Robotics and Autonomous Systems 170 (2023) 104544

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Simultaneous search and monitoring by multiple aerial robots

Haoyu Zhang [a,1], Sandor Veres [a,*,2], Andreas Kolling [b,3]

[a] ACSE, The University of Sheffield, Mappin Street, Sheffield, S1 3JD, United Kingdom
[b] Amazon Robotics, North Reading, USA

ARTICLE INFO

Keywords:
Search
Monitoring
POMDP
semi-Dec-POMDP
Multi-agent cooperation
Heuristic reactive policy planning

ABSTRACT

This paper studies simultaneous search and monitoring (SSM) between multiple unmanned aerial vehicles (UAVs) and multiple moving ground targets. Searching for unknown targets and monitoring known ones are two intrinsically related problems, but they have mostly been addressed in isolation. We combine the two tasks and exploit their interconnection as a synergy rather than a trade-off. We construct the single-robot SSM as a partially observable Markov decision process (POMDP) and the multi-robot SSM as a semi-decentralised POMDP (semi-Dec-POMDP). A novel heuristic reactive policy planning is proposed to solve the POMDP. It is then extended for semi-Dec-POMDP with game-theoretical methods. In simulations and experiments, the searchers will successfully locate unknown targets without losing known ones and cooperate by partitioning their tasks. With theoretical proofs, simulations, and experiments, we demonstrate that our method can perform better than conventional approaches and the state-of-the-art.

1. Introduction

1.1. Introduction of related works and SSM

Search and surveillance problems are relevant for single or multiple robotic systems that search, detect, or capture one or more targets [1]. In practice, most of these problems are divided into two main categories: one is searching for unknown targets [2], based on likelihood distributions of possible object locations; the other is monitoring known targets [3,4], given specific but uncertain target positions. In both categories, problem formulations are usually further refined by target and searcher capabilities, the complexity of the environments, and more detailed objectives. For example, in a search mission, the searcher may build a fixed formation to cover the whole area statically [5], sweep in a fixed pattern [6] [7], or explore dynamically [8] to achieve the fastest or best chance of detection. In a monitoring mission, the robots may track one individual target [9], cover multiple targets [10], or traverse them in a sequence [3] to update the target locations.

Due to the stark contrast that is used in target modelling, in the objectives, and in the approaches taken, the search and monitoring problems are mainly studied separately. In [11], a search and tracking (SaT) problem is addressed. However the search and monitoring are mostly independent since there is only one target. With multiple targets, in order to optimally acquire and update the dynamic information that emerges in realistic applications, cooperation between search and monitoring is necessary. A detected target should be put under surveillance, and a target lost in monitoring should be searched for again. To achieve such a synergy, here we address the problem of simultaneous search and monitoring (SSM), in which a single or multiple UAVs continuously search and monitor several ground targets. The experimental setup of SSM is illustrated in Fig. 1.

As a result of the aforementioned divisions in problem formulations, previous approaches to the merging of search and monitoring have treated the combination as a trade-off and have thus formulated it as a task assignment problem [12–14]. Such separation of tasks makes their integration complicated in look-ahead planning. Hence, the problem is solved by either a myopic method [13,14] or a myopic method with limited look-ahead [12]. By exploiting the connection between search and monitoring in our work, we join them with a united value function. This function allows the use of complex non-myopic planning such as a partially observable Markov decision process (POMDP).

In addition to the task assignment approach, another way of merging search and monitoring is to uniformly model target locations as probability distributions (pd-s), with no differentiation between known or unknown targets. Sparsity of the pd-s represents higher uncertainty, which can be quantified by information entropy [15]. Hence search and

* Corresponding author.
E-mail addresses: zhanghaoyu.buaa@gmail.com (H. Zhang), s.veres@sheffield.ac.uk (S. Veres), kollinga@amazon.com (A. Kolling).
1 Former Ph.D. student at ACSE.
2 Professor at ACSE.
3 Principal Applied Scientist at Amazon Robotics. His contributions to this work were done prior to Amazon Robotics.

https://doi.org/10.1016/j.robot.2023.104544
Received 9 January 2022; Received in revised form 28 August 2023; Accepted 26 September 2023
Available online 6 October 2023

