Comparing Stochastic Optimization Methods for Multi-Robot, Multi-Target Tracking

Pujie Xin and Philip Dames

Temple University, Philadelphia, PA 19122, USA,
{pujie.xin,pdames}@temple.edu,
home page: https://sites.temple.edu/trail/

Abstract. This paper compares different distributed control approaches that enable a team of robots search for and track an unknown number of targets. The robots are equipped with sensors which have limited field of view (FoV) and are required to explore the environment. The team uses a distributed formulation of the Probability Hypothesis Density (PHD) filter to estimate the number and the position of the targets. The resulting target estimate is used to select the future search locations for each robot. This paper compares Lloyd's algorithm, a traditional method for distributed search, with two typical stochastic optimization methods, Particle Swarm Optimization (PSO) and Simulated Annealing (SA). PSO and SA are traditionally used to find a single global maximum, therefore this paper describes novel formulations of PSO and SA to solve the problem of multi-target tracking. These new methods more effectively trade off between exploration and exploitation. Simulations demonstrate that the use of these stochastic optimization techniques improves coverage of the search space and reduces the error in the target estimates compared to the baseline approach.

Keywords: Multi-target tracking, Stochastic optimization, Distributed control

1 Introduction

Multi-robot, multi-target tracking (MR-MTT) encompasses many traditional robotics problems, including search and rescue [19], surveillance [10], and mapping [9]. All of these scenarios require a team of robots to search across an environment to detect, identify, and track individual objects. A generic solution to the MR-MTT problem consists of two parts: 1) an estimation algorithm to determine the number and states of all targets targets and 2) a control algorithm to drive the robots to seek out undetected targets as well as track detected targets. These problems are extensively studied in multi-robot systems [29].

1.1 Related Work

Multi-target tracking (MTT) is a problem wherein one or more sensors seek to determine the number of targets in a space and the state of each individual

target. Often the number of targets is unknown a priori and in many cases the targets are not uniquely identifiable. One of the main challenges in MTT is the data association problem, *i.e.,* matching measurements to potential targets. Stone et al. [25] discuss this problem in their book, and survey existing solutions, such as the Multiple Hypothesis Tracker (MHT) [1], Joint Probabilistic Data Association (JPDA) [12], and the Probability Hypothesis Density (PHD) filter [17]. Each of these different methods solve data association in a different way. We pick the PHD filter due to its relative simplicity and good performance.

Multi-robot search is another well studied problem in the literature, with many methods existing methods, such as those discussed in the survey [22]. In recent years, Multi-Agent Deep Reinforcement Learning (MADRL) has been popular to learn agent coordination policies [21, 30]. Some examples use partially observable Monte-Carlo planning [14] or perform a joint optimization of target assignment and path planning [20]. However, one main challenge in MADRL is many of the methods are based on centralized policy and global information. One of the most widely classical techniques is Lloyd's algorithm [11], a decentralized algorithm which iterative partitions the environment and drives the robots using a gradient descent controller. Due to its wide popularity, we will use Lloyd's algorithm as our baseline approach for comparison.

In this paper, we will examine the use of other numerical optimization algorithms to drive MR-MTT. These fall into two broad categories: deterministic and stochastic. Deterministic methods often suffer from local optima entrapment, especially for the gradient-based methods which require the knowledge of the whole search space [18], which we do not have in our setting because the target locations are unknown a priori. On the other hand, stochastic optimization methods do not rely on the gradient descent which makes these particularly fitted for solving problems with unknown search spaces. Based on the number of candidates solutions, there are individual-based and population-based algorithms. In the context of numerical optimization, these techniques are typically used to find a single global extremum of a function. However, in the MR-MTT problem, we instead wish to find all local maxima (*i.e.,* all target locations). Thus, we will need to reformulate these techniques to suit our problem.

In the family of the individual-based methods, there are Tabu Search [13], Hill Climbing [7], and simulated annealing (SA) [16]. The SA algorithm is widely used in solving unconstrained and bound-constrained optimization problems. This method models the physical activity of heating and tempering in material process. It is likely to accept worse solutions with a scale proportional to the temperature. This enables SA explore broad search space and escape from local optima, which will allow the robots to better explore the environment. People proposed different versions of distributed control with SA based on a variety of objective functions. One similar work is to minimize the total travel distance under Boolean specification using SA [24]. Their scenario is similar to that in Travelling salesman problem (TSP), which enumerates the trajectories. The trajectories are the combination of different nodes.

PSO [15] is one of the most successful in population-based algorithms. Derr et al. [8] proposed one decentralized version of PSO for MR-MTT, but in their experiment, the targets are mobile phones. This makes the problem much simpler since the sensors have unlimited range (*i.e.,* are able to detect the signal strength everywhere in the environment) and there is no ambiguity in data association. Tang et al. [26] proposed methods in the environment with obstacles, but the optimization function is still predefined, which means the global importance weight is known to each robot (*i.e.,* target locations are known). Our solution differs from these other works in two key ways: 1) the objective function or the global importance weight of the map is unknown and 2) the robots have a limited field of view.

1.2 Contributions

In this paper, we propose two novel reformulations of traditional stochastic optimization algorithms, SA and PSO, to solve the MR-MTT problem. We compare these against Lloyd's algorithm, a traditional method for multi-robot search. In all cases we keep the distributed MTT algorithm constant, so the only difference in performance is due to the search strategy. Our results show that the new techniques result in more complete coverage of the search space and lower errors in the target estimate. We also see that SA and PSO yield qualitatively different search patterns as a result of their different structures.

