2017 IEEE 56th Annual Conference on Decision and Control (CDC)
December 12-15, 2017, Melbourne, Australia

Simultaneous Search and Monitoring by Unmanned Aerial Vehicles

Haoyu Zhang[1], Sandor Veres[1], Andreas Kolling[2]

Abstract— **Simultaneous Search and Monitoring (SSM) is studied in this paper for a single Unmanned Aerial Vehicle (UAV) searcher and multiple moving ground targets. Searching for unknown targets and monitoring known targets are two intrinsically related problems, but have mostly been addressed in isolation. We combine the two problems with a joint objective function in a Partially Observable Markov Decision Process (POMDP). An online policy planning approach is proposed to plan a reactive policy to solve the POMDP, using both Monte-Carlo sampling and Simulated Annealing. The simulation result shows that the searcher will successfully find unknown targets without losing known ones. We demonstrate, with a theoretical proof and comparative simulations, that the proposed approach can deliver a better performance than conventional foresight optimization methods.**

I. INTRODUCTION

Search and Pursuit Evasion is a problem for a single or multiple robot system trying to detect or capture one or more targets [1]. In practice, relevant problems are mainly divided into two categories: one is searching for unknown targets [2], the other is monitoring known scattered targets to update their information [3][4]. In both categories, the problem formulations are further divided by how fast the agent can outrun the targets, how many targets each agent needs to cover, and how big area of the environment can be sensed. In search missions, the searcher may build a fixed formation to cover the whole area statically [5], or sweep in a fixed pattern [6], or explore dynamically to achieve fastest or best chance of detection [2]. In a monitoring mission the pursuers may track one individual target [7], or cover multiple ones [8], or traverse them in a sequence [3].

In most realistic applications, however, search and monitoring are both required. Unknown targets should be found and known ones should be kept under surveillance. To address the dichotomy in current works, we study a Simultaneous Search and Monitoring (SSM) problem, in which a single UAV searcher is required to search and continuously monitor several targets in a large environment. The pursuer tries to update the location information of as many targets as possible, through searching for unknown targets while monitoring known ones in parallel. At first glance, the search and monitoring solutions appear to be incompatible, and the trade-off between clashing interests of the UAV has previously been formulated as a Task Assignment Problem [9]. In our work, however, the trade-off is transformed into a cooperation, by treating search and monitoring combinatorially under a united objective function. This objective function can better address the objective in SSM problems, and hence can better exploit the potential of the UAV.

The main difficulties in search and monitoring problems relate to the uncertainties around target locations and motions. Many prior publications incorporated the uncertainties into rewards, which were deterministic and predictable with respect to pursuer actions. These rewards can be expected number of detections [10], expected monitoring or service levels [11], overall awareness [12], or information entropy [13]. Thus the resulting objective functions are not stochastic, and generate fixed sequences of actions as solutions. However, when search and monitoring are combined, new contingencies such as a detection of a new target or losing a known one, may greatly change the situation and affect the quality of an old fixed plan. The alternative is either a re-planning approach or an approach that plans for these contingencies. We choose a mixed approach and formulated the problem as a Partially Observable Markov Decision Process (POMDP), which models the uncertain system as a Markov Process, and plans a reactive policy with a look-ahead capability for possible future events.

It has been proven that the optimization problems for many variations of Search, Monitoring, and Pursuit Evasion are NP-hard [14], indicating the computational intractability of this problem. Some past works on POMDP take advantage of the piecewise linear property of the objective function, to do offline computation of an optimal policy [15][16]. These approaches can produce precise optimal policies which are fast to execute. But they require an enumeration of the state space and deal with a specific environment, thus are limited to small problems and are not adaptive to changing environments [17]. Therefore we apply an inflight planning approach, which combines Monte-Carlo Sampling and Heuristics to calculate solutions with a reasonable computational cost. Our method can hence also be interpreted as a stochastically verifiable search and monitoring behaviour of UAVs.

The paper is structured as follows: the basic assumptions and models are described in Section II, and the objective function is built in Section III. The policy and trajectory planning approaches are designed in Section IV and V. The simulation results, conclusion and future work are shown in Section VI and VII.

[1] Haoyu Zang is a research student and Sandor M. Veres is a Professor at ACSE, The University of Sheffield, corresponding author, s.veres@sheffield.ac.uk
[2] Andreas Kolling is a Principal Robotics Scientist at iRobot Corp. akolling@irobot.com

II. Target and Pursuer Modelling

A. Target Modelling

In a discretized arena $\varsigma = \{c_{i,j} | i = 1,2,...,n_x, j = 1,...,n_y\}$, where $c_{i,j}$ denote grid cells, there are n sparsely scattered ground targets and one aerial pursuer. Assume that n is known to the pursuer, and each target is distinguishable and is assigned with an ID $\lambda \in \Lambda$. Λ is the set of all the targets. Each cell can contain only one target and one pursuer. Both the pursuer and targets can move only between neighbouring cells. For a pursuer in $c_{i,j}$, its sensor footprint is the area $\Delta = \{c_{i+a,j+b} | a,b \in \{-k,-k+1,...,0,...,k\}\}$. Without losing generality, we assume that $n_x = (2k+1)L, n_y = (2k+1)M$. Thus if agent visits cells $C_s = \{c_{(2k+1)l-k,(2k+1)m-k} | l = 1,2,...,L, m = 1,2,...,M\}$, the whole environment can be swept by sensor footprint.

As the environment is partially observable, the target location $x_\lambda(t)$ is estimated by a probability map. Let $\hat{\mathrm{P}}_\lambda(c,t|Y_t)$ be the estimated probability distribution of target λ in cell c at time t, given Y_t which is the set of measurement up to time t. The whole environment is shown in Figure 1.

