples than our *Bayes-Swarm-P* method. It should be noted that the *Bayes-Swarm-P* method can operate with lower observation frequency and longer time horizon, since it builds a knowledge model that helps the robots to extrapolate. In contrast, the Glowworm swarm and Levy walk search methods need to operate on shorter decision time horizon, and thus require higher observation frequency.

4.3 Experiment 3: Scalability Analysis of Bayes-Swarm-P

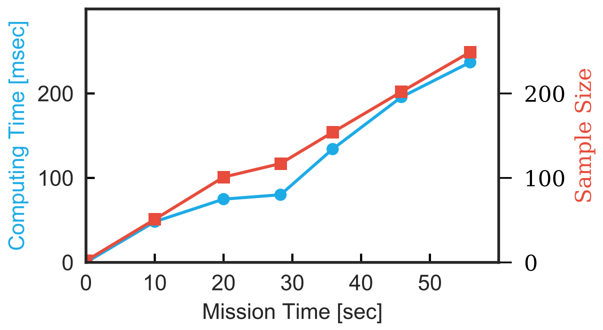
We run *Bayes-Swarm-P* simulations on Case 2 with swarm sizes varying from 5 to 50. Both optimization solver choices that we converged upon in Section 4.1 are considered here,



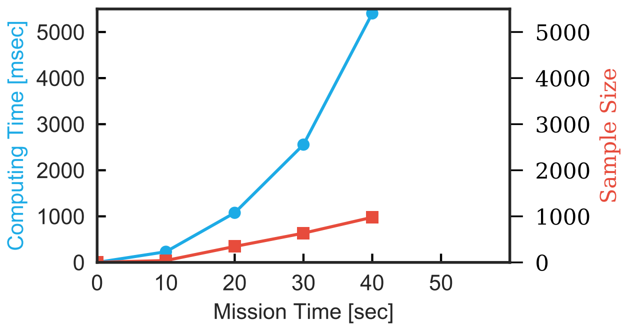
(a) Case 2 with 5 robots: time profiling of *Bayes-Swarm-P* for its most expensive decision-step.



(b) Case 2 with 30 robots: time profiling of *Bayes-Swarm-P* for its most expensive decision-step.



(c) Case 2 with 5 robots: computational cost and sample size of the GP model updating plotted versus the progressive (simulated) physical time.



(d) Case 2 with 30 robots: computational cost and sample size of the GP model updating plotted versus the progressive (simulated) physical time.

Fig. 5 Scalability analysis and time profiling of *Bayes-Swarm-P* for the first robot that reaches the source on Case 2 with 5-robot and 30-robot swarms. The COBYLA/L-BFGS-B optimizer is used for *Bayes-Swarm-P*.

namely: *Choice 1*: COBYLA for Eq. (1)–(2) and L-BFGS-B for Eq. (5), which gives the best performance in terms of computational efficiency; and *Choice 2*: PSO for both optimizations, which achieves the best performance in terms of finding or getting close to the optimum, and might be preferable for smaller swarm sizes and/or when sufficient onboard computing power is available. Figure 4 illustrates the results of this study in terms of the relative completion time (τ) and computing time for both choices – namely, results of Choice 1 and Choice 2 are shown in Fig. 4(a) and Fig. 4(b), respectively. Both metrics for Choice 1 and Choice 2 are respectively estimated over and 55 runs for each swarm-size scenario. Here, the computing time is measured in terms of the median "*computing time per planning instance*" across all waypoint decisions, estimated for the robot that reaches the signal source. It can be seen that the variance in the computing time for *Bayes-Swarm-P* with PSO is much larger than that of *Bayes-Swarm-P* with "COBYLA"-"L-BFGS-B". This observation is expected since due the stochastic nature of PSO, it is known to require largely varying number of iterations to converge to the same relative change (tolerance) criteria across different runs [61], leading to large variation in the cost for solving the optimizations in Eqs. 1-2 and 5.

The number of robots that is required to perform an efficient search for a given fixed environment depends on multiple factors, such as: 1) the size of the environment, 2) the complexity of the source signal, and 3) robots' capabilities (speed, sensing range, communication range, etc.). The scalability analysis for Case 2 (Fig. 4) can be used to find the minimum swarm size. As shown in Fig. 4, the mission completion time quickly reduces as the swarm sizes increases to 20 robots, and then saturates due to a diminishing marginal utility (given the fixed size of the search space). Performance of *Bayes-Swarm-P* is observed to be slightly better with the *choice 2* optimizer (PSO), in terms of mean and variance of the completion time across 5 runs. As the swarm size grows, the rate of collective sample measurements increases, which is expected to cause a cubic increase in the cost of updating the GP-based belief model by each robot for any given t-th waypoint planning instance. It is however interesting to notice (from Fig. 4) that with both optimization choices, the computing time increases linearly with increase in swarm size (albeit the linear-rate being higher with

PSO, which is expected). This observation can be partly attributed to the reduction in the number of waypoint planning instances caused by the decrease in completion time (and mission length thereof) as the swarm size grows. Considering that the computing cost of each waypoint planning depends on multiple process, including updating the GP and two optimizations (Eqs. (1) and (5)), dedicated time profiling is needed to get further insights into the real-time performance of *Bayes-Swarm-P*, as discussed next.

***Time Profiling of Bayes-Swarm-P*:** Figures 5(a) and 5(b) show the time profiling for the last waypoint planning instance of the robot that is found to be the first to find the signal source for Case 2, respectively in the 5 robot and 30 robot swarm scenarios. For both scenarios, the belief (GP) updating process is the dominant contributor to the computing time ($71\% - 87\%$), while solving the inner optimization (Eq. (5)) has the lowest computing cost (1.3–2.4%, roughly 8-84 milliseconds). These results show that the main bottleneck for scaling up the *Bayes-Swarm-P* method is the GP model updating. Figures 5(c) and 5(d) show how the computing time and associated sample size in updating the belief (GP) model for the first-to-find-the-source robot vary over the Case 2 mission, respectively with 5-robot and 30-robot swarms. Note that, in Case 2 with 30 robots (Fig. 5(d)), the available sample size of the robot at its last decision-making instance is 3,350, but a downsampled set of 1,000 observations are used to train the GP model.

Swarm-scale Case Study: As we observed above, Case 2 can be effectively searched using a team of 20 robots given the associated arena size and signal distribution complexity. Hence, we conduct an additional study to demonstrate the benefits of using a larger swarm of robots, and *Bayes-Swarm-P*'s effectiveness in handling such large swarms. For this purpose, Case 3 is introduced and a team of 48 and 100 robots are used to perform the search. Case 3 is shown in Fig. 2(c) and defined in Appendix C. In this case study, to take advantage of initial exploration, we use four starting points that lie at the four respective corners of the arena. The results are summarized in Table 4. It can be seen that a swarm with 100 robots is more effective in term of the completion time, compared to the 48-robot swarm – the 100-robot swarm finds the target on average 5% faster than the 48-robot swarm (with a p-value ≤ 0.05).