2 Problem Formulation

We have a team of R robots exploring a convex environment $E \subseteq R^2$ to search for unknown number of targets. The set of targets are $\mathbf{X}^t = \{\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots\}$, where $\mathbf{x}_i^t$ is the state of the i^{th} target at time t. At each time step, robot r receives a set of measurements $\mathbf{Z}_r^t = \{\mathbf{z}_{1,r}^t, \mathbf{z}_{2,r}^t, \ldots\}$. The number of measurements depends upon the number of targets within the robot's field of view (FoV) as well as the chance of receiving a false negative or false positive detection.

2.1 PHD Filter

The PHD filter [17] models the sets $\mathbf{X}$ and $\mathbf{Z}$ above as realizations of Poisson random finite sets (RFSs), which are sets composed of independently and identically distributed (i.i.d.) elements where the number of elements follows a Poisson distribution. The likelihood of such an RFS $\mathbf{X}$ is

$$p(\mathbf{X}) = e^{-\lambda} \prod_{\mathbf{x} \in \mathbf{X}} v(\mathbf{x}) \tag{1}$$

where $v(\mathbf{x})$ is the PHD and $\lambda = \int_E v(\mathbf{x}) d\mathbf{x}$. The PHD is a density function over the state space of the targets, and the expected cardinality of $\mathbf{X}$ is the integral of the PHD over E.

The PHD filter recursively updates $v(\mathbf{x})$, which completely encodes the distribution over sets, using the measurement sets collected by the robots and the motion model of targets. The PHD filter uses six models. 1) The target motion model $f(\mathbf{x}^t \mid \mathbf{x}^{t-1})$ characterizes the state transition model of $\mathbf{x}$. 2) The birth PHD $b(\mathbf{x}^t)$ describes both of the number of targets and their location at t. 3) The survival probability, $p_s(\mathbf{x}^{t-1})$, models the existence of a target with state $\mathbf{x}^{t-1}$. 4) The detection model is $p_d(\mathbf{x}^t \mid \mathbf{q})$ describes the targets' detected states $\mathbf{x}^t$ with a sensor state $\mathbf{q}$. 5) The measurement model, $g(\mathbf{z} \mid \mathbf{x}^t; \mathbf{q})$, describes the probability of a robot with state $\mathbf{q}$ receiving measurement $\mathbf{z}$ from a detected target with state $\mathbf{x}^t$. Combining this with the detection model yields $\psi_{\mathbf{z},\mathbf{q}}(\mathbf{x}^t) = g(\mathbf{z} \mid \mathbf{x}^t, \mathbf{q}) p_d(\mathbf{x}^t \mid \mathbf{q})$, which characterizes the probability of receiving measurement $\mathbf{z}$ from a target with state $\mathbf{x}^t$ by a sensor locating at $\mathbf{q}$. 6) Finally, the clutter model $c(\mathbf{z})$ is another PHD describing the number and states of false positive detections. With these targets and sensor models, the prediction and update of the PHD filter are:

$$\bar{v}^t(\mathbf{x}^t) = b(\mathbf{x}^t) + \int_E f(\mathbf{x}^t \mid \mathbf{x}^{t-1}) p_s(\mathbf{x}^{t-1}) v^{t-1}(\mathbf{x}^{t-1})\, d\mathbf{x}^{t-1} \tag{2}$$

$$v^t(\mathbf{x}^t) = (1 - p_d(\mathbf{x}^t \mid \mathbf{q}))\bar{v}^t(\mathbf{x}^t) + \sum_{\mathbf{z} \in \mathbf{Z}^t} \frac{\psi_{\mathbf{z},\mathbf{q}}(\mathbf{x}^t)\bar{v}^t(\mathbf{x}^t)}{\eta_{\mathbf{z}}(\bar{v}^t(\mathbf{x}^t))} \tag{3}$$

$$\eta_{\mathbf{z}}(v) = c(\mathbf{z} \mid \mathbf{q}) + \int_E \psi_{\mathbf{z},\mathbf{q}}(\mathbf{x}^t) v^t(\mathbf{x}^t)\, d\mathbf{x}^t \tag{4}$$

Here, $\eta_{\mathbf{z}}(v)$ is a normalization factor, though note that $\int_E v(\mathbf{x})\, d\mathbf{x} = \lambda$, *i.e.,* the expected number of targets in the search space, rather than 1, which is standard in single-target tracking.

Since the PHD $v(\mathbf{x})$ is a density function in the state space of targets, we need to make an additional assumption to parametrically represent it in a manner that is amenable to computations. The two traditional methods are as a set of weighted particles [28] or a mixture of Gaussians [27]. We use the former, as this allows for arbitrarily non-linear sensing and target models.

2.2 Lloyd's Algorithm

Lloyds' algorithm [3] iterative partitioning the environment and navigating the robots towards the weighted centroid of their Voronoi cell. This process locally minimizes the function:

$$\mathcal{H}(\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_R) = \int_E \min_{r \in \{1,...,R\}} f(d(\mathbf{x}, \mathbf{q}_r)) \phi(x) dx, \tag{5}$$

where $d(\mathbf{x}, \mathbf{q})$ measures the distance between elements in E, $f(\cdot)$ is a monotonically increasing function, and $\phi(\mathbf{x})$ is a non-negative weighting function. A general choice is $f(\mathbf{x}) = \mathbf{x}^2$. The inside of the integral approaches the minimum when the environment is continuously divided using the Voronoi partition.