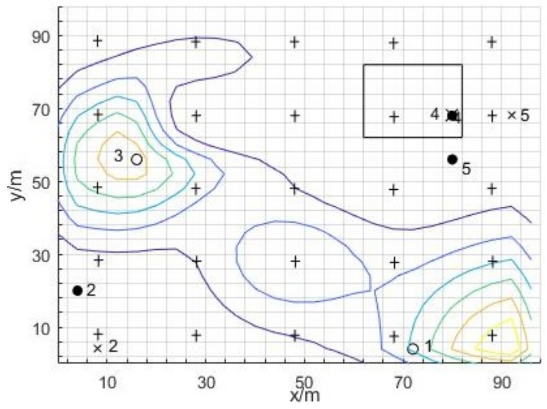

Fig. 1. Example of the environment, which are partitioned by grid cells. Circles denote the targets, among which the filled ones are known targets. The crosses are the estimated locations of known targets. The numbers label the target IDs. The plus signs denote cells in C_s. The rectangle is the agent sensor footprint. The contour denotes the probability distribution of all unknown targets

For the pursuers to update $\hat{\mathrm{P}}_\lambda(c,t|Y_t)$, we apply the Bayesian formulation, which is based on the work of [2], [18], and [19]. Let $t_- = t-1, t_+ = t+1$. And let $N(c)$ denote the neighbouring area of c. $\pi(c|c')$ is the transition function representing the probability if target moves from c' to c in a time step, where

$$\pi(c|c') = \begin{cases} p_s & \text{if } c = c' \\ p_{c|c'} & \text{if } c \in N(c') \\ 0 & \text{else} \end{cases} \tag{1}$$

$p_{c|c'}$ is the probability that the target moves from c' to a neighbouring cell $c \in N(c')$. p_s is the probability that target stays unmoved. And $p_s + \sum_{c \in N(c')} p_{c|c'} = 1$

$p(y_t|c)$ denotes the probability density function of sensing, indicating the probability of possible individual measurement at time t given that the target is at c, thus

$$p(y_t|c) = \begin{cases} p & \text{false positive} \\ 1-p & \text{true negative} \\ q & \text{false negative} \\ 1-q & \text{true positive} \end{cases} \tag{2}$$

Based on the above, the Bayesian formulation of the updating law for target estimation is as follows [2][18][19]:

1) Prediction. Compute prediction using the prior probability distribution $\hat{\mathrm{P}}_\lambda(c',t_-|Y_{t_-})$, the transition function (1), and the Chapman-Kolmogorov equation

$$\hat{\mathrm{P}}_\lambda(c,t|Y_{t_-}) = \sum_{c' \in \varsigma} \pi(c|c') \hat{\mathrm{P}}_\lambda(c',t_-|Y_{t_-}) \tag{3}$$

2) Correction by observation. Update the prediction for cells which are being observed, using Bayes' theorem

$$\hat{\mathrm{P}}_\lambda(c,t|Y_t) = \frac{\hat{\mathrm{P}}_\lambda(c,t|Y_{t_-}) p(y_t|c)}{\sum_{c' \in \Delta} \hat{\mathrm{P}}_\lambda(c',t|Y_{t_-}) p(y_t|c')} \tag{4}$$

3) Correction by inference. For cells outside of sensor footprint, the prediction can be corrected using the fact that $\sum_{c \in \varsigma} \hat{\mathrm{P}}_\lambda(c,t|Y_t) dc = 1$

$$\hat{\mathrm{P}}_\lambda(c,t|Y_t) = \hat{\mathrm{P}}_\lambda(c,t|Y_{t_-}) \frac{1 - \sum_{c' \in \Delta} \hat{\mathrm{P}}_\lambda(c',t|Y_t)}{\sum_{c' \in \varsigma/\Delta} \hat{\mathrm{P}}_\lambda(c',t|Y_{t_-})} \tag{5}$$

To simplify path planning, we categorise targets as known and unknown. If the aggregation level of $\hat{\mathrm{P}}_\lambda(c,t|Y_t)$ becomes higher than a upper threshold, λ is known. $\hat{x}_\lambda(t)$ is the estimation of target location $x_\lambda(t)$. Let $\Lambda_t \in \Lambda$ denote the set of known targets at time t. If the aggregation of $\hat{\mathrm{P}}_\lambda(c,t|Y_t)$ is lower than a bottom threshold, or if the agent fails to detect λ when agent traverses $\hat{x}_\lambda(t)$, λ is lost and becomes unknown. Each known target is under monitoring until lost.

For all the unknown targets $\lambda \in \Lambda/\Lambda_t$, let $\hat{\mathrm{P}}_u(c,t|Y_t)$ be their total probability distribution, thus

$$\hat{\mathrm{P}}_u(c,t|Y_t) = \sum_{\lambda \in \Lambda/\Lambda_t} \hat{\mathrm{P}}_\lambda(c,t|Y_t) \tag{6}$$

B. Pursuer Modelling

Assume that the pursuer could fly with a speed constraint and an arbitrarily small turning radius. The location of pursuer at time t is defined as $x_p(t)$.

C. Overall Model

The state space of an agent in SSM is denoted by S, and at time t, each state $s_t \in S = \{\{\hat{\mathrm{P}}_\lambda(c,t|Y_t) | \lambda \in \Lambda\}, \{\hat{x}_\lambda(t) | \lambda \in \Lambda_t\}, x_p(t), \Lambda_t, t\}$. Let a denote the agent action which is its movement between neighbouring cells. And A_s is the set of all possible actions when the agent is in state s. Thus $p(s_{t_+}|s_t,a)$ is a system state transition function, which is a probability distribution over S. The state space is then formulated as a Discrete-Time Markov Chain.

III. Objective Function

As introduced, the requirements for search and monitoring are contradicting as they may demand the agent to fly over different areas at the same time. By taking advantage of the stochastic character of both missions, we choose an objective function to be a unified goal for search and monitoring.

Definition 1: In state s_t, the *belief* of the estimated location $\hat{x}_\lambda(t)$ is the probability that $x_\lambda(t)$ is within $F(\hat{x}_\lambda(t)) = \{c_{i+a,j+b} | a,b \in \{-k,-k_+,...,0,...,k\}, c_{i,j} = \hat{x}_\lambda(t)\}$. The *belief* is denoted by $\tilde{B}_\lambda(s_t)$. $F(\hat{x}_\lambda(t))$ is the sensor footprint shaped area centred at $\hat{x}_\lambda(t)$. $\square$

The rationale of $\tilde{B}_\lambda(s_t)$ is that it provides a lower bound of probability which target λ will be re-detected, if agent visit $\hat{x}_\lambda(t)$ at t. For state s_t, we define $R(s_t) = \sum_{\lambda \in \Lambda_t} \tilde{B}_\lambda(s_t)$ to be the reward for SSM mission. It provides the lower bound for the expected number of targets which can be detected, if $m = |\Lambda_t|$ agents are deployed to reach the estimated location of each known target at time t.