Table 4 Swarm-Scale Study: *Bayes-Swarm-P* for Case 3. For running *Bayes-Swarm-P*, the following settings have been used for Case 3: $T_{\max} = 1000$s, $T = 100$s, $V = 1$m/s, $M = 1.2$, $L = 480$, and $\epsilon = 0.2$m. The COBYLA/L-BFGS-B optimizer has been used. The completion time (τ) is reported in mean±std-dev.

Swarm Size	Completion Time (τ)
48	5.18 ± 0.10
100	4.93 ± 0.11

4.4 Experiment 4: Analysing the Benefits of the Penalty Term in *Bayes-Swarm-P*

Table 5 summarizes the completion-time performance (τ) of the *Bayes-Swarm-P* method with and without the penalty term (Γ_r in Eq. (1)). To generate the most optimal results, the PSO solver is used and each case is run for 10 times. With the new penalty term (i.e., with *Bayes-Swarm-P*) the location of the global source is found 1.2 and 1.8 times faster, respectively in Case 1 and 2, than that without the penalty term (i.e., with original *Bayes-Swarm-O*). The penalty term in *Bayes-Swarm-P* is observed to also result in reduced variance, thus greater robustness, in performance across multiple runs, as seen from the results in Table 5. This is attributed to the ability of the penalty factor to mitigate interactions, and in that process limit the local search space in which the next waypoint of a given robot can be planned in an informed manner. While demonstrating the benefits of using penalization, these observations also point to the potential of the penalty term to be particularly helpful with more complex signal distributions (e.g., Case 2 vs. Case 1).

Table 5 Performance with the penalty term (*Bayes-Swarm-P*) vs. without the penalty term (*Bayes-Swarm-O*), in 5-robot and 10-robot swarms, applied on Case 1 and 2. The completion time (τ) is reported in mean±std-dev, over 5 runs. The PSO solver has been used to get most optimal results.

Case	Algorithm	Completion Time (τ)
1	*Bayes-Swarm-P*	0.34 ± 0.02
	Bayes-Swarm-O	0.35 ± 0.06
2	*Bayes-Swarm-P*	0.39 ± 0.00
	Bayes-Swarm-O	0.75 ± 0.42

For further analysis of the beneficial influence of the penalty term, for a representative run, we record the convergence history of *Bayes-Swarm-O*; specifically we measure the best value of the signal (across the swarm) and the distance to the target source over mission time spent. Note that the initial iteration of waypoint planning (where Eq. (1) is not used) is the same in both versions of the algorithm. Figure 6 show the results of this study on Case 2 with a 10-robot swarm. As it can be seen from Fig. 6, *Bayes-Swarm-P* and *Bayes-Swarm-O* initially observed locations with the same level of signal values. But after $t = 20$, *Bayes-Swarm-P* outperformed *Bayes-Swarm-O* by finding locations with larger signal values, and eventually finding the source 42% sooner. These characteristics are further corroborated by the observations (from Fig. 6) of how *Bayes-Swarm-P* leads to convergence of at least one swarm robot on the target source by ∼31.7s, while the robots driven by *Bayes-Swarm-O* experience oscillations before finally reaching the target at ∼55s.

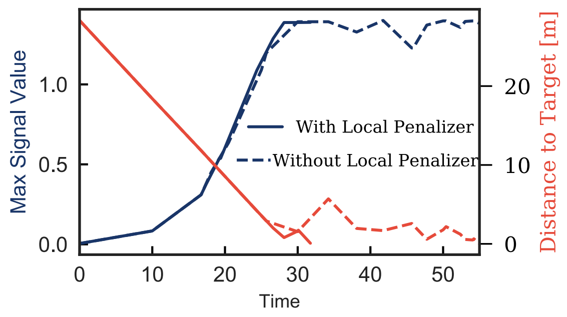


Fig. 6 Experiment 4, Case 2: Convergence history in terms of both largest measured signal and the minimum distance to the target over the time mission.

4.5 Experiment 5: Analysis of Impact of Partial (Peer) Observation on *Bayes-Swarm-P*

Here we analyze how *Bayes-Swarm-P* is impacted by an imperfect communication network, which leads to partial observability of peers' signal measurements. This study is done for Case 1 and 2 with 5 and 10 robots, respectively. We assume the following communication ranges for each case: (i) Case 1: 0.1 m, 0.5 m, and 1 m; (ii) Case 2: 5 m, 10 m, and 20 m. Each scenario (case and the communication range) is run 55 times and the results are compared in Table 6, including the baseline case with perfect communication (i.e., w/o range limitation). As seen from the table, range-limited communication has different impact on the swarm search efficiency in Case 1 and Case 2. In Case 1, the swarm search efficiency increases when the communication range is limited. This is likely due to the simplicity of the signal distribution in Case 1, which can be successfully searched with high exploitation (i.e., a greedy search, with less collaboration and exploration). On the other hand, the swarm in Case 2 takes progressively more time to find the location of the source signal (i.e., registers larger completion time), when the communication range is increasingly constrained. This performance loss in Case 2 demonstrates the importance of communication between robots when dealing with more complex signal distributions. While an approach for making probabilistic estimations on the waypoint decisions of out-of-range peers could be explored in future to address this performance loss under partial (peer) observation, this is expected to present computational challenges as the swarm size increases.

4.6 Experiment 6: Analysis of Impact of Noise on *Bayes-Swarm-P*

In this section, we evaluate how the measurement noise impacts the performance of *Bayes-Swarm-P*. Since higher noise-to-signal ratio is expected to make exploitative behavior less reliable, we also study how the effect of the exploitation weight α is dependent on the noise levels. For this purpose, we run *Bayes-Swarm-P* simulations on Case 1 and Case 2 with 5 and 10 robots, respectively. For each case, we run *Bayes-Swarm-P* with four levels of noise (i.e., $\sigma = \{0, \frac{\|A\|}{50}, \frac{\|A\|}{20}, \frac{\|A\|}{10}\}$) where A is the measured signal magnitude; and two levels of exploitation weighting ($T_{\max} = \{100, 1000\}$) are also considered. For the measurement noise, we assume a white Gaussian noise ($\mathcal{N}(0, \sigma^2)$). In order to achieve different levels of exploitation, we change the $T_{\max}$ in computing of α, where larger value of $T_{\max}$ decreases exploitation tendency. The results are summarized in Table 7.

Table 6 Impact of partial peer observation on *Bayes-Swarm-P* performance, for the following scenarios: Case 1 with 5 robots and Case 2 with 10 robots. The completion time (τ) is reported in mean±std-dev, over 55 runs. The COBYLA/L-BFGS-B optimizer has been used.