List of Symbols

ς	An area discretised by grid cells
$c_{i,j}$	The cell at ith row and jth column in ς
λ	ID of a target
Λ	Set of all target IDs
Λ_t	Set of IDs of targets known at time t
γ	ID of an agent
Γ	Set of all agent IDs
O_γ	Sensor footprint of agent γ
O	Total sensor footprint of all agents
C_s	Set of search cells
x_λ	Location of target λ
x_γ	Location of agent γ
$\hat{x}_\lambda$	Estimated target location
$\hat{x}_\gamma$	Planned agent location
$Pr(x_\lambda)$	Probability distribution of target λ
P_u	Total probability distribution of all unknown targets
z_t	Agent measurement at time t
Z_t	History of measurements up to time t
p_{fp}	Probability of false positive detection
p_{fn}	Probability of false negative detection
$N(x)$	Neighbouring cells of x
$Pr(x\|x')$	Probability if a target moves from x' to x in a time step
p_s	Probability of a target not moving for a time step
s_t	System state at time t
S	System state space
b_t	Belief state at time t
Δ	Belief state space
a	Action of an agent
θ	Joint actions of all agents
A	Action space
$T(b_t, \theta, b'_{t+1})$	Belief state transition function
$\tilde{B}_\lambda$	Belief probability
$\tilde{B}_l$	Lower threshold of belief probability for losing a target
$\rho(b)$	Reward of SSM at belief state b
V	Value of SSM
π	Policy of an agent
Π	Joint policy of all agents
π^*	The optimal policy
$\hat{\pi}$	An approximation of the optimal policy
$\hat{\pi}_f$	Policy of fixed sequence of actions
$\hat{\pi}_a$	Heuristic reactive policy
t_0, t_f	Initial and terminal time of planning horizon
T_h	Planning horizon
dt	Time step
$b^\bullet$	Non-branching belief state
b°	Branching belief state
χ	Base trajectory
Ψ	Joint base trajectories
f	Branching function
K	Set of known targets to be monitored along current base trajectory
$M(\chi)$	Mutation function of base trajectory
H	Entropy of target location distribution
I	Mutual information of target location distribution and measurements

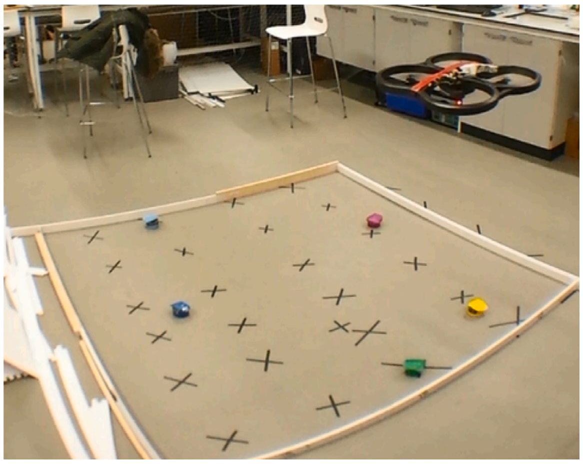
Fig. 1. Experiment setup for simultaneous search and monitoring between one agent (the quadrotor) and five ground targets (the ground robots in different colours) in a square-shaped arena.

monitoring becomes a unified information gathering task to dynamically reduce the overall entropy of the pd-s. Extensive prior research addressed the search mission with frameworks of information gathering in [16–19][20,21]. The approach used in these papers can also be applied in synergistic search and monitoring. Nonetheless, such a formulation does not take advantage of the specific locations of known targets in target monitoring, instead it deals with more generic entropy distributions. As such, it is a very complex problem for look-ahead planning [20,21]. For this reason most past publications on the entropy reduction methods do single-step myopic planning [16–18]. In this paper we categorise targets as known and unknown in order to utilise known target locations to facilitate non-myopic stochastic planning. A comparison of our work with the approach of entropy reduction is also provided.

The methods in this paper naturally divide into two parts: the first is strategy planning of a single robot in a stochastic and partially observable environment. The second is an extension to multi-agent cooperation. To solve the first problem, prior publications prefer to bypass stochastic planning by setting a goal of collecting high-level statistical information about the targets, such as the expected number of detections [22], expected monitoring or service levels [23], or overall awareness [24]. Although the environment in these approaches is usually assumed to be stochastic, the goal is still deterministic and predictable from a fixed plan of the searcher. Another common solution is minimum time search [2,25,26], which reduces the likelihood of zero detection along a plan. Such planning appears to be stochastic, although future detections are deterministically predicted to be zero; thus, it can also be solved by a fixed plan. However, in SSM, we are concerned with the specific information of every target. Each contingency, such as target detection or loss, will trigger a branching event that may dramatically change the situation. Therefore we need a reactive strategy for SSM. We formulate the single-robot SSM problem as a POMDP and solve it with online policy planning. For the second problem, prior publications have studied the cooperative robot search-and-pursuit-evasion problem as a partially observable stochastic game

(POSG) [27] or decentralised partially observable Markov decision process (Dec-POMDP) [28]. However, very few practical online solutions are proposed for POSGs or Dec-POMDPs. Capitalising on the derived POMDP for a single robot, we build a semi-Dec-POMDP for our cooperative SSM to achieve feasible online planning.