μ denotes an agent policy to decide which action to take, according to current information. s_{t_f} denotes one of the possible terminal states, and $p(s_{t_f}|\mu, s_{t_i})$ is its probability distribution over S given policy μ and initial state s_{t_i}. According to [20], objective function for SSM over time horizon $T = t_f - t_i$ is formulated as the expected average of the rewards for all time steps within time horizon:

$$\begin{aligned} G(\mu, s_{t_i}, t_f) &= E\{\frac{\Delta T}{T} \sum_{t=t_i}^{t_f} R(s_t)\} = \frac{\Delta T}{T} \sum_{t=t_i}^{t_f} E\{R(s_t)\} \\ &= \frac{\Delta T}{T} \sum_{t=t_i}^{t_f} \sum_{s_t \in S} p(s_t|\mu, s_{t_i}) R(s_t) \\ &= \frac{\Delta T}{T} \sum_{t=t_i}^{t_f} \sum_{s_t \in S} p(s_t|\mu, s_{t_i}) \sum_{\lambda \in \Lambda_t} \tilde{B}_\lambda(s_t) \end{aligned}$$

where ΔT is the time step of system.

This objective can be increased by synergies of search and monitoring through cooperation. The agent can search new targets to enlarge Λ_t, or to monitor known ones to increase $\tilde{B}_\lambda(s_t)$. Thus by planning a policy, the agent can do both missions simultaneously for the same goal.

IV. Policy Planning

A. Solution Methods for POMDP

As we have defined the tuple $(S, A_s, p(s_{t_+}|s_t, a), G(\mu, s_{t_i}, t_f))$ for the system, the SSM problem is formulated as a Finite-Horizon Partially Observable Markov Decision Process (POMDP), of which the goal is to plan the policy μ of pursuer, to optimize the objective value.

Lemma 4.1: For the POMDP defined by tuple $(S, A_s, p(s_{t_+}|s_t, a), G(\mu, s_{t_i}, t_f))$, there exists a deterministic history dependent policy $\mu^*(s_t, h_t)$ to decide the action a_t at each time t, which can achieve the optimal objective value. Where $h_t = (h_{t_-}, a_{t_-}, s_t)$ denotes system history. [20]

Some past work applied a policy iteration method to compute $\mu^*(s_t, h_t)$ offline [20]. Other works reduced the dimension of offline enumeration by utilizing the piecewise linear character of the objective function [15][16]. However, these offline methods require enumeration of state space, thus the computation cost grows exponentially with the number of states. Also the planned policy has not been adaptive to the environmental changes, thus might not have been flexible for a scenario with uncertainty in environment [17].

Instead, we apply online planning [17], which explores the reachable states from the initial state until a time horizon, then plans a local policy. The local policy will be implemented till time horizon or the occurrence of certain events, and will then be replanned. This approach focuses on local and current information, thus can be computed online with moderate cost and can be adaptive to environmental changes.

The objective value can be estimated in a recursive way:

$$\begin{aligned} & G(\mu, s_{t_i}, t_f) = \\ & \frac{\Delta T}{T} R_{t_i}(s_{t_i}) + \frac{T - \Delta T}{T} \sum_{s_{t_{i+}} \in S} p(s_{t_{i+}}|s_{t_i}, a) G(\mu, s_{t_{i+}}, t_f) \end{aligned} \tag{7}$$

Thus the vital part of planning is to calculate the expected future rewards $\sum_{s_{t_{i+}} \in S} p(s_{t_{i+}}|s_{t_i}, a) G(\mu, s_{t_{i+}}, t_f)$. The backward induction method [20] is infeasible because of its computational intractability. In [21], some different approximation approaches for expected future rewards are introduced. Amongst those approaches, the foresight optimization is commonly used in relevant problems [4]. It plans a deterministic path which is a fixed sequence of actions [21]. It has been stated in [21] that foresight optimization can guarantee a lower bound of optimal reward.

However, as mentioned in introduction, in our case, because of the sensitivity of the state s and objective function $G(\mu, s_{t_i}, t_f)$ to contingencies such as new detection or failed monitoring, the reactions to future events should be considered in planning. Therefore, we need a trade-off between computational efficiency and optimality. Some heuristics and Monte-Carlo methods will be implemented as follows.

B. Simplifications

To reduce computational complexity, we make the following assumptions/simplifications for planning:

1) **Perfect Sensor Assumption**. We assume that for the sensor model in equation (2), $p = 0, q = 0$.
2) **Contingency Density Assumption**. As the target distribution is sparse, we assume that for each time step, only one contingency may happen. The contingencies can be four kinds of events: 1. detecting a new target, 2. re-detecting a known target, 3. losing a known target, 4. other events.
3) **Probability Distribution Update Simplification**. $\hat{P}_u(c, t|Y_t)$ is estimated by both target dynamics and sensing. In policy planning, for a future time instant, we ignore the influence of sensing on $\hat{P}_u(c, t|Y_t)$.
4) **Location Update Assumption**. Once a known target λ is redetected, $\hat{x}_\lambda(t)$ will be updated. We assume that this update does not dramatically change s_t.

Based on assumption 2), we classify the states in which the agent fails to detect a new target or succeed to monitor a known one as s', and other states as s°. States s° are called branching states.

Based on assumptions 3), we make the simplification that when doing the prediction in planning, the $\hat{\mathrm{P}}_u(c,t|Y_t)$ is a fixed sequence within time horizon, which are predicted only by information at initial time, thus $\hat{\mathrm{P}}_u(c,t|Y_t) = \hat{\mathrm{P}}_u(c,t|Y_{t_i})$. The induced error is compensated by not allowing an agent to search for a location twice within time horizon. And according to 4), in prediction, we do not consider the adaptation to any change of $\hat{x}_\lambda(t)$ although it may be updated by monitoring. Both simplifications prune branching.

It should be noted that all these simplifications only apply to planning, when the agent is estimating future events. It does not apply during the execution of a policy.

C. Concept for Policy Planning

At initial state s_{t_i}, we propose a deterministic trajectory for agent: $\chi = \{x'_p(t)|t = t_i, t_{i+}, ..., t_f, x'_p(t_i) = x_p(t_i)\}$, called base trajectory. $x'_p(t)$ denotes the location that the target is planned to visit at time t. The trajectory will include a set of target locations $X(t_i) = \{\hat{x}_\lambda(t_i)|\lambda \in \Pi(t_i)\}$, where $\Pi(t_i) \in \Lambda(t_i)$ is the set of known targets to be monitored along χ.