Case	Communication	Completion Time (τ)
1	No Limitation	0.44 ± 0.10
	Limited Range (1 m)	0.05 ± 0.01
	Limited Range (0.5 m)	0.05 ± 0.01
	Limited Range (0.1 m)	0.05 ± 0.01
2	No Limitation	0.89 ± 0.45
	Limited Range (20 m)	0.96 ± 0.70
	Limited Range (10 m)	1.26 ± 0.49
	Limited Range (5 m)	1.34 ± 0.78

It can be seen from Table 7 that increasing the level of noise decreases the search efficiency in terms of the completion time, for both Case 1 and Case 2. The most significant reduction in performance, namely 2 times longer completion time, occurs when the noise changes from $\|A\|/50$ to $\|A\|/20$. When we compare the performance of *Bayes-Swarm-P* in the noise-free condition with the large noise condition ($\|A\|/10$), there is a statistically highly significant evidence (p-value < 0.001) to support the detrimental impact of signal noise on search efficiency. Interestingly, decreasing exploitative behavior (by increasing $T_{\max}$) led to a slight increase in completion time under all noise levels in Case 1, and mixed results under Case 2. Hence, at this stage it remains challenging to conclude a clear inter-relation between signal noise and exploitation behavior, agnostic of the signal complexity.

5 Application: Avalanche Search & Rescue with Multi-UAV Teams

5.1 Problem Description & Numerical Settings

The popular sport of backcountry skiing brings the risk of getting buried under an avalanche. Currently, backcountry skiers use beacons to facilitate getting found, if they encounter an avalanche accident. It has been shown that more than 90% of people buried by avalanches survive if found and dug out within 15 minutes; unfortunately, after 45 min-

Table 7 Analysis of Impact of Noise on *Bayes-Swarm-P* for Case 1 with 5 robots. The completion time (τ) is reported in mean±std-dev, over 55 runs. Smaller value of $T_{\max}$ decreases the level of exploitation in *Bayes-Swarm-P*. The COBYLA/L-BFGS-B optimizer has been used.

Case	Noise (σ)	$T_{\max}$	Completion Time (τ)
1	0	100	0.44 ± 0.10
		1,000	0.52 ± 0.13
	$\frac{\|A\|}{50}$	100	0.51 ± 0.35
		1,000	0.84 ± 0.48
	$\frac{\|A\|}{20}$	100	1.07 ± 0.55
		1,000	1.35 ± 0.70
	$\frac{\|A\|}{10}$	100	1.22 ± 0.71
		1,000	1.64 ± 0.79
2	0	100	0.83 ± 0.49
		1,000	0.89 ± 0.45
	$\frac{\|A\|}{50}$	100	0.97 ± 0.42
		1,000	0.83 ± 0.44
	$\frac{\|A\|}{20}$	100	1.58 ± 0.71
		1,000	1.54 ± 0.81
	$\frac{\|A\|}{10}$	100	2.58 ± 2.13
		1,000	2.76 ± 2.15

utes, the odds of survival drop to about 20% due to risks of hypothermia and suffocation [62]. As a part of the European project, Smart collaboration between Humans and ground-aErial Robots for imProving rescuing activities in Alpine environments (SHERPA) [2], UAVs have been shown to serve as an effective partner in such rescue operations [3]. Utilizing UAVs to locate the victim can decrease the search time by at least 50%, and thus increase the chances of survival [11,3]. Not only can UAV's travel faster than skiers when searching, but they also do not have to worry about the terrain and steepness of the mountain unlike skiers. It is conceivable that using a team of small UAVs can provide even faster search times compared to single units, especially if guided by an efficient collective search method. To explore this concept, here we develop a discrete event simulation of the process of searching for signal source (i.e., beacon on victim) trapped under an avalanche. A high-level illustrative description of how the manned mission progresses vs. how a team of UAVs would conceivably operate is provided in Fig. 9 in the Appendix.

Avalanche Beacon Signal: To model the spatial distribution of the beacon's electromagnetic signal strength (H), we use the model given by [63], where H is expressed as a function of distance ($\mathbf{r} = [r_x, r_y, r_z]^T$) and the magnetic moment of the transmitter ($\mathbf{m}$), as given by:

$$\mathbf{H} = \frac{1}{4\pi r^5}\mathbf{A}\mathbf{m} \tag{13}$$

where

$$\mathbf{A} = \begin{bmatrix} 2r_x^2 - r_y^2 - r_z^2 & 3r_x r_y & 3r_x r_z \\ 3r_x r_y & 2r_y^2 - r_x^2 - r_z^2 & 3r_y r_z \\ 3r_x r_z & 3r_y r_z & 2r_z^2 - r_x^2 - r_y^2 \end{bmatrix} \tag{14}$$

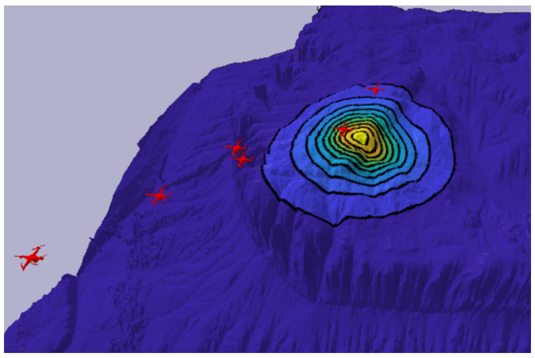


Fig. 7 A screenshot of the Pybullet-based avalanche environment.

Environment Settings & Assumed UAV Platform: We use Pybullet and a mountain object to simulate the mountain environment. Figure 7 shows a screenshot of the environment with 6 UAVs. Here, we assume the use of a small UAV with 25 min flight time, 1 km flight range, and 0.5 m/s operational speed. We set this speed based on a physical experiment conducted by [11] for this application, which ensures appropriate measurement of the beacon signal. We consider a 40×40 sq.m search environment, given the limited range within which the beacon signal can be registered by the receiver. In practice, the team of UAVs might start from a depot that is farther apart, and initiates a scouting process that divides the overall larger search area into partitions, with *Bayes-Swarm-P* (or Glowworm, Levy walk etc.) taking over once any UAV registers a non-zero signal measurement. In our experiments, this initial scouting process was not used, solely to keep the overall simulation costs low. We also consider the possibility of experiencing measurement noise, and present results where the H measurements have a noise of $\mathcal{N}(0, \frac{\|H_{\max}\|^2}{100^2})$.

5.2 Results

We use a 6-UAV team to perform the search, with the beacon located at (20,20). We run the simulation using four methods, namely, *Bayes-Swarm-P*, Glowworm swarm, Levy walk, and exhaustive search. Due to the inherent stochastic nature of each of the search methods, they are run 5 times on the same search & rescue mission scenario. Table 8 summarizes the results in term of normalized completion time, τ (in mean±std over 5 runs). Here *Bayes-Swarm-P* comes out to be a clear winner. The results show that *Bayes-Swarm-P* outperforms Levy walk and exhaustive search by providing two orders of magnitude smaller completion time. The completion time (τ) of *Bayes-Swarm-P* (with different α) is also observed to be approximately 1/5 to 1/3 of that of Glowworm search.