The major challenge in this paper is about computational complexity. It is proven in [29] that solving POMDPs is PSPACE-hard, and in [30] Dec-POMDP is shown to be NEXP-complete. To solve POMDPs or Dec-POMDPs, prior approaches applied offline methods, such as value iteration [31–34][35] or policy iteration [28,36,37], to calculate strategies to cover all situations. These approaches can produce good policies that are fast to execute. For the multi-agent case, distributed policies can also be planned offline in a centralised manner. However, the offline methods take hours or days to compute and only deal with a specific environment. Thus, they are limited to small problems and are not adaptive to changing environments [38]. Therefore, we first apply an online planning approach that combines heuristics and Monte Carlo sampling to tackle POMDP in real time. We then apply a game-theoretical approach to extend this method to multi-agent scenarios. Early results of this research have been presented in [39], which addressed the single-robot SSM problem. Following that, we optimised our policy planning code to enable real-time implementation and conducted an experimental study. The early results were extended and investigated in the case of multi-agent SSM in this paper.

1.2. Contributions

In summary, the contributions of this paper are as follows:

(1) A SSM problem is identified and studied, which combines search and monitoring in a single mission.
(2) The problem is formulated as a $\rho POMDP$. We design a novel heuristic reactive policy planning to solve this challenging problem online.
(3) Our solution to $\rho POMDP$ is combined with game-theoretical method to tackle cooperative SSM online.
(4) With theoretical proofs, simulations, and experiments, our approaches are compared with the state-of-the-art and conventional methods, and are proven to be superior in solving SSM problem.

1.3. Structure of paper

The paper is structured as follows: the basic assumptions and models are described in Section 2. The value function is introduced in Section 3. The policy planning approaches for the single-robot SSM are designed in Sections 4 and 5. The extension to multi-agent SSM can be found in Section 6. Simulation and experimental results are shown in Section 7. Section 8 provides the Conclusions.

2. Target and pursuer modelling

2.1. Target modelling

In a discretised arena $\varsigma = \{c_{i,j} | i = 1, 2, \ldots, n_x, j = 1, \ldots, n_y\}$, where $c_{i,j}$ denotes a grid cell, there are n sparsely scattered ground targets and m aerial robots. We use the terms robot and agent interchangeably. Assume that n is known to the robots, and each target is distinguishable and is assigned an ID $\lambda \in \Lambda$. Λ is the set of all target IDs. The robots are indexed by γ and their set is denoted by Γ. The time t is discrete, with a fixed time step. At each time step, both the agents and the targets can make a move between neighbouring cells, and each robot can take a measurement within its sensor footprint. There can only be a maximum of one target at each cell. For a robot γ at position $c_{i,j}$, its sensor footprint is the area $O_\gamma = \{c_{i+a,j+b} | a, b \in \{-k, -k+1, \ldots, 0, \ldots, k\}\}$. $O = \{O_\gamma | \gamma \in \Gamma\}$ is the total sensor footprint of all robots. Without losing

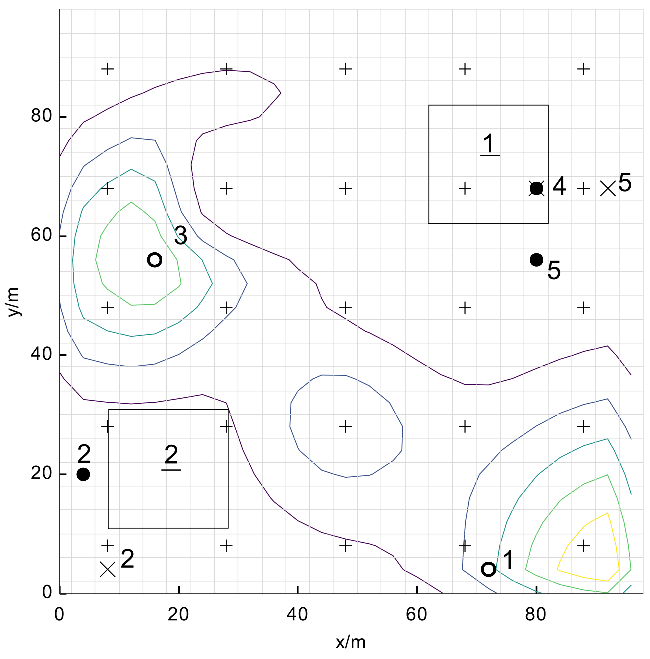

Fig. 2. Example of the environment that is partitioned into grid cells. Circles denote the targets; the filled circles are known targets. The rectangles are the agents' sensor footprints. The crosses (×) are the estimated locations of known targets. The numbers label the target IDs, and the underlined numbers label the agent IDs. The plus signs (+) denote cells in C_s. The level lines indicate the probability density of unknown targets, where the lighter colours indicate higher probability density.

generality, we assume that $n_x = (2k+1)L, n_y = (2k+1)M$, where L and M can be any positive integers. Consequently, if the agents were to visit the set of cells $C_s = \{c_{(2k+1)l-k,(2k+1)m-k} | l = 1, 2, \ldots, L, m = 1, 2, \ldots, M\}$, then the whole environment could be swept by their sensor footprint. Cells in C_s are called 'search cells'.