$V_i = \{\mathbf{x} \mid d(\mathbf{x}, \mathbf{q}_i) \leq d(\mathbf{x}, \mathbf{q}_j), \forall i \neq j\}$ where V_i is the Voronoi cell for robot i. The minimum with respect to robot's position is

$$(\mathbf{q}^i)^* = \frac{\int_{V_i} \mathbf{x}\phi(\mathbf{x})\, dx}{\int_{V_i} \phi(\mathbf{x})\, dx}, \tag{6}$$

This is a distributed control algorithm since if each robot knows the positions of its neighbors, the robot is able to compute its Voronoi cell and move to the weighted centroid of its Voronoi cell. In our case, the PHD serves as weighting function $\phi(\mathbf{x})$, converting the traditional coverage problem into a search and tracking problem.

2.3 Distributed PHD Filter

We utilize the distributed PHD filter from our previous work [6]. The key to this is that each robot maintains the portion of the PHD within its own Voronoi cell and exchanging the data with its Voronoi neighbors. There are then three algorithms that account for the motion of Voronoi cells (*i.e.,* motion of robots), motion of targets across cell boundaries, and measurements within other robots' cells. This estimation process is described and the convergence is proved in the previous work [6]. Since the distributed estimation method is same, we will focus on the distributed control method. Also note that since all teams use the distributed PHD filter, all use the same Voronoi-based partitioning scheme regardless of the choice of search strategy.

2.4 Assumptions

Throughout this work, the first assumption is the each robot knows its own position. This is a strong assumption but it is easily achieved with localization algorithm in indoor environment[4] and GPS receivers in outdoor environment. Additionally, we have shown that the distributed PHD filter can work well even with some bounded uncertainty in the pose estimates [6].

Another assumption is that all robots can communicate with all of their Voronoi neighbors, a necessary assumption under Lloyd's algorithm (and necessary for our distributed PHD filter formulation). In a practical setting, this problem might be solved by using robots as communication relay or using multi-hop communication strategies.

3 Distributed Control

As is mentioned in the introduction, stochastic optimization algorithm can be divided into population-based algorithm and individual-based algorithm according the number of candidate solutions. In this paper, we pick one typical method from each family and compare these methods with the standard baseline of Lloyd's algorithm. Specifically, we use SA as an example of individual-based algorithms and PSO as an example of population-based algorithms.

Additionally, in traditional optimization, algorithms attempt to find a single global maximum of some function. However, in the multi-target tracking case we instead wish to find all the local maxima. In the context of PHD, the local maxima represents the PHD peaks in search space, which correspond to the targets' location.

3.1 Simulated Annealing

Simulated annealing takes inspiration from the annealing process in manufacturing, where materials are slowly cooled to allow them to come to a stable configuration. In SA, at each iteration, a new candidate point locating at $\hat{\mathbf{q}}^{t+1}$ is randomly generated near the last solution point $\mathbf{q}^t$, where the distance $\delta = |\hat{\mathbf{q}}^{t+1} - \mathbf{q}^t|$ is proportional to the "temperature" T. Suppose we seek to maximizing the objective function. The solution then moves to the new location if the value is better at the new location or with some probability if the value is worse. This probability of accepting a suboptimal solution is given by:

$$P_{\text{accept}} \sim \text{Bernoulli}\left(\frac{1}{1+\exp\left(\frac{\delta}{\max(T)}\right)}\right) \tag{7}$$

which allows the solution to escape from local optima to hopefully reach a global optimum, with the probability of a jump decreasing over time as the temperature cools.

The rules governing simulated annealing correspond to a single searching agent. Thus, in a multi-robot setting, each agent selects actions on its own, based on the information stored in the shared, distributed PHD. Our approach is outlined in Algorithm 1. First, robot r expands its Voronoi cell $\hat{V}_r$ and exchange the particles PHD with its neighbor (line 1-5). The weight c_{old} is from the sum of PHD of the particles in the overlapped area between $\hat{V}_r$ and its FoV. Robots then search for the best action, from a finite set of actions composed of angles θ following discrete uniform distribution and fixed step size $\mathbf{d}$ (line 7). Then robot r searches over potential new locations (line 8), where T is the temperature, R is a rotation matrix, and $\mathbf{d}$ is a standard step. For each potential location, the robot predicts the weight it may obtain when it move around using the sum of PHD of the particles in the overlapped area between $\hat{V}_r$ and its potential future location (line 9). The movement θ^* leading to the max Δc will be selected as the next movement step (line 11). Robot r will move to the new position $\hat{\mathbf{q}}^{t+1}$ only if the $\Delta c > 0$ or $\text{rand}(0,1) < P_{\text{accept}}$ (lines 12-17). Otherwise, the robot falls back to the standard Lloyd's algorithm (line 19).