Based on the results in Table 8, the mean value of the actual completion times in minutes roughly translate to 1.1 and 1.5 mins, respectively for *Bayes-Swarm-P* and Glowworm. Thus, *Bayes-Swarm-P* and Glowworm swarm search are the only two approaches that are able to find victims within the practically desirable time frame of < 5 mins. We further compare the performance of these two methods under measurement noise, both in terms of completion time and communication/planning-load. While the performance of both methods become worse under measurement noise, *Bayes-Swarm-P* is less affected compared to Glowworm search. As seen from Table 8, under noise, the completion time of *Bayes-Swarm-P* remains an order of magnitude smaller than that of Glowworm (these roughly 1.6 mins vs. 9.7 min in actual time). Moreover, Glowworm incurs a much higher planning and communication load (357 planning and information exchange instances per robot) compared to *Bayes-Swarm-P* (5 planning and information exchange instances per robot) in achieving these completion performance. These results point to the real-world performance benefits of *Bayes-Swarm-P*.

Table 8 Comparative analysis of *Bayes-Swarm-P* on the avalanche problem with a team of 6 robots. The COBYLA/L-BFGS-B optimizer has been used for *Bayes-Swarm-P*.

Noise $\mathcal{N}(0,\sigma^2)$	**Algorithm**	**Completion Time** (τ)
-	*Bayes-Swarm-P*	0.19 ± 0.09
	Bayes-Swarm-P ($\alpha = 1$)	0.12 ± 0.03
	Glowworm Swarm Search	0.62 ± 0.1
	Levy Walk Search	$^*33.06 \pm 19.99$
	Exhaustive Search	$^\dagger 14.09 \pm 0.00$
$\mathcal{N}(0, \frac{\lVert H_{\max}\rVert^2}{100^2})$	*Bayes-Swarm-P*	0.66 ± 0.47
	Glowworm Swarm Search	9.30 ± 0.86

The idealized time, the robot velocity, and the Lipschitz constant are set as follows: $T_{\max} = 100$s, $T_{\text{idealized}} = 56.57$s, $T = 10$s, $V = 0.5$m/s, $L = 2$, $\epsilon = 0.15$m. The observation frequency for *Bayes-Swarm-P* is set at 0.5Hz. The maximum value of signal is $\lVert H_{\max}\rVert = 47$. * Levy walk search fails to find the target in several sample runs, and the results here represent that of 5 best runs among roughly a total of 9-13 runs. † Exhaustive search is conducted in parallel by 6 robots.

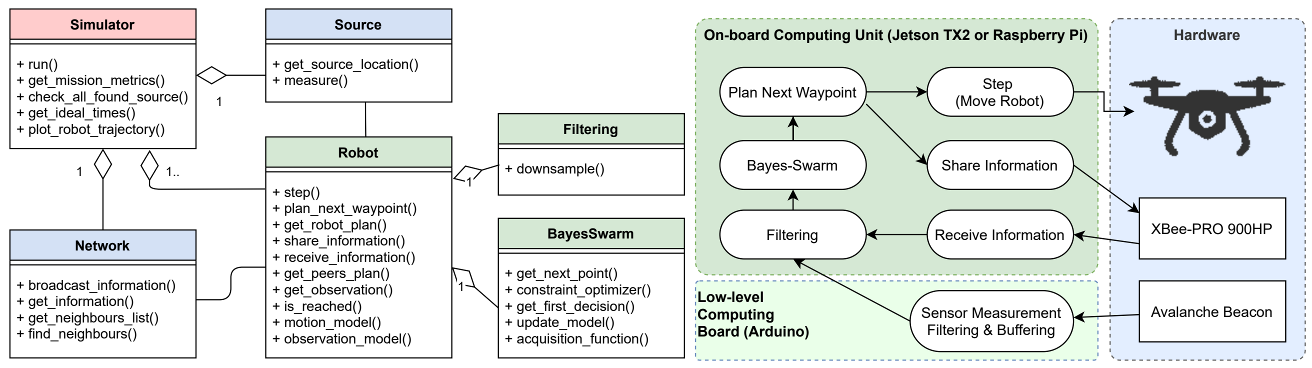
6 Path Towards Implementation on Physical Robots

In this paper, the proposed *Bayes-Swarm-P* algorithm has been tested on synthetic environments with few real-world complexities, in order to focus on the novel algorithmic development underlying the work, and the unique scalability and real-time performance it offers. Now, although there is limited precedence in the literature to physical demonstration of swarm-robot search with functional ground/aerial robots (e.g., UAVs), it is important to lay down a tangible path towards making that possible; and in that process, outline the potential challenges that one might face due to reality gaps. Both of these aspects are discussed in this subsection, with regards to translating *Bayes-Swarm-P* to practice.

Realistic Simulation: Figure 8(a) shows the architecture of the current simulation code as a UML diagram. As shown in this figure, the code has been developed based on the Object-Oriented Programming (OOP) paradigm, with different components such as robot, network, and world abstracted as separate classes. This modular and OOP design not only enables transfer and testing on other state-of-the-art simulators (e.g., Gazebo), but also allows porting the current code to physical implementation. For this purpose, each abstracted component must be replaced with corresponding hardware component. Figure 8(b) shows how a physical implementation of this code might be structured. A Robot Operating System (ROS) framework can be used to perform this software integration in hardware, where most of the software behavior can be run as separate ROS nodes.

Hardware Implementation: For physical robots, UAVs with avalanche beacon are required. There are two options for this purpose: **i)** Enhance an off-the-shelf or customized UAV by equipping it with an avalanche beacon similar to the demonstrated implementation in [11]. **ii)** Use a commercially available UAV that is equipped with a built-in avalanche beacon sensor, such as PowderBee [64]. Another critical hardware required for running such multi-robot algorithms is the communication network that robots use to share information with each other. For providing an effective inter-robot communication link, a 900MHz frequency band, e.g., XBee Pro 900HP [65] can be used. This communication link has been previously used for multi-UAV applications by [66], with a reported range of 5 km and frugal energy footprint, which makes it suitable for our conceived applications.

Potential Challenges and Considerations: Key challenges expected in successfully transitioning from the synthetic environment to a physical demonstration include: **i)** Noisy signal measurements and localization errors: these can be in part dealt with low pass filters, and moreover the Bayesian algorithm can handle noise implicitly as well, unlike other methods. **ii)** Straight-line paths may not be viable in real-world environments, and a prediction of the path length to the candidate go-to points must be used, which can be handled as an uncertainty itself within the optimal planning process, thus requiring the solution of a robust optimization at every path planning instance of a robot. **iii)** Imperfect communication: As discussed in Section 4 - Experiment 5, *Bayes-Swarm-P* can work with partial observations, as is expected in real-world settings; the asynchronous and low-frequency information sharing occurring in *Bayes-Swarm-P* alleviates its dependence on perfect communication. **iv)** Tackling cost of GP updating: One of the challenges in preserving the real-time performance of *Bayes-Swarm-P* is the cost of updating the GP model at each decision-making instance, which grows with the number of robots, mission time and sampling rate. In the future, one could ex-



(a) Simulation Implementation as a UML Diagram (b) Physical Implementation

Fig. 8 System architecture

plore scalable GPs and blending of localized GPs [67,68,69], as well as data compression heuristics, as potential approaches to retain the modeling advantages of GP (in capturing noisy, highly nonlinear signal distributions), while allowing real-time performance as we translate the techniques to compute-scarce hardware.