We assume that agents can share information about their own locations and the detection of targets with each other in real time. In many practical applications it is reasonable to assume that locating aerial teammates and exchanging information with them is not difficult to technically arrange. It is also reasonable to assume that target detections are rare and the communication bandwidth requirements for sharing detection are low. Given this, we assume all agents share the same knowledge about targets. Later on we will consider the case where there is a maximum range of communication and each agent maintains different knowledge.

Because of the limited ability of the robots to take measurements, the location of a target λ at time t, denoted by $x_\lambda(t)$, is estimated by the robots as a probability map. Let $Pr(x_\lambda | Z_t)$ be the jointly estimated probability of target location x_λ, given $Z_t = \{z_t, z_{t-1}, z_{t-2}, \ldots\}$ as the history of joint measurements by all robots. $z_t(\lambda) = \{c | no\ detection\}$ denotes the joint measurement, where $c \in O$ and $\lambda \in \Lambda$. It indicates that target λ is either detected at a location c at time t, or not detected. An environment example is shown in Fig. 2.

We apply a Bayesian formulation for the robots to update $Pr(x_\lambda | Z_t)$ [8,40,41]. Let $N(x)$ be the set of cells neighbouring x. $Pr(x|x')$ is the transition function representing the probability that a target moves from x' to x in one time step, where

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

$p_{x|x'}$ is the probability that a target moves from x' to a neighbouring cell $x \in N(x')$. p_s is the probability that target stays in the same cell, hence $p_s + \sum_{x \in N(x')} p_{x|x'} = 1$. $Pr(z_t|x)$ denotes the probability density of sensing, which is defined in Eq. (2):

$$Pr(z_t|x) = \begin{cases} 1 - p_{fp} & \text{if } x \notin O \text{ and } z_t = no\ detection \\ p_{fp}/|O| & \text{if } x \notin O \text{ and } z_t \neq no\ detection \\ p_{fn} & \text{if } x \in O \text{ and } z_t = no\ detection \\ 1 - p_{fn} & \text{if } x \in O \text{ and } z_t = x \\ 0 & \text{if } x \in O \text{ and } z_t = x' \neq x \end{cases} \tag{2}$$

where x is the presumed target location, and $|O|$ is the size of O. p_{fn} and p_{fp} are probabilities of false negative and false positive.

Based on the above, the Bayesian updating of target estimation is as follows [8,40,41]:

(1) *Prediction*. Compute the prediction using the prior probability distribution $Pr(x(t-1)|Z_{t-1})$, the transition function (1), and the Chapman–Kolmogorov equation

$$Pr(x(t)|Z_{t-1}) = \sum_{x' \in \varsigma} Pr(x|x')Pr(x'(t-1)|Z_{t-1}) \tag{3}$$

(2) *Update*. The prediction can be updated by current measurement information using Bayes' theorem:

$$Pr(x(t)|Z_t) = \frac{Pr(x(t)|Z_{t-1})p(z_t|x)}{\sum_{x' \in \varsigma} Pr(x'(t)|Z_{t-1})p(z_t|x')} \tag{4}$$

To simplify planning, we categorise targets as known and unknown. An unknown target is set to be known once being detected. $\hat{x}_\lambda(t)$ is the estimation of target location $x_\lambda(t)$. For simplicity, we set $\hat{x}_\lambda(t)$ to be the last measured location of target λ, which is defined in Eq. (5).

$$\hat{x}_\lambda(t) = \begin{cases} z_t(\lambda) & \text{if } z_t(\lambda) \neq no\ detection \\ \hat{x}_\lambda(t-1) & \text{if } z_t(\lambda) = no\ detection \text{ and } \lambda \in \Lambda_t \\ \emptyset & \text{if } \lambda \notin \Lambda_t \end{cases} \tag{5}$$

where $\Lambda_t \in \Lambda$ denotes the set of known targets at time t. If the aggregation level of $Pr(x_\lambda|Z_t)$ is lower than a threshold after a known target λ having been unattended for too long, or if an agent fails to detect λ when it traverses $\hat{x}_\lambda$, λ is lost and becomes unknown. The aggregation level of $Pr(x_\lambda|Z_t)$ will be defined in Section 3. Each known target is said to be under monitoring until it is lost. For all the unknown targets $\lambda \in \Lambda \setminus \Lambda_t$, let $P_u(x(t)|Z_t) = \sum_{\lambda \in \Lambda \setminus \Lambda_t} Pr(x_\lambda(t)|Z_t)$ be their total probability distribution.