Assume that there is a function $\chi^\circ = f(s^\circ, \chi, h_t)$ which maps a branching state s°, current base trajectory χ and system history h_t, to a new base trajectory χ° starting from $x_p \in s^\circ$. Also let $\{a_t|a_t = x_p(t_+), t \in \{t_i, t_{i+}, ..., t_{f-}\}\}$ be the action taken by the agent to decide the next immediate location. Thus we define a policy μ as Algorithm 1:

Algorithm 1: $a_t = \mu(s_t, \chi, h_t)$

```
if $s_t \in s^\circ$ then
  | $\chi^\circ = f(s^\circ, \chi, h_t)$, $\chi = \chi^\circ$
end
$a_t = x_p(t_+) \in \chi$
```

where $h_t = (h_{t_-}, a_{t_-}, s_t)$ denotes system history.

Theorem 4.2: For the POMDP defined by tuple $(S, A_s, p(s_{t_+}|s_t, a), G(\mu, s_{t_i}, t_f))$, there exists a deterministic history dependent policy $\mu(s_t, \chi, h_t)$ defined in Algorithm 1, to be optimal. □
The proof is given in the Appendix A.

After the formulation of Algorithm 1, the approach of policy planning can be achieved in three steps: 1. propose a candidate policy $\mu(s_t, \chi, h_t)$; 2. estimate objective value $G(\mu, s_{t_i}, t_f)$; 3. search among all possible $\mu(s_t, \chi, h_t)$ and find the optimal one. However, in step one, the selection pool of $\mu(s_t, \chi, h_t)$ is enormous considering system state space; and in step two, the possible branchings in estimation are still enormous. Thus, further simplification should be applied in both steps to make them feasible.

D. Heuristic Policy Approximation

To avoid Backward Induction, we propose a heuristic structure of policy, to approximate the optimal policy $\mu^*(s_t, \chi, h_t)$ via adjusting the parameter in that structure.

In a base trajectory χ which traverses target locations $X(t_i)$, the nodes to traverse known targets are called monitoring nodes. At states s', the agent will keep following χ. Thus we define a heuristic function $\chi^\circ = f(s^\circ, \chi, h_t)$ that is only reactive to the states s°:

1) **Detecting a New Target**. If there is a detection of a new target λ at time t_d, then $\Pi(t_d) = \Pi(t_d) \bigcup \lambda$. The remaining part of χ is χ^r. We let $f(s^\circ, \chi, h_t) = \chi^r$, which does not change the original path.
2) **Losing a Known Target**. If a known target λ is lost at time t_d, then $\Pi(t_d) = \Pi(t_d)/\lambda$, and the remaining part of χ is χ^r. Then we refine χ^r in three steps: 1 *Prune*. We remove all the monitoring nodes from χ^r which traverse λ; 2 *Straighten*. For each pruned node, we use a straight line to connect the possible monitoring nodes before and after the pruned one, to replace the original segments of path connecting between them. Thus χ^r is straightened to be χ^{rs}; 3 *Complement*. The straightening may make χ^{rs} shorter than χ^r for a length of l_c. For the remaining monitoring nodes which are not pruned in all previous branchings, we assume that there is a polyline P_l connecting them in their original sequence. We truncate P_l to a length of l_c, and add it to the end of χ^{rs}, then obtain χ^{rsc}.
$\mu_s(s_t, \chi, h_t) = x_p(t_+) \in \chi^{rsc}$ and $\mu_r(s_t, \chi, h_t) = x_p(t_+) \in \chi^r$ are the fixed action sequence policies with respect to χ^{rsc} and χ^r. The objective value, $G(\mu_s, s_{t_d}, t_f)$ and $G(\mu_r, s_{t_d}, t_f)$, can be calculated deterministically. μ_s is to prune monitoring nodes of lost target, to focus on later search and monitoring; μ_r maintain the old route on the contrary. Let $\chi^c = \chi^r$ if $G(\mu_r, s_{t_d}, t_f) > G(\mu_s, s_{t_d}, t_f)$, or $\chi^c = \chi^{rsc}$ if $G(\mu_s, s_{t_d}, t_f) > G(\mu_r, s_{t_d}, t_f)$, which is to compare and choose between two policies. We let $f(s^\circ, \chi, h_t) = \chi^c$ in this case.

The full $f(s^\circ, \chi, h_t)$ is presented in Algorithm 2.

Algorithm 2: $\chi^\circ = f(s^\circ, \chi, h_t)$

```
$\chi^\circ = \chi^r$
if *losing a known target* then
  | calculate $\chi^c$ based on $\chi^\circ$, then $\chi^\circ = \chi^c$
end
output $\chi^\circ$
```

Thus we have built our heuristic reactive policy $a_t = \mu_a(s_t, \chi, h_t)$ based on Algorithm 1 and 2. We will then prove the sub-optimality of this approximation, which is through comparison with foresight optimization. In our application, the foresight optimal policy is a fixed sequence of actions $a_t = \mu_f(s_t, \chi, h_t) = x_p(t_+) \in \chi$, which maintains following the fixed χ regardless of any contingencies.

Lemma 4.3: For a foresight optimal policy μ_f, its estimated objective value is a lower bound of the maximum objective value achieved by optimal policy μ^* [21].

Theorem 4.4: The optimal reactive policy μ_a^* has an better estimated objective value than that of foresight optimal policy. □

The proof is in Appendix B.

Theorem 4.4 shows that the reactive policy μ_a can better approximate the optimal policy compared with foresight optimization. As function $\chi^\circ = f(s^\circ, \chi, h_t)$ is defined, the policy planning is transformed into the path planning of χ. In real application, replanning and revising will be introduced to better adapt to future contingencies. If a known target is successfully monitored, χ will be revised to adapt to new target location. If a new target is detected or a known one is lost, the $\mu_a(s_t, \chi, h_t)$ will be replanned to adapt.

E. Monte-Carlo Estimation of Objective Value

To estimate the objective value of a candidate policy, we apply the Monte-Carlo Sampling method.

We do m cases of samples. In each sample, the agent applies the policy $\mu_a(s_t, \chi, h_t)$, and let branching happens stochastically based on their probability. In each sample $i = 1, \dots, m$, the achieved hindsight objective value $G_i(\mu_a, s_{t_i}, t_f)$ can be computed based on the corresponding events occurred. Thus $G(\mu_a, s_{t_i}, t_f)$ can be approximated by:

$$G(\mu_a, s_{t_i}, t_f) = \sum_{i \in [1,m]} G_i / m \tag{8}$$

With a practical method of estimating the objective value of each candidate policy, we search the best χ by the following path planning algorithm based on Simulated Annealing.