7 Conclusion

In this paper, we advanced our new decentralized swarm-robotic method to perform searching for the maximum strength source of a spatially distributed signal, which could in practice cater to search & rescue and hazard localization applications. A novel modification of the batch Bayesian optimization formalism is used to construct this method, named as *Bayes-Swarm-P*. The original modifications account for the constraints and capabilities that differentiate embodied search from a non-embodied Bayesian Optimization process. The fundamental advancements, to this original *Bayes-Swarm-O* method, presented in this paper include: i) incorporation of a marginalization-penalization approach to account for the interactions among the waypoints, leading to improved search efficiency and convergence; and ii) dynamic adaptation of the exploitation-exploration balance during the mission. Targeted simulation experiments were presented to study the performance and scalability of the advanced *Bayes-Swarm-P* method under different optimization solver choices, and under the impact of the new penalty formulation. We also investigated how this performance compares with well-known swarm search methods. A new PyBullet based discrete-event simulator, with distributed agent control, was also implemented to run these experiments, and could serve as an open-source environment for benchmarking distributed search and target localization methods.

Both a COBYLA/L-BFGS-B combination and a PSO implementation was observed to serve well in the role of the optimization solver within the *Bayes-Swarm-P* method. In terms of comparative performance, *Bayes-Swarm-P* exhibited clearly superior mission completion times – 100 times faster than Levy-Walk and exhaustive search, and 4 times faster than Glowworm search – over simulated multimodal signal distributions. The performance advantage of *Bayes-Swarm-P* persisted over the multi-UAV avalanche search & rescue problem, and was further enhanced when signal noise was taken into consideration.

Scalability analysis of *Bayes-Swarm-P* demonstrated a superlinear reduction in completion time and mapping error with increasing number of robots. The computing cost per waypoint planning did increase sharply with increasing swarm size, since swarm size exacerbates the cost of re-fitting the GP (onboard swarm robots), which grows as the mission progresses. Thus, while *Bayes-Swarm-P*'s model-based approach provides search efficiency benefits over model-free swarm heuristics like Glowworm, onboard computing costs remains a concern. Dedicated refitting methods, e.g., particle learning, will be explored in the future to address this concern. Limiting communication range and thus reduced peer observability seemed to cause performance losses that are unlikely to justify additional expectation modeling (and its associated computing costs) over unobserved peers. However, these findings were dependent on the case studies that we used here – given the capabilities of the open-source GP libraries that we used. We mainly considered uni-/multi-modal continuous signal distributions with moderate signal-to-noise ratios. This leaves scope for future investigations into scenarios with other types of signal distributions, such as involving gradual variation over large distances, higher noise-to-signal ratios and/or discontinuities e.g., due to impassable areas. In the literature [70,71], it has been shown that by choosing correct kernel functions, GP is able to model non-exponential functions and those with discontinuities, and hence in principle *Bayes-Swarm-P* could be extended to cover signal distributions with such characteristics with informed choice of kernels and hyper-parameters. These advancements along with physics-infused

GP modeling could further enhance the efficiency and robustness of *Bayes-Swarm-P*, and translate it to field experiments.

Nomenclature

α	The exploitation weight, where $\alpha = 1$ would be purely exploitative.
$\Delta\theta$	Initial feasible direction
ϵ	Detection range, a distance that robot can detect the signal source location
$\Gamma_r(.)$	The local penalty term of robot-r in *Bayes-Swarm-P*
$\hat{X}^{k_r}_{-rp}$	Current local peer-p's next waypoint of robot-r at the decision-time k_r
$\hat{\mathbf{X}}^{k_r}_{-r}$	Current local peers' next waypoint of robot-r at the decision-time k_r; i.e., $\hat{\mathbf{X}}^{k_r}_{-r} = \bigcup_{p=1;p\neq r} \hat{X}^{k_r}_{-rp}$
$\mathbf{x}^{k+1}_r$	Next waypoint of robot-r at the decision-time k_r
$\mathbf{X}^i_r$	Location of the observations made by robot-r while it is moving from waypoint-$(i-1)$ to waypoint-i
$\mathbf{y}^i_r$	Source signal measurements made by robot-r while it is moving from waypoint-$(i-1)$ to waypoint-i
$\mathcal{D}^{1:k_r}$	Observations history of robot-r, including self-observations and shared by its peers, from beginning of the mission until finishing its i-th waypoint
$\mathcal{D}^i_r$	Observations of environment that made by robot-r after finishing its i-th waypoint; i.e., $\mathcal{D}^i_r = [\mathbf{X}^i_r, \mathbf{y}^i_r]$
$\Omega_r(.)$	The source seeking term of robot-r in *Bayes-Swarm-P*
$\Sigma_r(.)$	The knowledge-uncertainty reducing term of robot-r in *Bayes-Swarm-P*
GP_r	Gaussian process (GP) model trained and used by robot-r
l	Length of path s
m	Number of robots (swarm size)
$N_{\max}$	Downsample threshold, which defines the maximum allowed samples for fitting the GP model by each robot
r	Robot index, a value between 1 and m
T	Decision-horizon time of robots
V	Nominal velocity of robots

Appendix

A Preliminaries: Gaussian Process Model

Gaussian process (GP) models provide non-parametric surrogates [72] that can be used for Bayesian inference over a function space [36]. For a set of n observations, $\mathcal{D} = \mathbf{x}_i, y_i | i = 1 \dots n$, GP expresses the observed values y_i as a summation of the approximating function $f(\mathbf{x}_i)$ and an additive noise ϵ_i, i.e., $y_i = f(\mathbf{x}_i) + \epsilon_i$. Assuming the noise follows an independent, identically distributed Gaussian distribution with zero mean and variance, σ^2_ϵ, we have $\epsilon \sim \mathcal{N}(0, \sigma^2_\epsilon)$. The function $f(\mathbf{x})$ can then be estimated by a GP with mean $\mu(\mathbf{x})$ and a covariance kernel $\sigma^2(\mathbf{x})$:

$$P(f \mid \mathbf{x},\ \mathbf{X},\ \mathbf{y}) = \mathcal{N}\big(\mu(\mathbf{x}, \mathbf{X}, \mathbf{y}), \sigma^2(\mathbf{x}, \mathbf{X})\big) \tag{15}$$

$$\mu(\mathbf{x}, \mathbf{X}, \mathbf{y}) = \mathbf{k}_n(\mathbf{x})^T (\mathbf{K}^\epsilon)^{-1} \mathbf{y} \tag{16}$$

$$\sigma^2(\mathbf{x}, \mathbf{X}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_n(\mathbf{x})^T (\mathbf{K}^\epsilon)^{-1} \mathbf{k}_n(\mathbf{x}) \tag{17}$$

$$\mathbf{K}^\epsilon(\mathbf{x}) = \mathbf{K} + \sigma^2_\epsilon(\mathbf{x}) \mathbf{I} \tag{18}$$

Here $\mathbf{K} = \mathbf{K}(\mathbf{X}, \mathbf{X}|\theta)$ is the covariance matrix, $(\mathbf{K})_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, with $\mathbf{k}_n(\mathbf{x}) = [k(\mathbf{x}_1, \mathbf{x}), \dots, k(\mathbf{x}_n, \mathbf{x})]^T$. In this paper, the squared exponential kernel is used to define the covariance $k(\mathbf{x}_i, \mathbf{x})$.