2.2. Pursuer modelling

We assume that all robots move with bounded speed and arbitrarily small turning radius. The location of robot γ at time t is defined as $x_\gamma(t)$.

2.3. Overall model

The system state is defined as $s_t = \{\{x_\lambda(t)|\lambda \in \Lambda\}, \{x_\gamma(t)|\gamma \in \Gamma\}, t\} \in S$, where S is the state space. Let $a_\gamma = x_\gamma(t)$ denote the action of agent γ, which is its movement to a neighbouring location $x_\gamma(t)$. The robot speed and time step are both constant, however the distances between adjacent grid cells in diagonal and non-diagonal directions are different. Thus in later sections, we formulate the action backwards by proposing a trajectory first, then dividing it by the fixed time step to get a sequence of discrete locations w.r.t. time, thus the grid cell each location falls into is the action $a_\gamma = x_\gamma(t)$. A_γ is the action space of γ, and $A = \{A_\gamma|\gamma \in \Gamma\}$. Let $\theta = \{a_\gamma|\gamma \in \Gamma\}$ define the joint actions of all the agents.

The state transition function $Pr(s_{t+1}|s_t, \theta)$ can be determined by the agent actions and the target transition function (Eq. (1)). Thus the system state space is a discrete-time Markov chain. However, the system state can only be partially observed by the agents, and the SSM is an information gathering problem which does not directly deal with the state. A novel $\rho POMDP$ is proposed in [34] and is further studied in [35]. In this formulation, the problem is oriented to the belief state rather than state, and is focused on having a better estimation of the environment. This suits the requirements of SSM, thus we will build our problem based on the framework of $\rho POMDP$.

The belief state is defined as $b_t = \{\{Pr(x_\lambda(t)|Z_t)|\lambda \in \Lambda\}, \{\hat{x}_\lambda(t)|\lambda \in \Lambda_t\}, \{x_\gamma(t)|\gamma \in \Gamma\}, \Lambda_t, t\} \in \Delta$, where Δ is the belief state space. The belief state is shared among all agents. Given the state transition function and observation functions (Eqs. (2), (3), (4), and (5)), the transition function for belief space can also be determined: $T(b_t, \theta, b'_{t+1}) = Pr(b'_{t+1}|b_t, \theta)$. Hence, the belief state space is also formulated as a discrete-time Markov chain.

3. Value function

For a multi-task planning problem, the value function is constructed to define the goal of all agents, and can also be used to set the relationship between each task. One intuitive formulation is to build a separate reward for each task, then combine them in a trade-off, as it is done in task assignment problems [12–14]. Such a trade-off fails to consider that both search and monitoring increase the up-to-date target information in different ways. Thus there is the need for a clear and explainable united value function that can enable complex planning for synergistic SSM.

We set the total certainties of the known targets to be the value of SSM. The specific locations and respective confidence of known targets are directly useful for other requirements, such as capture or rescue. Thus, such a value is an accurate quantification of reward. Furthermore, to increase this reward, searching for new targets and monitoring detected ones are both expected, and each target may have to be addressed by both search and monitoring at different times. Hence this reward encourages cooperative efforts rather than separate ones.

Definition 1. At belief state b, the belief probability of the estimated location $\hat{x}_\lambda$ is the probability that x_λ is within the region $F_k(\hat{x}_\lambda) = \{c_{i+a,j+b}|a, b \in \{-k, -k+1, \dots, 0, \dots, k\}, c_{i,j} = \hat{x}_\lambda\}$. The belief probability is denoted by $\tilde{B}_\lambda(b)$. $F_k(\hat{x}_\lambda)$ is the sensor-footprint-shaped area that is centred at $\hat{x}_\lambda$.

The rationale of $\tilde{B}_\lambda(b_t)$ is the lower bound of the probability that target λ will be re-detected if an agent visits $\hat{x}_\lambda(t)$ at time t. We define $\rho(b_t) = \sum_{\lambda \in \Lambda_t} \tilde{B}_\lambda(b_t)$ to be the reward function for the SSM mission at belief state b_t. It provides the lower bound of the expected number of targets to be detected, if $m = |\Lambda_t|$ agents are deployed to reach the estimated location of each known target at time t. We also let $\tilde{B}_\lambda(b_t)$ to represent the aggregation level of $Pr(x_\lambda(t)|Z_t)$. A low $\tilde{B}_\lambda(b_t)$ means a low belief in the estimated location of a known target. Given a threshold $\tilde{B}_l$, if $\tilde{B}_\lambda(b_t) < \tilde{B}_l$, that target is lost.