V. Path Planning based on Simulated Annealing

A. Further simplification

A further assumption is made to facilitate planning.

5) **Trajectory Planning Constraint**. We assume that the vertices of a planned path can only be $C_s \bigcup \{\hat{x}_\lambda(t) | \lambda \in \Lambda_t\}$, which are enough to cover the whole environment without undermining performance. The former set of cells are called search cells, and the later are called monitoring cells.

B. Candidate Trajectory Mutation

To generate candidate path χ, we design a mutation function to get neighbouring candidate solutions. Let $\hat{\chi} = M(\chi)$ be the mutation function for a trajectory, which include four kinds of mutations as inspired by [22]: 1. Add: at one position of χ, add a new node. 2 Prune: prune one node from χ. 3 Swap: swap the position of two nodes in χ or swap one node in χ with a new location. 4 Null: keep χ unchanged. Mutation 1-3 are shown in Figure 2

The red triangle is the current agent location. Green vertices and lines denote the planned trajectory. The numbers show the sequence of nodes. The cells with a plus signs are search cells, and the cells with blue solid circles are estimated location of known targets.

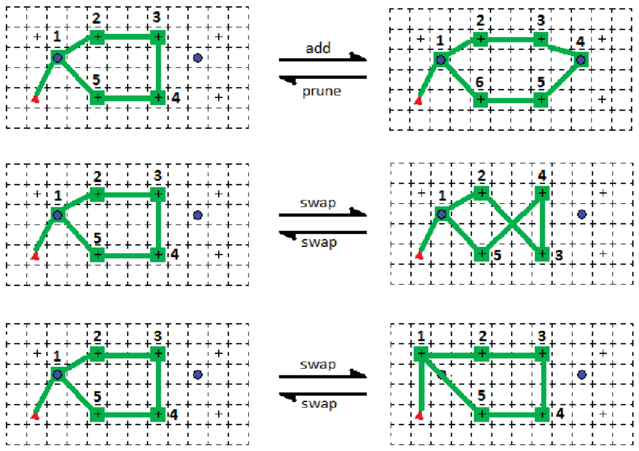

Fig. 2. Mutations on trajectory

Algorithm 3: Simulated Annealing

```
initialization;
$\chi = \{x'_p(t) | t = t_i, t_{i+}, \dots, t_f\}$,$T_e = T_{e0}$,$k_B = const$,$G_c = 0$
while $T_e \geq T_{default}$ do
    $\hat{\chi} = M(\chi)$. $G_p = -G(\mu_a, s_{t_i}, t_f)$, $E = |G_p - G_c|$
    if $G_p > G_c$ then
        $p = exp(-E/k_B T_e)$
        if $random(0,1) \leq p$ then
            accept = true
        else
            accept = false
        end
    else
        accept = true
    end
    if $accept$ = $true$ then
        $G_c = G_p$, $\chi = \hat{\chi}$
    end
    Lower the temperature $T_e$
end
Output $\chi$
```

C. Path Planning based on Simulated Annealing

With the mutation function, the path χ can be planned by a Path Planning algorithm based on Simulated Annealing (Algorithm 3) [23][24]. Simulated Annealing is widely used in path planning and can effectively avoid local minima [24].

Then, the reactive policy can be planned by above steps, which can be executed by the agent for SSM mission.

VI. Simulation

A. Case Study

Consider a $100m \times 100m$ square environment ς, which is discretized into 25×25 cells. The agent sensor can cover 5×5 cells. There are 5 unknown targets and 1 pursuer scattered in the environment. For each time step $\Delta T = 0.2s$, there will be $p_s = 80\%$ probability that a target will stay within the current location. The pursuer can move at speed $V = 20m/s$.

The agent will plan and execute proposed reactive policy μ_a for SSM task, with a time horizon $T = 10s$. When a contingency state s° is reached, or when it has been after T_p long time since last planning, a replanning will be triggered. We set $T_p = 5s < T$ to make the planning more adaptive to environmental changes. The initial target probability distribution $\hat{\mathrm{P}}_\lambda(c,t|Y_t)$ is uniform within the environment, and targets are randomly scattered.

Figure 3,4 and 5 are the snapshots of simulation.

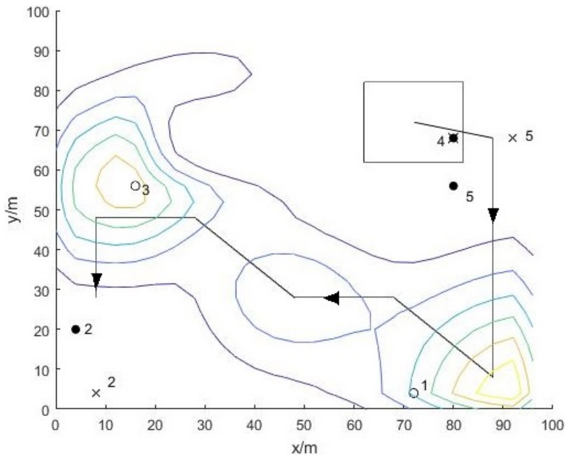

Fig. 3. Search

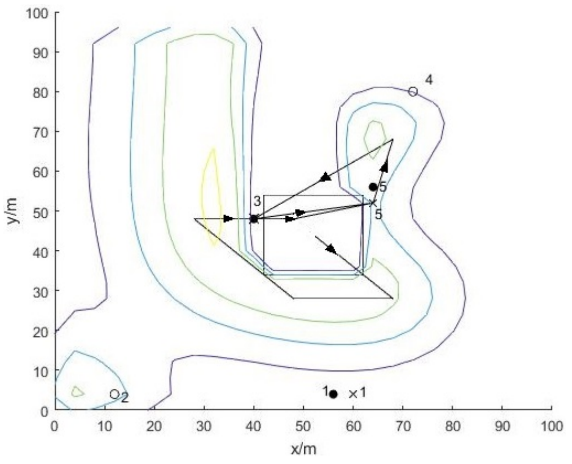

Fig. 4. search and monitoring

The polygons with arrows are the plans of base trajectory. It can be seen from Figure 3 that when there is an area with high distribution of unknown targets, the agent will sweep that area to search. Figure 4 shows that when some targets are known to be nearby, and there is likely to be an unknown target in the neighbouring area, the agent may try to explore the neighbouring unknown area and traverse the known targets, thus combining search and monitoring in the same path. And Figure 5 shows that when the monitoring is saturated, which is when there are some known targets nearby, but there is unlikely to be unknown targets in vicinity,