B Communication Settings

Further details of the communication settings is given in Table 9.

Table 9 Content, size, and frequency of information shared by robot-r via communication across the swarm.

Item	Descriptions
Inter-robot communication frequency	After each instance of waypoint planning
Content of transmitted data	• Robot-r's planned next waypoint to visit ($\mathbf{x}^{k_r}_r$) • Its observations along the last path ($\mathcal{D}^{k_r}_r$)
Average size of outgoing data packets (with time-horizon 1 min)	364 Bytes
Potential weaker source candidates	Each robot has a list to track of potential weaker sources

C Definition of Case Studies

C.1 Case Study 1: Small arena, non-convex signal distribution

$$f = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_1\|^2}{3}\right) + \frac{1}{2}\exp(-2\|\mathbf{x} - \mathbf{c}_2\|^2) \tag{19}$$

Here, $\mathbf{x} = (x_1, x_2)$, where $0 \le x_i \le 2.4$, $\mathbf{c}_1 = (1, 2)$, and $\mathbf{c}_2 = (2, 0.5)$. The initial feasible direction, $\Delta\theta$ is set at 90.

C.2 Case Study 2: Large arena, highly multi-modal signal distribution

$$f = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_1\|^2}{130}\right) + \frac{2}{5}\sum_{i=2}^{7}\exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{40}\right) \tag{20}$$

Here, $\mathbf{x} = (x_1, x_2)$, where $-24 \le x_i \le 24$. Moreover, $\mathbf{c}_1 = (21, 19)$, $\mathbf{c}_2 = (21, -19)$, $\mathbf{c}_3 = (0, -15)$, $\mathbf{c}_4 = (0, 15)$, $\mathbf{c}_5 = (-19, 10)$, $\mathbf{c}_6 = (21, 19)$, $\mathbf{c}_7 = (-15, -15)$. The initial feasible direction, $\Delta\theta$ is set at 360.

C.3 Case Study 3: Very large arena, highly multi-modal signal distribution

$$f = \exp(-\frac{||\mathbf{x}-\mathbf{c}_1||^2}{6,500}) + \frac{2}{5}\sum_{i=2}^{7}\exp(-\frac{||\mathbf{x}-\mathbf{c}_i||^2}{1,600}) \quad (21)$$

Here, $\mathbf{x} = (x_1, x_2)$, where $-24 \leq x_i \leq 24$. Moreover, $\mathbf{c}_1 = (55, 45)$, $\mathbf{c}_2 = (105, -95)$, $\mathbf{c}_3 = (0, -75)$, $\mathbf{c}_4 = (-25, 100)$, $\mathbf{c}_5 = (-95, 50)$, $\mathbf{c}_6 = (125, 40)$, $\mathbf{c}_7 = (-75, -75)$. There are four depot locations, where located at each corner of the area; i.e., $(-120, -120)$, $(120, -120)$, $(120, 120)$, and $(-120, 120)$.

D Bayes-Swarm-P & Glowworm Settings

Table 10 summarizes the major settings that have been used for *Bayes-Swarm-P* and *Bayes-Swarm-O* for all experiments and case studies. Tables 11 and 12 respectively list the settings that have been used for the Glowworm Search and the Levy Walk algorithms in Experiment 2 and the avalanche problem.

Table 10 The settings of the *Bayes-Swarm-P* algorithm.

Experiment	Case Study	m	V [m/s]	T [s]
1	1	5	0.1	10
1	2	5	1	20
2	1	5	0.1	10
2	2	10	1	20
3	2	[5-50]	1	20
4	1	5	0.1	10
4	2	10	1	20
5	1	5	0.1	10
5	2	10	1	20
6	3*	6	1	20

m: # robots; V: Velocity of robots; and T: Decision-horizon length.
*Case 3 represents the avalanche problem in Sec. 5.

Table 11 Glowworm algorithm settings (experiment 2 & avalanche problem).

ρ	γ	β	r_s	Δs [m]
0.4	0.6	0.08	20	0.03

m: Number of robots; ρ: Luciferin decay constant; γ: Luciferin enhancement constant; β: Decision range gain; r_s: Sensor range of robots; and Δs: Distance moved by each Glowworm when a decision is taken.

E Avalanche: Search & Rescue Operation

Figure 9 illustrates a high-level description of how the manned mission progresses vs. how a team of UAVs would conceivably operate. The left side of Fig. 9 shows a typical search procedure that is conducted by human rescuer using a beacon receiver. This includes four main stages [73]: **A) Signal Search:** In this signal search or scouting process, the rescuer skies downslope along the path taken by the victim to detect measurable beacon signal; **B) Coarse Search:** After detecting the first signal, the rescuer searches the area with a coarse step size to get within the vicinity of the buried victim; **C) Fine Search:** The rescuer searches the vicinity of the buried victim with smaller step size to reach to get closer to the victim; and **D) Pinpointing:** At the end, the rescuer uses a probe to pinpoint the exact location of the victim. It should be noted the rescuer uses the beacon receiver during the first three stages of the search, and uses a probe for the last stage.

Table 12 Levy walk search algorithm settings (experiment 2 & avalanche problem).

ρ	μ	l_0
0.2	2	0.5

ρ: Luciferin decay constant; μ: Luciferin enhancement constant; l_0: Decision range gain.

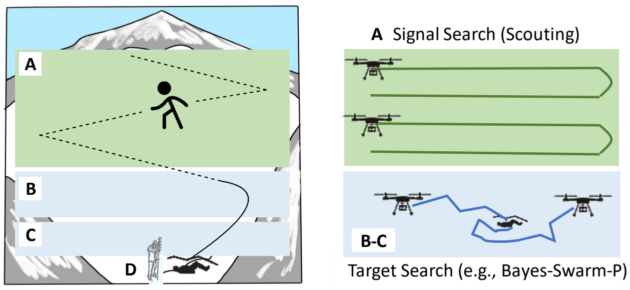

A
B
C
D
A Signal Search (Scouting)
B-C
Target Search (e.g., Bayes-Swarm-P)

Fig. 9 Process of searching for skier trapped under avalanche: Left figure demonstrates manned mission progress; right figure shows how a team of UAVs would operate. [Image recreated based on [73]]

The first three stages of the search (A-C) can be achieved using a UAV or a team of UAVs, as shown on the right side of Fig. 9. For this purpose, the signal search can be achieved using an exhaustive scanning search. The stages B and C can be conducted using *Bayes-Swarm-P* or other similar methods.