3.2 Particle Swarm Optimization

In PSO, a collection of particles P move on the search space and the algorithm evaluate the objective value at each particle. These particles are the candidate

Algorithm 1 Distributed control with SA for Robot r

$\hat{V}_r = V_r \oplus \mathcal{B}(d)$ ▷ Expand the Voronoi Cell
for $i \in \mathcal{N}(r)$ **do** ▷ This step can be combined with particle exchange in estimation
  Compute $\Delta V_{i,r} = V_i \cap V_r$
  Send particles in $\Delta V_{i,r}$ to robot i
end for
Compute $c_{\text{old}} = \sum_{\hat{V}_r \cap FoV_r(\mathbf{q}_r)} v(x)$
for $\theta \in [0 : \Delta\theta : 2\pi]$ **do**
  $\hat{\mathbf{q}}^{t+1}(\theta) = q_r + TR(\theta)\mathbf{d}$
  $\Delta c(\theta) = \sum_{\hat{V}_r \cap FoV_r(\hat{\mathbf{q}}^{t+1}(\theta))} v(x) - c_{\text{old}}$
end for
$\theta^* = \arg\max_\theta c(\theta)$
if $\Delta c(\theta^*) > 0$ **then**
  $\mathbf{q}_r^{t+1} = \hat{\mathbf{q}}^{t+1}(\theta^*)$
  $T = T_{\text{init}}$ ▷ $T_{\text{init}} = 1$ in experiment
else if $\text{rand}(0,1) < P_{\text{accept}}$ **then** ▷ P_{accept} from (7)
  $\mathbf{q}_r^{t+1} = \hat{\mathbf{q}}^{t+1}(\theta^*)$
  $T := 0.95T$ ▷ 0.95 is a typical cooling factor
else
  Set $\mathbf{q}_r^{t+1}$ according to (6).
  $T := 0.95T$
end if

solutions where the particle i locates at $\mathbf{q}_t^i$ at time t. The velocity $\mathbf{v}_t^i$ of particle i is updated with the algorithm evaluation function.

$$\begin{aligned} \mathbf{q}_i^t &= \mathbf{q}_i^{t-1} + \mathbf{v}_i^t \\ \mathbf{v}_i^t &= w\mathbf{v}_i^{t-1} + c_1 r_1(\mathbf{q}_{i,b}^* - \mathbf{q}_i^t) + c_2 r_2(\mathbf{q}_{i,g}^* - \mathbf{q}_i^t) \end{aligned} \tag{8}$$

where r_1 and r_2 are random numbers between 0 and 1, c_1 and c_2 are predefined constant numbers in PSO. And w is called as the inertia term. $\mathbf{q}_{i,b}^{i*}$ is the position of particle i in which the objective function get the best value (*i.e.,* a local best). And $\mathbf{q}_{i,g}^*$ is the position where the best objective function value in the history (*i.e.,* a global best).

PSO is a metaheuristic method which generates multiple candidate solution in search space at the same time. This has a natural analog in multi-robot search, where each robot represents a single particle. However, slightly different from standard PSO, a robot can measure the value of points everywhere in its FoV instead of a single point. Additionally, we do not want to find a single global best value for the objective function, but rather to find all local maxima. Thus, we need to redfine the meaning of $\mathbf{q}_{\cdot,g}^*$ and $\mathbf{q}_{\cdot,b}^*$. In Algorithm 2, we set $\mathbf{q}_{\cdot,g}^*$ equal to the weighted center of Voronoi cell (*i.e.,* the location from Lloyd's algorithm). And if the robots obtains the highest weight in the history at current location, then it will set $\mathbf{q}$ as the $\mathbf{q}_{\cdot,b}^*$. The highest weight in the history w_{history} is initialized as the obtained weight at $t = 1$ and then it is updated with the current weight w_t if $w_t \geq w_{\text{history}}$. Since $r_1, r_2 \sim \text{rand}(0,1)$, the robot moves with random trade off between $\mathbf{q}_{\cdot,b}^*$ and $\mathbf{q}_{\cdot,g}^*$.

Algorithm 2 Distributed control with PSO for Robot r

1: Compute local $w_t = \sum_{\mathbf{x} \in FoV} v(\mathbf{x})$ at time t.
2: Set $\mathbf{q}_g^*$ according to (6)
3: **if** $w_t \geq w_{\text{history}}$ **then**
4: $\quad \mathbf{q}_{r,b}^* = \mathbf{q}_r$ ▷ Current robot location
5: $\quad w_{\text{history}} = w_t$
6: **end if**
7: $\mathbf{v}_r^t = w\mathbf{v}_r^{t-1} + c_1 r_1(\mathbf{q}_{r,b}^* - \mathbf{q}_r^t) + c_2 r_2(\mathbf{q}_{r,g}^* - \mathbf{q}_i^t)$ ▷ Experiments set $w, c_1, c_2 = 1$
8: $\mathbf{q}_r^{t+1} = \mathbf{q}_r^t + \mathbf{v}_r^t$ ▷ Robot moves to $\mathbf{q}_{t+1}$

3.3 Communication and Complexity

All teams use the distributed PHD filter for MTT regardless of the search strategy. This is the limiting factor when it comes to both computational complexity and communication bandwidth as it requires robots to exchange ownership of portions of the PHD density function as the Voronoi partitions shift [6, Algorithm 1], exchange information about targets that may move across the boundaries of Voronoi cells [6, Algorithm 2], and exchange measurements about targets viewed in neighboring Voronoi cells [6, Algorithm 3]. The standard PHD filter equations scale linearly in complexity with the number of targets (*i.e.,* measurements) [6, eq. (4)]. The complexity and communication load of the distributed PHD filter depends upon the situation. For example, the particle exchange step [6, Algorithm 1] depends upon the amount of area changing ownership, which can be large when the entire team is tightly packed into one area (yielding very large Voronoi cells for boundary robots). The PHD update [6, Algorithm 3] scales linearly with the number of robots who can see into one another's Voronoi cells, again making the tightly packed case difficult. When all robots are evenly distributed in the space then the complexity and communication bandwidth will be minimal. Note also the distributed PHD filter assumes that all robots can communicate with their Voronoi neighbors, have lossless communication channels, and ignores the effects of latency. Addressing these real-world factors will be the subject of future work.