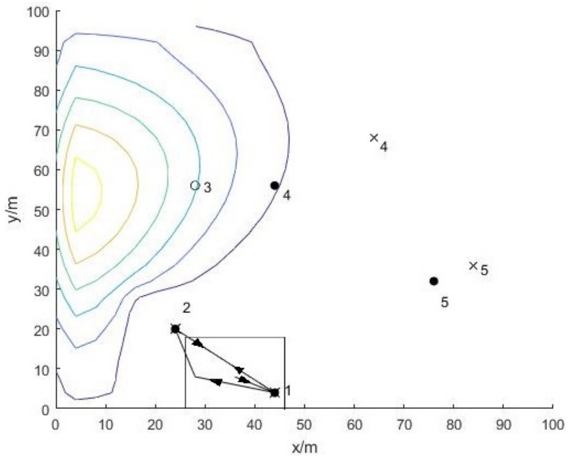

Fig. 5. monitoring

the agent will focus on traversing nearby known targets back and forth.

B. Comparison

We did quantitative study of the SSM, and compared the performance of the proposed reactive policy planning with foresight optimization. Scenarios with n=2, 3, 5 and 7 targets had been studied, and with $p_s = 60, 70, 80\%$. For each scenario, we did 200 cases of simulation for 200 seconds long each. The reward of a scenario is the average reward in each time step, and the average computation time of each planning is recorded as well. We also consider the cases with imperfect sensor, where at each time step, for the sensing of each target, there would be 0.2% chance of false positive or 5% chance of false negative. Figure 6 shows the performances in each scenarios. Each simulation is done by one core of E5 2650V2 processor (2.6 GHz).

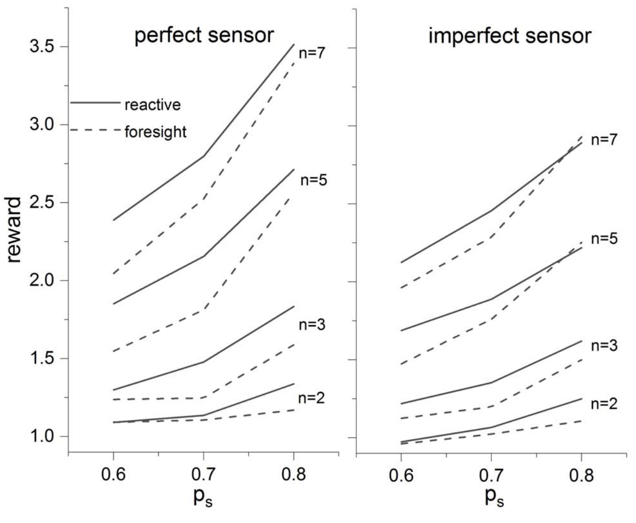

Fig. 6. Comparison of Performances

It can be seen that in most scenarios, the reward of reactive policy is better than foresight optimization. It proves that, if the future contingencies and corresponding reactions are considered during planning, the agent can make better decision about future actions, which is consistent with Theorem 4.4 .

To explain the advantage of reactive policy, we study the following case. In a situation where there are only two known targets and $p_s = 80\%$, we plan the policy using both proposed reactive policy planning and foresight optimization. The base trajectories planned by both methods are shown in Figure 7

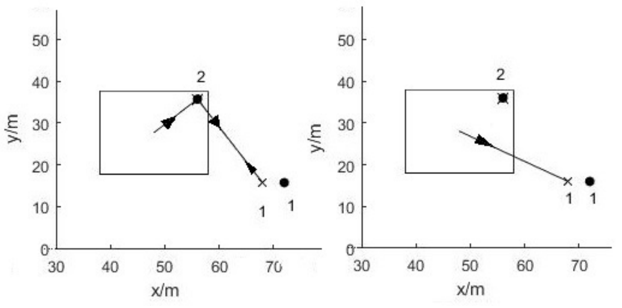

Fig. 7. base trajectory planned by reactive policy planning (left), and foresight optimization (right).

In this case, the foresight optimization keeps the robot to follow only one target, with a estimated objective value of 1.70. On the contrary, the reactive policy planning drives the robot to go back and forth between two known targets, with a better objective value of 1.92. Foresight optimization does not choose the back-and-forth route because if it follows such a fixed route, the agent will not react if one target is lost and will still go back and forth, thus the remaining target will always have a chance to escape between each visit. However, if the policy is reactive, the agent will keep monitoring the remaining target if another is lost, which is more rational with higher estimated objective value.

It is also shown that, in the case of imperfect sensor, there will be a decrease in the performance of both approaches. However, this can be improved by introducing sensor filtering to reduce the influence of false measurement.

According to simulation, each planning of proposed reactive policy takes 0.87 seconds in average, which is much slower than foresight optimization, which takes average 0.01 seconds. Nevertheless, the speed of reactive policy is practical for real time implementation.

VII. Conclusion

Through modelling the system and formulating an objective function, we set up a POMDP framework for a SSM problem, which reconciles search and monitoring missions under a united goal. A novel reactive policy planning method is proposed to solve the problem. Some measures were taken to tackle the computational intractability of finding the best policy and estimating stochastic reward, which include designing a heuristic structure of reactive policy and applying Monte-Carlo sampling. The case study simulation result shows that the proposed approach can effectively search for unknown targets in an initially unknown environment, and can maintain the surveillance of them, with a moderate computational cost. Whenever the monitoring capability is not saturated, the agent will try to find more targets without losing current known ones. We have theoretically proved that our proposed method should work better than conventional foresight optimization and tested that in simulation.

Currently we are testing our methods on multi-rotors and rovers at our Field Robotics Centre near Sheffield, UK.