Let $\pi_\gamma : \Delta \to A_\gamma$ denote the policy of agent γ, where $a_\gamma = \pi_\gamma(b)$. $\Pi = \{\pi_\gamma|\gamma \in \Gamma\} : \Delta \to A$ is the joint policy of all the agents. By abuse of notation, we let $\theta = \Pi(b)$. According to [42], the value function for SSM is formulated as the expected average reward within the time horizon:

$$V_\Pi(b) = E\{\sum_{t=t_0}^{t_f} \rho(b_t)|b_{t_0} = b, \theta_t = \Pi(b_t)\}$$

$$= E\{\sum_{t=t_0}^{t_f} \sum_{\lambda \in \Lambda_t} \tilde{B}_\lambda(b_t)|b_{t_0} = b, \theta_t = \Pi(b_t)\}$$

where t_0 and t_f are the initial and terminal time of planning horizon.

To improve $V_\Pi(b)$, the agents can either search in areas with large $P_u(x|Z_t)$ to get higher chance of detecting unknown targets, thus enlarging Λ_t; or visit $\{\hat{x}_\lambda(t)|\lambda \in \Lambda_t\}$ to increase each $\tilde{B}_\lambda(b_t)$. This should be balanced in planning.

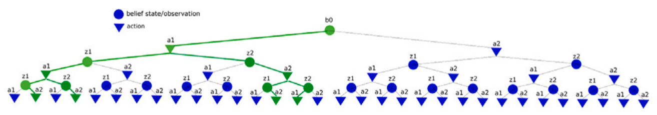
Fig. 3. An example of a search tree (blue) and a policy tree (green). The search tree defines the finite-horizon Markov chain of the belief state. At each belief state, there are two available actions, a_1 and a_2. Each action may lead to two stochastic observations, z_1 and z_2, each of which results in a new belief state. A policy is a decision tree, which is a sub-tree of the search tree. It chooses a specific action at each belief state.

4. Policy planning for a single agent

As an initial step to solve the multi-agent problem, we start with the scenario of a single UAV, and extend the result into multi-UAV case after this.

4.1. Concept for solving POMDP

In a single UAV scenario, the tuple $\langle \Delta, A, T(b_t, \theta, b'_{t+1}), \rho\rangle$ becomes $\langle \Delta, A, T(b_t, a, b'_{t+1}), \rho\rangle$. Then the SSM problem is formulated as a finite-horizon $\rho POMDP$, the goal of which is to plan a pursuit policy π to optimise the value function. Note that this formulation has already transformed the $\rho POMDP$ into a belief Markov decision process (MDP), in which the state is the belief state rather than the actual system state. This is to facilitate later the reactive policy design.

Lemma 4.1. *For the $\rho POMDP$ defined by tuple $\langle \Delta, A, T, \rho\rangle$, there exists a deterministic Markovian policy π^* that can achieve the optimal value. [42]*

The value $V_\pi(b)$ can be estimated by a Bellman equation:

$$V_\pi(b) = \rho(b) + \sum_{b'\in\Delta} T(b, \pi(b), b')V_\pi(b') \tag{6}$$

An example policy is illustrated in Fig. 3.

The optimal policy can be calculated offline or online. As mentioned in the introduction, we do not choose offline methods such as value iteration [31,33,34,43][35] or policy iteration [37,44,45]. Instead, we use online planning [38], which focuses on local and current information and plans a partial policy. The partial policy will be implemented until the time horizon is reached or certain events occur, and then it will be replanned. In our problem, for the ease of incorporating heuristics into planning, we take an online policy improvement approach, described as follows:

(1) Build a structure of policy $\hat{\pi}$ that has the potential to approximate the optimal policy π^*.
(2) Find the $\hat{\pi}^* = argmax_{\hat{\pi}} V_{\hat{\pi}}$, which is the best policy to be achieved with the structure of $\hat{\pi}$.
(3) Implement $\hat{\pi}^*$ until replanning.

Sections 4.2 to 4.5 will provide details on how to build the structure of $\hat{\pi}$. Sections 4.6 and 4.7 will describe the optimisation with the structure of $\hat{\pi}$.