The different control policies (Lloyd's algorithm, SA, and PSO) all utilize the resulting PHD estimate to find the goal location. Since robots compute their goal locations using only the PHD within its own Voronoi cell, no additional communication is required for the control computations.[1] Additionally, the control computations are all quite straightforward and have constant complexity with respect to team size and target count.

Lastly, it should be noted that the "particles" used within our SA and PSO algorithms are actually individual robotic agents in the team. Thus, while it is well known that the performance of SA and PSO is dependent upon the number of particles, in our case this simply means it is a function of team size.

[1] This assumes that, per the note in Algorithm 1, line 2, the Voronoi computations are combined with the distributed PHD filter step.

4 Experimental Results

In this section, we conduct a set of simulated experiments using MATLAB so as to 1) evaluate the correctness and 2) demonstrate the efficacy of the proposed distributed estimation and control algorithm by comparing with the distributed methods used in our previous work [6].

The environment is an open 100×100 m area with no obstacles. For simplicity, we represent PHD using a uniform grid of particles with 1 m spacing. The initial weight for each grid is $w_i = 1 \cdot 10^{-4}$ so the initial number of expected targets in the environment is 1. The robots have a maximum velocity of 2 m/s and are holonomic. The sensor collects the data at 2 Hz and the sensor specification are:

$$\begin{aligned} p_d(\mathbf{x} \mid \mathbf{q}) &= \begin{cases} 0.8 & \|\mathbf{x}-\mathbf{q}\| \leq 5\,\text{m} \\ 0 & \text{else} \end{cases} \\ g(\mathbf{z} \mid \mathbf{x}, \mathbf{q}) &= \mathcal{N}(\mathbf{z} \mid \mathbf{x}-\mathbf{q}, 0.25 I_2) \\ c(\mathbf{z} \mid \mathbf{q}) &= 3.66 \cdot 10^{-3} \end{aligned} \tag{9}$$

The expected number of clutter detections per scan is $\int_{FoV} c(\mathbf{z} \mid \mathbf{q})\, d\mathbf{z} = 0.287$. Note, these values do not represent any specific real-world situation, but could represent a team of drones equipped with downward facing cameras. For a real situation, one could follow the procedures outlined in our other work to carefully characterize the sensor models [5, 2].

4.1 Evaluation Metric

To evaluate the results we use the Optimal SubPattern Assignment (OSPA) metric [23], which is widely used in the PHD literature. The OSPA between two sets X, Y, where $|X| = m \leq |Y| = n$ without loss of generality, is

$$d(X,Y) = \left(\frac{1}{n} \min_{\pi \in \mathit{\Pi}_n} \sum_{1}^{m} d_c(x_i, y_{\pi(i)})^p + c^p(n-m) \right)^{\frac{1}{p}} \tag{10}$$

where c is the cutoff distance where we set $c = 10$ m. The $d_c(x,y) = \min(c, \|x-y\|)$ and the $\mathit{\Pi}_n$ is the set of permutations of $\{1,2,3,...,n\}$. We set $p = 1$ as the p-norm. Since the OSPA can fluctuate due to false negative and false positive detections, we use the median of the OSPA over the last 100 s of each trial to obtain a steady estimate in each experiment and measure the variance of the OSPA to evaluate the stability of the tracker.

4.2 Stationary Targets

In our first set of experiments we assume all targets are static, so the transition model is the identity map, the survival model is 1, and the birth model is 0. We use teams 10 to 100 robots (in steps of 10) searching for 10, 30, or 50 targets and

compare the new SA and PSO control methods against our previous work that uses Lloyd's algorithm alone [6]. The target positions are drawn uniformly at random while the robots all begin in a $20 \times 10\,\mathrm{m}$ rectangle at the bottom center of the environment. For each configuration of target number, robot number, and control algorithm we run 10 trials. Note, to ensure consistency in the results, we use identical target distributions across all team/control methods, *i.e.,* there are 10 different randomized target distributions for each configuration.

Tracking Accuracy Figure 1 shows the statistics of the median OSPA error over the ten trials using our original function (PHD), PSO, and SA respectively. We see in all cases that as the size of the team increases and surpasses the number of targets, the OSPA error reaches a minimum and does not decrease, even with the addition of more robots. The OSPA error of around 0.8 indicates that almost all targets have been detected and tracked with higher accuracy than the PHD grid (1 m). This empirical minimum is based on the accuracy of the sensors, the particle spacing, and our method for extracting target locations (*i.e.,* finding peaks in the PHD density function [6]).

We see that the PSO and SA based algorithm both have a much lower OSPA error when the number of robots is smaller than the targets number, with the PSO algorithm having a lower average value than SA or PHD. The difference between the new methods and PHD is especially evident in the 10-robot, 10-target case, where the original method fails to find about half of the targets (*i.e.,* OSPA about halfway between the minimum of 0.8 and maximum of $c = 10$). We also see that the spread in the OSPA values is smallest for the SA case, indicating that it offers the most reliable performance across all three approaches. In summary, PSO offers the lowest average OSPA, SA offers the most consistent results, and the original is not the best in any category.