Acknowledgements Support from the National Science Foundation (NSF) Awards IIS-1927462 and CMMI-2048020 is gratefully acknowledged. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

1. Tan, Y., Zheng, Z.y.: Research advance in swarm robotics. Defence Technology **9**(1), 18–39 (2013)
2. Marconi, L., Melchiorri, C., Beetz, M., Pangercic, D., Siegwart, R., Leutenegger, S., Carloni, R., Stramigioli, S., Bruyninckx, H.,

Doherty, P., et al.: The sherpa project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments. In: 2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp. 1–4. IEEE (2012)
3. Rahman, M.A., Azad, S., Asyhari, A.T., Bhuiyan, M.Z.A., Anwar, K.: Collab-sar: A collaborative avalanche search-and-rescue missions exploiting hostile alpine networks. IEEE Access **6**, 42094–42107 (2018)
4. Tadokoro, S.: Disaster Robotics: Results from the ImPACT Tough Robotics Challenge, vol. 128. Springer (2019)
5. Baetz, W., Kroll, A., Bonow, G.: Mobile robots with active ir-optical sensing for remote gas detection and source localization. In: 2009 IEEE International Conference on Robotics and Automation, pp. 2773–2778. IEEE (2009)
6. Viseras, A., Wiedemann, T., Manss, C., Magel, L., Mueller, J., Shutin, D., Merino, L.: Decentralized multi-agent exploration with online-learning of gaussian processes. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 4222–4229. IEEE (2016)
7. Song, D., Kim, C.Y., Yi, J.: Simultaneous localization of multiple unknown and transient radio sources using a mobile robot. IEEE Transactions on Robotics **28**(3), 668–680 (2012)
8. Behjat, A., Manjunatha, H., Kumar, P.K., Jani, A., Collins, L., Ghassemi, P., Distefano, J., Doermann, D., Dantu, K., Esfahani, E., et al.: Learning robot swarm tactics over complex adversarial environments. In: 2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), pp. 83–91. IEEE (2021)
9. Sibanyoni, S.V., Ramotsoela, D.T., Silva, B.J., Hancke, G.P.: A 2-d acoustic source localization system for drones in search and rescue missions. IEEE Sensors Journal **19**(1), 332–341 (2018)
10. Ghassemi, P., Chowdhury, S.: An Extended Bayesian Optimization Approach to Decentralized Swarm Robotic Search. Journal of Computing and Information Science in Engineering **20**(5) (2020)
11. Silvagni, M., Tonoli, A., Zenerino, E., Chiaberge, M.: Multipurpose uav for search and rescue operations in mountain avalanche events. Geomatics, Natural Hazards and Risk **8**(1), 18–33 (2017)
12. Pugh, J., Martinoli, A.: Inspiring and modeling multi-robot search with particle swarm optimization. In: Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, pp. 332–339. IEEE (2007)
13. Yang, G.Z., Bellingham, J., Dupont, P.E., Fischer, P., Floridi, L., Full, R., Jacobstein, N., Kumar, V., McNutt, M., Merrifield, R., et al.: The grand challenges of science robotics. Science robotics **3**(14), eaar7650 (2018)
14. Gunning, D.: Explainable artificial intelligence (xai). Defense Advanced Research Projects Agency (DARPA), nd Web (2017)
15. Senanayake, M., Senthooran, I., Barca, J.C., Chung, H., Kamruzzaman, J., Murshed, M.: Search and tracking algorithms for swarms of robots: A survey. Robotics and Autonomous Systems **75**, 422–434 (2016)
16. Sinha, A., Mishra, R.K.: Convergence of multi-agent systems to unknown source of an odor. In: 2018 3rd International Conference for Convergence in Technology (I2CT), pp. 1–6 (2018). DOI 10.1109/I2CT.2018.8529713
17. Liu, M., Sivakumar, K., Omidshafiei, S., Amato, C., How, J.P.: Learning for multi-robot cooperation in partially observable stochastic environments with macro-actions. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1853–1860. IEEE (2017)
18. Wiedemann, T., Shutin, D., Hernandez, V., Schaffernicht, E., Lilienthal, A.J.: Bayesian gas source localization and exploration with a multi-robot system using partial differential equation based modeling. In: Olfaction and Electronic Nose (ISOEN), 2017 ISOCS/IEEE International Symposium on, pp. 1–3. IEEE (2017)
19. Charrow, B., Michael, N., Kumar, V.: Cooperative multi-robot estimation and control for radio source localization. The International Journal of Robotics Research **33**(4), 569–580 (2014)
20. Hajieghrary, H., Hsieh, M.A., Schwartz, I.B.: Multi-agent search for source localization in a turbulent medium. Physics Letters A **380**(20), 1698–1705 (2016)
21. Kennedy, J.: Particle swarm optimization. Encyclopedia of machine learning pp. 760–766 (2010)
22. Kennedy, J.: Swarm intelligence. In: Handbook of nature-inspired and innovative computing, pp. 187–219. Springer (2006)
23. Bonabeau, E., Marco, D.d.R.D.F., Dorigo, M., Théraulaz, G., Theraulaz, G., et al.: Swarm intelligence: from natural to artificial systems. 1. Oxford university press (1999)
24. Jatmiko, W., Sekiyama, K., Fukuda, T.: A pso-based mobile sensor network for odor source localization in dynamic environment: Theory, simulation and measurement. In: 2006 IEEE International Conference on Evolutionary Computation, pp. 1036–1043. IEEE (2006)
25. Pang, B., Song, Y., Zhang, C., Wang, H., Yang, R.: A swarm robotic exploration strategy based on an improved random walk method. Journal of Robotics **2019** (2019)
26. Kaipa, K., Ghose, D.: Glowworm swarm optimization-modifications and applications. Swarm Intelligence Algorithms: Modifications and Applications p. 187 (2020)
27. Kolling, A., Walker, P., Chakraborty, N., Sycara, K., Lewis, M.: Human interaction with robot swarms: A survey. IEEE Transactions on Human-Machine Systems **46**(1), 9–26 (2016)
28. Brochu, E., Cora, V.M., De Freitas, N.: A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599 (2010)
29. González, J., Dai, Z., Hennig, P., Lawrence, N.: Batch bayesian optimization via local penalization. In: Artificial Intelligence and Statistics, pp. 648–657 (2016)
30. Klavins, E.: Communication complexity of multi-robot systems. In: Algorithmic Foundations of Robotics V, pp. 275–291. Springer (2004)
31. Cáp, M., Novák, P., Seleckỳ, M., Faigl, J., Vokffnek, J.: Asynchronous decentralized prioritized planning for coordination in multi-robot system. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3822–3829. IEEE (2013)
32. Ghosh, S.: Understanding complex, real-world systems through asynchronous, distributed decision-making algorithms. Journal of Systems and Software **58**(2), 153–167 (2001)
33. Sujit, P., Ghose, D.: Negotiation schemes for multi-agent cooperative search. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering **223**(6), 791–813 (2009)
34. Ginsbourger, D., Le Riche, R., Carraro, L.: A multi-points criterion for deterministic parallel global optimization based on gaussian processes (2008)
35. Azimi, J., Fern, A., Fern, X.Z.: Batch bayesian optimization via simulation matching. In: Advances in Neural Information Processing Systems, pp. 109–117 (2010)
36. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, pp. 2951–2959 (2012)
37. Chevalier, C., Ginsbourger, D.: Fast computation of the multi-points expected improvement with applications in batch selection. In: International Conference on Learning and Intelligent Optimization, pp. 59–69. Springer (2013)
38. Wang, J., Clark, S.C., Liu, E., Frazier, P.I.: Parallel bayesian global optimization of expensive functions. arXiv preprint arXiv:1602.05149 (2016)
39. Ghassemi, P., Lulekar, S.S., Chowdhury, S.: Adaptive Model Refinement with Batch Bayesian Sampling for Optimization of Bio-inspired Flow Tailoring. In: AIAA Aviation 2019 Forum. American Institute of Aeronautics and Astronautics, Dallas, TX, USA (2019). DOI 10.2514/6.2019-2983