Appendix I
Proof of Theorem 4.2

Proof: Assume that there is an arbitrary deterministic history dependent policy $\mu(s_t,h_t)$, applied on a arbitrary initial system state s_{t_i}, until terminal time t_f. Let $a_{t_i} = \mu(s_{t_i},h_{t_i})$ be the first action. For all the later states, if $\{s_t|t = t_{i+},...t_f\} \in s'$, let $\{a'_t|t = t_{i+},...t_f, a'_t = \mu(s_t,h_t)\}$ denotes the corresponding sequence of actions taken by policy. Let $\chi = \{a_{t_i}, a'_{t_{i+}},...,a'_{t_f}\}$. If at time t_1, $s_{t_1} \in s^\circ$, the immediate action taken is $a^\circ_{t_1} = \mu(s_{t_1},h_{t_1})$, and let $\{a''_t|t = t_{1+},...t_f, a''_t = \mu(s_t,h_t)\}$ denotes all the corresponding actions for later states if those states belong to s'. Let $\chi^\circ = \{a^\circ_{t_1}, a''_{t_{1+}},...,a''_{t_f}\}$. It can be seen that after recursively applying this process, all possible states and corresponding actions can be reconstructed by χ and all branching χ°, which means that $\mu(s_t,h_t)$ can be fully reconstructed by $\mu(s_t,\chi,h_t)$ in Algorithm 1. According to Lemma 4.1, there exist a optimal $\mu^*(s_t,h_t)$, and thus it can be reconstructed by a $\mu(s_t,\chi,h_t)$, which is also optimal. ■

Appendix II
Proof of Theorem 4.4

Proof: Assume that there is a foresight optimal policy $a_t = \mu_f(s_t,\chi_f,h_t) = x_p(t_+) \in \chi_f$. And build a reactive policy $a_t = \mu_a(s_t,\chi_f,h_t)$ based on Algorithm 1 and 2 which takes χ_f as the initial trajectory. For a agent applying policy μ_a,

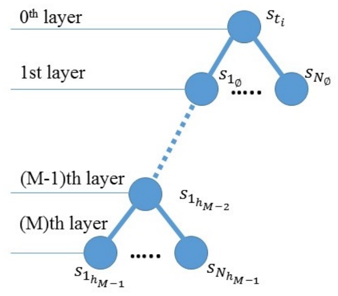

Fig. 8. branching tree

all the branchings are triggered by branching state $s \in s^\circ$. Let $s_{x_{h_{k-1}}}$ denote the state that the xth possible branching in kth layer happens, given h_{k-1} which is the history of prior branchings. $h_0 = \phi$ denotes that there is no priori branching. Let $N_{h_{k-1}}$ denote the number of possible branchings given h_{k-1}. Assuming that there can only be at most M layers of branchings within time horizon. The branching tree is illustrated in Figure 8. Let $t_{x_{h_{k-1}}}$ and $p(x_{h_{k-1}})$ denote the time instant and conditional probability of $x_{h_{k-1}}$ to occur, given h_{k-1}. And $p_{0_{h_{k-1}}}$ denote the conditional probability that $s \in s^{\circ}$ does not happen given h_{k-1}. Let $G_h(t_i, t_f)$ denote the hindsight objective value according to hindsight history from time t_i to t_f, where no branching is made. Let $G_f(\chi_f, s_{t_i}, t_f) = G(\mu_f, s_{t_i}, t_f)$ to be the expected objective value of applying μ_f, and let $G_a(\chi_f, s_{t_i}, t_f) = G(\mu_a, s_{t_i}, t_f)$ to be the objective value of applying μ_a. Thus $G_a(\chi_f, s_{t_i}, t_f)$ can be constructed as follows, including all the possible branchings

$$
\begin{aligned}
& G_a(\chi_f, s_{t_i}, t_f) = p(0_\phi) G_h(t_i, t_f) + p(1_\phi)(\delta_{1_\phi} G_h(t_i, t_{1_\phi}) \\
& + (1-\delta_{1_\phi}) G_a(\chi^{\circ}_{1_\phi}, s_{1_\phi}, t_f)) + \ldots + p(N_\phi)(\delta_{N_\phi} G_h(t_i, t_{N_\phi}) \\
& + (1-\delta_{N_\phi}) G_a(\chi^{\circ}_{N_\phi}, s_{N_\phi}, t_f)); \\
& \ldots \\
& G_a(\chi^{\circ}_{x_{h_{k-1}}}, s_{x_{h_{k-1}}}, t_f) = p(0_{h_k}) G_h(t_{x_{h_{k-1}}}, t_f) + p(1_{h_k})(\delta_{1_{h_k}} \\
& G_h(t_{x_{h_{k-1}}}, t_{1_{h_k}}) + (1-\delta_{1_{h_k}}) G_a(\chi^{\circ}_{1_{h_k}}, s_{1_{h_k}}, t_f)) + \ldots + p(N_{h_k}) \\
& (\delta_{N_{h_k}} G_h(t_{x_{h_{k-1}}}, t_{N_{h_k}}) + (1-\delta_{N_{h_k}}) G_a(\chi^{\circ}_{N_{h_k}}, s_{N_{h_k}}, t_f)); \\
& \ldots k \in [1, M]
\end{aligned}
$$

where $\delta_{x'_{h_k}} = (t_{x'_{h_k}} - t_{x_{h_{k-1}}})/(t_f - t_{x_{h_{k-1}}})$, $\chi^{\circ}_{x'_{h_k}} = f(s^{\circ}_{x'_{h_k}}, \chi^{\circ}_{x_{h_{k-1}}}, h_k)$, and $h_k = \{x_{h_{k-1}}, h_{k-1}\}$.

As there will be no more branching after $x_{h_{M-1}}$, then $G_a(\chi^{\circ}_{x_{h_{M-1}}}, s_{x_{h_{M-1}}}, t_f) = G_f(\chi^{\circ}_{x_{h_{M-1}}}, s_{x_{h_{M-1}}}, t_f)$. Based on the definition of $f(s^{\circ}, \chi, h_t)$, $G_a(\chi^{\circ}_{x_{h_{M-1}}}, s_{x_{h_{M-1}}}, t_f) = G_f(\chi^{\circ}_{x_{h_{M-1}}}, s_{x_{h_{M-1}}}, t_f) \geq G_f(\chi^{\circ}_{x'_{h_{M-2}}}, s_{x_{h_{M-1}}}, t_f)$, where $x'_{h_{M-2}} \in h_{M-1}$, thus