Coverage We also wish to examine how well each algorithm explores the environment, as this is another indicator of success. Figure 2 show the robots' trajectories in the 10-robots, 10-targets case, where the orange diamonds are the targets location and the lines in different colors are the robots' trajectories. We can observe that in all cases most of the robots eventually find targets and oscillate around them. However, the paths they take to get there are different. Lloyd's algorithm, in Fig. 2a, spends the least time exploring the space, which contributes to it having the highest OSPA error. In this figure, two targets are not detected. The robots move directly to their weighted Voronoi Center and become idle once they have arrived at the goal point. This is a great waste of the searching resource. We see in Fig. 2b that the robots cover much more of the search space in the same amount of time. Thus, the addition of the new local best term (since the global best is identical to Lloyd's), causes the robots to explore more. We also see that the robot paths are relatively smooth, a result of the inertia term in (8) causing the heading direction to remain somewhat consistent. We see in Fig. 2c that these robots take more of a random walk. This is because the annealing process drive the robots oscillate between the best point

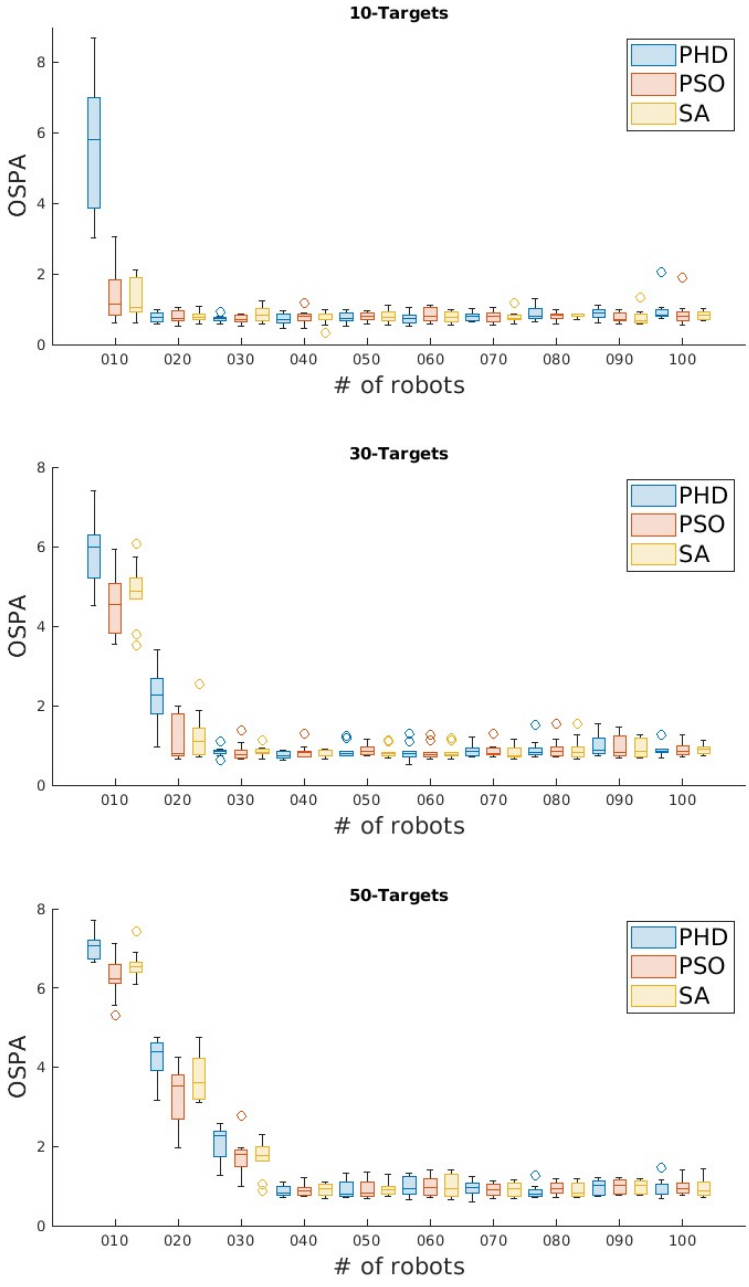
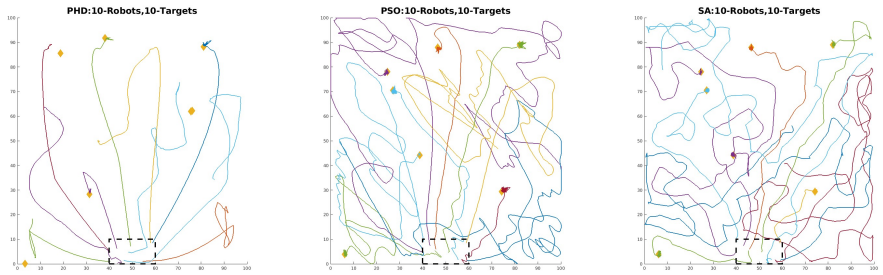


Fig. 1. Boxplots shows the median OSPA statistics over 10 runs for teams of 10-100 robots and 10, 30, and 50 static targets for original PHD, PSO, and SA respectively.



(a) PHD (b) PSO (c) SA

Fig. 2. Robots' trajectories using different control methods. Robots all begin in the dashed box at the bottom of the environment.

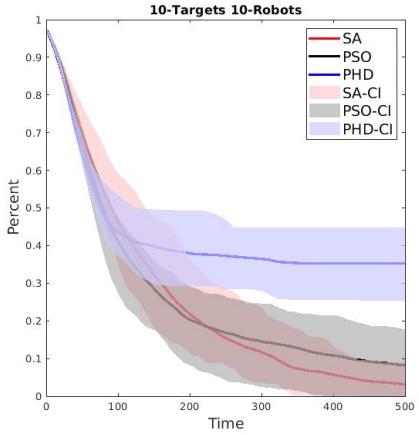


Fig. 3. Percent of undetected area over time over 10 runs for 10-robot, 10-target case, where the solid curves show the mean value and the shaded areas show the range.