4.2. Fixed sequence of actions vs. reactive policy

In [46], various approaches have been introduced for approximating the optimal policy. Amongst those approaches, the fixed sequence of actions (FSOA) is commonly used in relevant problems. It plans a deterministic trajectory regardless of future events [4,46,47]. It has been stated in [46] that the optimal policy of FSOA can guarantee a lower bound of optimal reward. In addition, in [48], it has been proven that the optimal FSOA is at least as good as an optimal open-loop policy. The FSOA policy is illustrated in Fig. 4.

However, if we compare Figs. 4 and 3, we can see that it is almost impossible for an FSOA policy to precisely approximate an optimal policy. In our case, as mentioned in the introduction, the belief state b and value function V_π are very sensitive to contingencies such as new detections or failed monitorings; thus we need a reactive policy that includes branchings for future events. Feasible online planning requires a compromise between efficiency and optimality. Some simplifications, heuristics, and Monte Carlo methods will be implemented.

In addition, FSOA treats search or monitoring tasks along the action sequence as separate subtasks with no correlation between them, which is very similar to the formulation of task assignment problems. Hence the FSOA policy, which is non-myopic, should also provide an upper bound of reward for myopic task assignment methods as in [12,13], and [14]. Therefore, in simulations and experiments, we will compare our work with FSOA to show both the advantage of reactive policy over FSOA and the superiority of synergistic SSM over the task assignment method.

4.3. Simplifications

To reduce computational complexity, we make the following assumptions and simplifications for planning. It should be noted that all these simplifications only apply to planning, when the agent is predicting future events. They do not apply to the estimation of current information during the execution of a policy.

(1) **Perfect Sensor Assumption**. When doing policy planning and estimating the environment, the robot always assumes that its sensor is accurate with no false positive or false negative.
(2) **Environment Assumption**. Assume that the targets are sparsely scattered, which can easily be outrun by the agent.
(3) **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.
(4) **Contingency Type Assumption**. Assume that event 2 or 3 happens only when the agent is positioned at the estimated location of a known target. Event 1 or 4 happens only when the agent is at other locations.
(5) **Probability Distribution Update Simplification.** $P_u(x(t)|Z_t)$ should be estimated by both target dynamics and sensing. However in policy planning, for a future time instant, we ignore the influence of sensing on $P_u(x(t)|Z_t)$.
(6) **Location Update Simplification**. In policy planning, if a known target λ is predicted to be re-detected at time t, we assume that it moves back to its previously estimated location instantaneously. This means $\hat{x}_\lambda(t) = x_\lambda(t) = \hat{x}_\lambda(t-1)$ if λ is re-detected at t. After t, the target moves freely until the next detection.
(7) **Agent Motion Constraint**. We assume that an agent can only visit a set of cells in $C_s \bigcup \{\hat{x}_\lambda(t)|\lambda \in \Lambda_t\}$, which is a union of search cells and cells at known target locations. The latter set of cells is called monitoring cells.

Given assumption 2, we assume that event 2 and 4 are more likely to happen along the mission, compared with event 1 and 3. Hence, based on assumption 3, we classify the belief states with event 2 and 4 as $b^\bullet$. The other belief states are classified as b° (with event 1 and 3). Belief states b° are called branching belief states. The sets of non-branching and branching belief states are denoted by $\Delta^\bullet$ and Δ°, respectively. Event 1 and 3 are called branching events.

Assumption 4 indicates that there can only be two contingencies at each time step, which are event 1 and 4, or event 2 and 3. Thus the possible belief states at a time step are always a combination of a branching belief state and a non-branching belief state.

With the classification of non-branching and branching events, in Section 4.4, a sequence of non-branching belief states and reasonable actions is defined as base trajectory. Given assumption 2, non-branching events are more likely to happen, thus reducing the chance

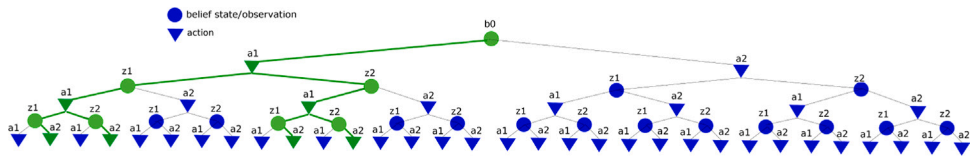

Fig. 4. Policy of fixed sequence of actions (green). Compared with the generic policy in Fig. 3, in this policy, the actions at the same time step are the same, regardless of different belief-states.

of branchings along base trajectory, making it easy for the playout and execution of a policy. The advantage of such classification will be further exploited in Section 4.4.