40. Ghassemi, P., Chowdhury, S.: Informative Path Planning with Local Penalization for Decentralized and Asynchronous Swarm Robotic Search. In: International Symposium on Multi-Robot and Multi-Agent Systems, MRS 2019, pp. 188–194. Institute of Electrical and Electronics Engineers., New Brunswick, NJ (2019). DOI 10.1109/MRS.2019.8901084
41. Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org (2016-2019)
42. Kaipa, K.N., Ghose, D.: Glowworm swarm optimization: theory, algorithms, and applications, vol. 698. Springer (2017)
43. Pasternak, Z., Bartumeus, F., Grasso, F.W.: Lévy-taxis: a novel search strategy for finding odor plumes in turbulent flow-dominated environments. Journal of Physics A: Mathematical and Theoretical **42**(43), 434010 (2009)
44. Wood, G., Zhang, B.: Estimation of the lipschitz constant of a function. Journal of Global Optimization **8**(1), 91–103 (1996)
45. Das, A., Kempe, D.: Estimating the average of a lipschitz-continuous function from one sample. In: European Symposium on Algorithms, pp. 219–230. Springer (2010)
46. Beard, R.W., McLain, T.W.: Multiple uav cooperative search under collision avoidance and limited range communication constraints. In: Decision and Control, 2003. Proceedings. 42nd IEEE Conference on, vol. 1, pp. 25–30. IEEE (2003)
47. Li, M., Lu, K., Zhu, H., Chen, M., Mao, S., Prabhakaran, B.: Robot swarm communication networks: architectures, protocols, and applications. In: 2008 Third International Conference on Communications and Networking in China, pp. 162–166. IEEE (2008)
48. Wade, G.: Signal coding and processing. Cambridge university press (1994)
49. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM Journal on scientific computing **16**(5), 1190–1208 (1995)
50. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E.W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Contributors, S...: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020)
51. Nash, S.G.: Newton-type minimization via the lanczos method. SIAM Journal on Numerical Analysis **21**(4), 770–788 (1984)
52. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360), pp. 69–73. IEEE (1998)
53. Lee, A.: Particle swarm optimization (pso) with constraint support (2014). URL https://pypi.org/project/pyswarm/
54. Powell, M.J.: A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Advances in optimization and numerical analysis, pp. 51–67. Springer (1994)
55. Kraft, D., et al.: A software package for sequential quadratic programming. Tech. rep., DFVLR Obersfaffeuhofen, Germany (1988)
56. Byrd, R.H., Hribar, M.E., Nocedal, J.: An interior point algorithm for large-scale nonlinear programming. SIAM Journal on Optimization **9**(4), 877–900 (1999)
57. Krishnanand, K., Ghose, D.: Glowworm swarm optimisation: a new method for optimising multi-modal functions. International Journal of Computational Intelligence Studies **1**(1), 93–119 (2009)
58. Liao, W.H., Kao, Y., Li, Y.S.: A sensor deployment approach using glowworm swarm optimization algorithm in wireless sensor networks. Expert Systems with Applications **38**(10), 12180–12188 (2011)
59. Cao, M., Meng, Q., Luo, B., Zeng, M.: Experimental comparison of random search strategies for multi-robot based odour finding without wind information. Austrian Contributions to Veterinary Epidemiology **8**, 43–50 (2015)
60. Peterson, J., Li, W., Cesar-Tondreau, B., Bird, J., Kochersberger, K., Czaja, W., McLean, M.: Experiments in unmanned aerial vehicle/unmanned ground vehicle radiation search. Journal of Field Robotics **36**(4), 818–845 (2019)
61. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information processing letters **85**(6), 317–325 (2003)
62. Ayuso, N., Cuchí, J., Lera, F., Villarroel, J.: A deep insight into avalanche transceivers for optimizing rescue. Cold Regions Science and Technology **111**, 80–94 (2015)
63. Piniés, P., Tardós, J.D.: Fast localization of avalanche victims using sum of gaussians. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pp. 3989–3994. IEEE (2006)
64. Powderbee. URL https://www.bluebirdmountain.de/?page_id=54
65. Datasheet: Digi xbee-pro 900hp. Digi International Inc (2018)
66. Silva, M.R., Souza, E.S., Alsina, P.J., Leite, D.L., Morais, M.R., Pereira, D.S., Nascimento, L.B., Medeiros, A.A., Junior, F.H.C., Nogueira, M.B., et al.: Performance evaluation of multi-uav network applied to scanning rocket impact area. Sensors **19**(22), 4895 (2019)
67. Nguyen-Tuong, D., Peters, J., Seeger, M.: Local gaussian process regression for real time online model learning and control. In: Proceedings of the 21st International Conference on Neural Information Processing Systems, pp. 1193–1200 (2008)
68. Quinonero-Candela, J., Rasmussen, C.E., Williams, C.K.: Approximation methods for gaussian process regression. In: Large-scale kernel machines, pp. 203–223. MIT Press (2007)
69. Liu, H., Ong, Y.S., Shen, X., Cai, J.: When gaussian process meets big data: A review of scalable gps. IEEE transactions on neural networks and learning systems **31**(11), 4405–4423 (2020)
70. Branson, Z., Rischard, M., Bornn, L., Miratrix, L.W.: A nonparametric bayesian methodology for regression discontinuity designs. Journal of Statistical Planning and Inference **202**, 14–30 (2019)
71. Mohammadi, H., Challenor, P., Goodfellow, M., Williamson, D.: Emulating computer models with step-discontinuous outputs using gaussian processes. arXiv preprint arXiv:1903.02071 (2019)
72. Williams, C.K., Rasmussen, C.E.: Gaussian processes for machine learning, vol. 2. MIT press Cambridge, MA (2006)
73. Mammut: Barryvox s extended reference guide (2020)