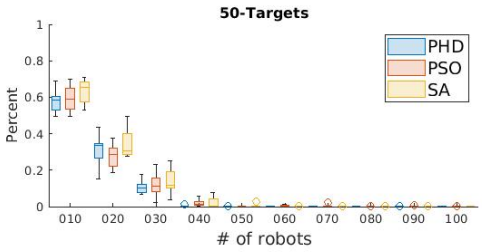


Fig. 4. Boxplots show the percent of undetected area at the end of the experiment over 10 runs for teams of 10-100 robots and 50 static targets for the original PHD, PSO, and SA search strategies.

and the potential point. However, since the temperature keeps decreasing, the oscillation range is smaller than that of PSO.

We can look at these trends in a quantitative manner by plotting the percentage of unexplored area over time for each method in Fig. 3. We see that the average exploration speed is similar across all methods over the first 100 s, with SA showing a larger range. After that, we see how the Lloyd's algorithm exploration levels off around 0.35 due to robots getting stuck. The other methods both have ways of escaping local minima, with SA yielding the highest and most consistent exploration rate as time increases.

Figure 4 shows the statistics for the final unexplored area in the 50 target case. We see here that the trends are slightly different, with SA tending to have the most unexplored area. This seems to occur because the robots frequently get stuck in local minima due to the higher density of targets in the environment. Once the number of robots exceeds the number of targets, all yield similar results and cover more than 99% of the environment for ≥ 60 robots in the allotted time. This is because there are free robots to continue to explore the remainder of the space even when each target has a robot assigned to track it.

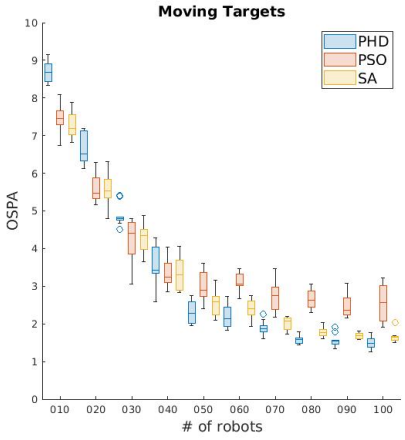


Fig. 5. Boxplots shows the median OSPA statistics over 10 runs for teams of 10-100 robots in tracking moving targets for original PHD, PSO, and SA respectively.

4.3 Dynamic Targets

We also conduct a set of experiments with moving targets. Target have a max velocity 1 m/s and may enter or leave the environment by moving across the boundary. When a new target crosses the boundary, it moves along a random direction. Targets become inactive if they move out of the environment. To

model the birth of the targets, we set the birth PHD to be non-zero near the map boundary (∂E)

$$b(x) = \begin{cases} 5.26 \times 10^{-5} & \|x - \partial E\| \leq 5m \\ 0 & \text{else} \end{cases} \tag{11}$$

This model expects one new target to appear per 10 s. Based on the appearance and disappearance of targets, the number of targets reaches a steady state of around 30 targets regardless of the initial number of targets.

Figure 5 shows the median steady state OSPA value, as measured by the average over the final 200 out of 1000 time steps. We see that in all cases that the OSPA decreases as the number of robots increases and appears to reach an asymptote. However, the PHD method yields the highest OSPA error when the number of robots is small, but has the lowest asymptote when the number of robots is high. We see that the PSO and SA methods are quite similar with 10-40 robots. However, above that we see that PSO reaches a higher asymptote and has a higher variance in the OSPA, meaning it is both less accurate and less reliable than SA. This is similar to the static target case, where Lloyd's algorithm get stuck tracking one target while PSO and SA can both escape these local minima to switch to another target. This escape behavior is very beneficial when there are fewer robots than targets but slightly harmful to tracking performance when there are more robots than targets.

5 Conclusion

In this paper, we examined the performance of different search strategies to solve the MR-MTT problem. All methods share a common structure with two components: 1) an estimation algorithm based on the distributed PHD filter and 2) a control strategy combining the stochastic optimization algorithm with Voronoi-based control. The robots use the estimate from the distributed PHD filter to weight the relative importance of the area within their Voronoi cell and select an action. We compare the standard Lloyd's algorith with two new methods based of standard stochastic optimization algorithms, namely simulated annealing (SA) and particle swarm optimization (PSO). Both these new methods allows robots to trade-off between a global goal and other local areas known to contain targets, unlike Lloyd's algorithm which only uses local information. We see that the new stochastic methods enable the robot team to explore more search area and yield better tracking results when there are fewer robots than targets. In other words, the robots make a better trade-off between exploration and exploitation.

Acknowledgement

This work was funded by NSF grants IIS-1830419 and CNS-2143312.