$$
\begin{aligned}
& G_a(\chi^{\circ}_{x_{h_{M-2}}}, s_{x_{h_{M-2}}}, t_f) = p(0_{h_{M-1}}) G_h(t_{x_{h_{M-2}}}, t_f) + p(1_{h_{M-1}}) \\
& (\delta_{1_{h_{M-1}}} G_h(t_{x_{h_{M-2}}}, t_{1_{h_{M-1}}}) + (1-\delta_{1_{h_{M-1}}}) G_a(\chi^{\circ}_{1_{h_{M-1}}}, s_{1_{h_{M-1}}}, t_f)) \\
& + \ldots + p(N_{h_{M-1}})(\delta_{N_{h_{M-1}}} G_h(t_{x_{h_{M-2}}}, t_{N_{h_{M-1}}}) + (1-\delta_{N_{h_{M-1}}}) \\
& G_a(\chi^{\circ}_{N_{h_{M-1}}}, s_{N_{h_{M-1}}}, t_f)) \geq p(0_{h_{M-1}}) G_h(t_{x_{h_{M-2}}}, t_f) + p(1_{h_{M-1}}) \\
& (\delta_{1_{h_{M-1}}} G_h(t_{x_{h_{M-2}}}, t_{1_{h_{M-1}}}) + (1-\delta_{1_{h_{M-1}}}) G_f(\chi^{\circ}_{x_{h_{M-2}}}, s_{1_{h_{M-1}}}, t_f)) \\
& + \ldots + p(N_{h_{M-1}})(\delta_{N_{h_{M-1}}} G_h(t_{x_{h_{M-2}}}, t_{N_{h_{M-1}}}) \\
& + (1-\delta_{N_{h_{M-1}}}) G_f(\chi^{\circ}_{x_{h_{M-2}}}, s_{N_{h_{M-1}}}, t_f)) \\
& = G_f(\chi^{\circ}_{x_{h_{M-2}}}, s_{x_{h_{M-2}}}, t_f) \geq G_f(\chi^{\circ}_{x'_{h_{M-3}}}, s_{x_{h_{M-2}}}, t_f)
\end{aligned}
$$

Applying the same process, it can be seen that

$$
\begin{aligned}
& G_a(\chi^{\circ}_{x_{h_{k-1}}}, s_{x_{h_{k-1}}}, t_f) \geq G_f(\chi^{\circ}_{x_{h_{k-1}}}, s_{x_{h_{k-1}}}, t_f) \\
& \geq G_f(\chi^{\circ}_{x'_{h_{k-2}}}, s_{x_{h_{k-1}}}, t_f) \\
& \ldots \\
& G_a(\chi_f, s_{t_i}, t_f) \geq G_f(\chi_f, s_{t_i}, t_f)
\end{aligned}
$$

Thus it proves that for arbitrary foresight optimal policy $\mu_f(s_t, \chi_f, h_t)$, there will always be a reactive policy to achieve better estimated objective value, which proves the theorem. ■

References

[1] Timothy H Chung, Geoffrey A Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics. *Autonomous robots*, 31(4):299–316, 2011.

[2] Rene Vidal, Omid Shakernia, H Jin Kim, David Hyunchul Shim, and Shankar Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *Robotics and Automation, IEEE Transactions on*, 18(5):662–669, 2002.

[3] Zhijun Tang and Umit Ozguner. Motion planning for multitarget surveillance with mobile sensor agents. *IEEE Transactions on Robotics*, 21(5):898–908, 2005.

[4] Scott A Miller, Zachary A Harris, and Edwin KP Chong. A pomdp framework for coordinated guidance of autonomous uavs for multitarget tracking. *EURASIP Journal on Advances in Signal Processing*, 2009(1):1–17, 2009.

[5] Mac Schwager, Daniela Rus, and Jean-Jacques Slotine. Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research*, 28(3):357–375, 2009.

[6] Timothy G McGee and J Karl Hedrick. Guaranteed strategies to search for mobile evaders in the plane. In *American Control Conference, 2006*, pages 6–pp. IEEE, 2006.

[7] Sonia Martínez and Francesco Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, 2006.

[8] Andreas Kolling and Stefano Carpin. Cooperative observation of multiple moving targets: an algorithm and its formalization. *The International Journal of Robotics Research*, 26(9):935–953, 2007.

[9] Yan Jin, Yan Liao, Ali A Minai, and Marios M Polycarpou. Balancing search and target response in cooperative unmanned aerial vehicle (uav) teams. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(3):571–587, 2005.

[10] Hiroyuki Sato and Johannes O Royset. Path optimization for the resource-constrained searcher. *Naval Research Logistics (NRL)*, 57(5):422–440, 2010.

[11] Dimitris J Bertsimas and Garrett Van Ryzin. A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research*, 39(4):601–615, 1991.

[12] Cheng Song, Lu Liu, Gang Feng, Yong Wang, and Qing Gao. Persistent awareness coverage control for mobile sensor networks. *Automatica*, 49(6):1867–1873, 2013.

[13] Ben Grocholsky, James Keller, Vijay Kumar, and George Pappas. Cooperative air and ground surveillance. *Robotics & Automation Magazine, IEEE*, 13(3):16–25, 2006.

[14] KE Trummel and JR Weisinger. Technical note-the complexity of the optimal searcher path problem. *Operations Research*, 34(2):324–327, 1986.

[15] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

[16] Matthijs T J Spaan, Tiago S. Veiga, and Pedro U. Lima. Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Autonomous Agents and Multi-Agent Systems*, 29(6):1157–1185, 2014.

[17] Stéphane Ross, Joëlle Pineau, Sébastien Paquet, and Brahim Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

[18] Kamil Dedecius. Diffusion estimation of state-space models: Bayesian formulation. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6. IEEE, 2014.

[19] Zhijun Tang and Ümit Özgüner. Sensor fusion for target track maintenance with multiple uavs based on bayesian filtering method and hospitability map. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 1, pages 19–24. IEEE, 2003.

[20] Martin L Puterman. Markov decision processes. discrete stochastic dynamic programming mvspa. 2005.

[21] EKP Chong, C Kreucher, and AO Hero III. Pomdp approximation methods based on heuristics and simulation. *Foundations and Applications of Sensor Management*, 8:95–120, 2007.

[22] Songhwai Oh, Shankar Sastry, and Luca Schenato. A hierarchical multiple-target tracking algorithm for sensor networks. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2197–2202. IEEE, 2005.

[23] Przemek. ANNEAL.M. https://github.com/adgon92/optimalization-project/blob/dd04eafd18d8cc0adb87f880e2421947c2f053e0/examples/anneal.m, 2015.

[24] PJ van Laarhoven and EH Aarts. *Simulated Annealing: Theory and Applications*, volume 37. Springer Science & Business Media, 2013.