References

1. Blackman, S.: Multiple hypothesis tracking for multiple target tracking. IEEE Aerospace and Electronic Systems Magazine **19**(1), 5–18 (2004). DOI 10.1109/MAES.2004.1263228
2. Chen, J., Xie, Z., Dames, P.: The semantic PHD filter for multi-class target tracking: From theory to practice. Robotics and Autonomous Systems **149** (2022). DOI 10.1016/j.robot.2021.103947
3. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Transactions on Robotics and Automation **20**(2), 243–255 (2004). DOI 10.1109/TRA.2004.824698
4. Dames, P., Kumar, V.: Autonomous localization of an unknown number of targets without data association using teams of mobile sensors. IEEE Transactions on Automation Science and Engineering **12**(3), 850–864 (2015). DOI 10.1109/tase.2015.2425212. URL `https://doi.org/10.1109/tase.2015.2425212`
5. Dames, P., Kumar, V.: Experimental characterization of a bearing-only sensor for use with the phd filter (2015)
6. Dames, P.M.: Distributed multi-target search and tracking using the phd filter. Autonomous robots **44**(3), 673–689 (2020)
7. Davis, L.: Bit-climbing, representational bias, and test suit design. In: Proc. Intl. Conf. Genetic Algorithm, 1991, pp. 18–23 (1991)
8. Derr, K., Manic, M.: Multi-robot, multi-target particle swarm optimization search in noisy wireless environments. In: 2009 2nd Conference on Human System Interactions, pp. 81–86 (2009). DOI 10.1109/HSI.2009.5090958
9. Deutsch, I., Liu, M., Siegwart, R.: A framework for multi-robot pose graph slam. In: 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR), pp. 567–572 (2016). DOI 10.1109/RCAR.2016.7784092
10. Doitsidis, L., Weiss, S., Renzaglia, A., Achtelik, M.W., Kosmatopoulos, E., Siegwart, R., Scaramuzza, D.: Optimal surveillance coverage for teams of micro aerial vehicles in gps-denied environments using onboard vision. Autonomous Robots **33**(1), 173–188 (2012). DOI 10.1007/s10514-012-9292-1
11. Du, Q., Faber, V., Gunzburger, M.: Centroidal voronoi tessellations: Applications and algorithms. SIAM Review **41**(4), 637–676 (1999). DOI 10.1137/S0036144599352836
12. Fortmann, T., Bar-Shalom, Y., Scheffe, M.: Sonar tracking of multiple targets using joint probabilistic data association. IEEE Journal of Oceanic Engineering **8**(3), 173–184 (1983). DOI 10.1109/JOE.1983.1145560
13. Glover, F.: Tabu search—part i. ORSA Journal on computing **1**(3), 190–206 (1989)
14. Goldhoorn, A., Garrell, A., Alquézar, R., Sanfeliu, A.: Searching and tracking people with cooperative mobile robots. Autonomous Robots **42**(4), 739–759 (2018). DOI 10.1007/s10514-017-9681-6. URL `https://doi.org/10.1007/s10514-017-9681-6`
15. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 vol.4 (1995). DOI 10.1109/ICNN.1995.488968
16. Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598), 671–680 (1983)
17. Mahler, R.: Multitarget bayes filtering via first-order multitarget moments. IEEE Transactions on Aerospace and Electronic Systems **39**(4), 1152–1178 (2003). DOI 10.1109/TAES.2003.1261119

18. Mirjalili, S.: Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems **89**, 228–249 (2015). DOI https://doi.org/10.1016/j.knosys.2015.07.006
19. Murphy, R.: Human-robot interaction in rescue robotics. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **34**(2), 138–153 (2004). DOI 10.1109/TSMCC.2004.826267
20. Qie, H., Shi, D., Shen, T., Xu, X., Li, Y., Wang, L.: Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning. IEEE Access **7**, 146264–146272 (2019). DOI 10.1109/ACCESS.2019.2943253
21. Rizk, Y., Awad, M., Tunstel, E.W.: Decision making in multiagent systems: A survey. IEEE Transactions on Cognitive and Developmental Systems **10**(3), 514–529 (2018). DOI 10.1109/TCDS.2018.2840971
22. Robin, C., Lacroix, S.: Multi-robot target detection and tracking: taxonomy and survey. Autonomous Robots **40**(4), 729–760 (2016)
23. Schuhmacher, D., Vo, B.T., Vo, B.N.: A consistent metric for performance evaluation of multi-object filters. IEEE Transactions on Signal Processing **56**(8), 3447–3457 (2008). DOI 10.1109/TSP.2008.920469
24. Shi, W., He, Z., Tang, W., Liu, W., Ma, Z.: Path planning of multi-robot systems with boolean specifications based on simulated annealing. IEEE Robotics and Automation Letters **7**, 6091–6098 (2022). DOI 10.1109/LRA.2022.3165184
25. Stone, L.D., Streit, R.L., Corwin, T.L., Bell, K.L.: Bayesian multiple target tracking. Artech House (2013)
26. Tang, Q., Yu, F., Xu, Z., Eberhard, P.: Swarm robots search for multiple targets. IEEE Access **8**, 92814–92826 (2020). DOI 10.1109/ACCESS.2020.2994151
27. Vo, B.N., Ma, W.K.: The gaussian mixture probability hypothesis density filter. IEEE Transactions on signal processing **54**(11), 4091–4104 (2006)
28. Vo, B.N., Singh, S., Doucet, A.: Sequential monte carlo methods for multitarget filtering with random finite sets. IEEE Transactions on Aerospace and electronic systems **41**(4), 1224–1245 (2005)
29. Wang, B.: Coverage problems in sensor networks: A survey. ACM Comput. Surv. **43**(4) (2011). DOI 10.1145/1978802.1978811
30. Zhang, C., Lesser, V.: Coordinating multi-agent reinforcement learning with limited communication. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, p. 1101–1108. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2013)