Based on assumptions 5, in a prediction, $P_u(x(t)|Z_t) = P_u(x(t)|Z_{t_0})$, where Z_{t_0} denotes the history of measurements up to the initial time of planning t_0. When formulating search tree in planning, $P_u(x(t)|Z_{t_0})$ is used to approximate the likelihood of detection at each time step. The induced error is compensated by not planning to do search at a location twice within the time horizon. According to assumption 6, a re-detected target is put straight back to where it was first found. Thus the agent only has to visit the same location each time to monitor a known target, which simplifies assessing a policy of FSOA in Section 4.5. This is justified by the assumption that the main challenge posed by the uncertain movement of a known target is the chance of losing it in the next visit, rather than having to visit it at a different place each time. With assumption 7, an agent can still cover all the unsearched area and known targets, while facilitating the trajectory design in Section 4.7. Assumptions 5, 6 and 7 simplify the transition function $T(b, \theta, b')$, pruning a large number of branchings.

4.4. Policy reconstruction

An online solution to $\rho POMDP$ is studied in [49], which is based on Monte Carlo tree search (MCTS). In [49], a particle filter efficiently predicts the belief state and is combined with MCTS method to build a decision tree for a complex problem in real time. This method is demonstrated to be very versatile to various kinds of $\rho POMDP$ problems, and is the state-of-the-art. However, in our problem, because of the long look-ahead horizon and complicated state space, the search tree can be too complex to explore by even MCTS. Thus we take the forementioned policy improvement method and still receive inspiration from MCTS. Instead of having a decision tree growing from root to leaf, we reconstruct the decision tree with a base trajectory and a branching function. The reconstruction is detailed as follows:

At initial belief state b_{t_0}, we propose a deterministic trajectory for the agent: $\chi = \{\hat{x}_\gamma(t)|t = t_0, t_0+1, \ldots, t_f; \hat{x}_\gamma(t_0) = x_\gamma(t_0)\}$, called the base trajectory. $\hat{x}_\gamma(t)$ denotes the location that the agent is planning to visit at time t. The base trajectory starting from the initial state is called the root base trajectory.

Assume there is a branching function $\chi^\circ = f(b^\circ, \chi)$ that maps a branching belief state b° and current base trajectory χ to a new base trajectory χ° that starts from the current location x_γ. With such a branching function, we define a policy structure $\hat{\pi}$ in Algorithm 1:

Algorithm 1: $a_t = \hat{\pi}(b, \chi)$
```
if $b \in \Delta^\circ$ then
    $\chi = f(b, \chi)$
end
$a_t = \hat{x}_p(t+1) \in \chi$
```

The rationale of Algorithm 1 is that when no branching event happens, the agent simply follows the base trajectory. In the case of a branching event, the agent would find a new base trajectory to follow.

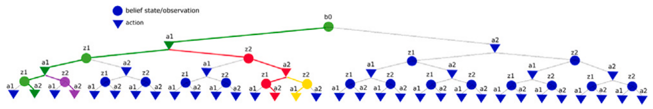

Fig. 5. Policy reconstruction. A generic policy in Fig. 3 is reconstructed in this way. We let observation z_1 correspond to non-branching events. The root base trajectory (green) starts from the root of decision tree and goes along actions and non-branching events. In case of branching events z_2, the later sequence of actions and non-branching events in the decision tree is a new base trajectory, which is in a different colour. Further branchings may also occur along the new base trajectory.

Theorem 4.2. *Given the assumptions made in Section 4.3, for the $\rho POMDP$ defined by tuple $\langle \Delta, A, T, \rho \rangle$, there exists a deterministic policy $\hat{\pi}^*(b, \chi)$, which is defined in Algorithm 1, to be optimal.*

Proof. See Appendix A. □

Theorem 4.2 shows that although a base trajectory is a fixed sequence of actions, the optimal policy π^* can still be fully reconstructed by a root base trajectory and a branching function, which is illustrated in Fig. 5. It then proves that a policy with such a structure has the potential to approximate the optimal policy, which achieves Step 1 in Section 4.1. With such a formulation of policy, multiple simulations called *playouts* can be run along the base trajectory to estimate the value of such a policy. In a *playout*, each branching event triggers a branching function, and the *playout* continues recursively along the new base trajectory till the end of the planning horizon. Thus, if we have an adequate definition of the branching function, we only need to keep modifying the base trajectory at the root and estimate the value through *playouts* until we find the optimal policy with such a structure. This process achieves Step 2 in Section 4.1. Once an optimal policy $\hat{\pi}^*(b, \chi)$ has been found, it can be implemented until a branching event occurs or till the end of the planning horizon. This achieves Step 3.

The rationale for decomposing and reconstructing a policy comes in four parts: first, a base trajectory can be interpreted as a fixed sequence of best actions and likely events, which is intuitive and thus can be configured with heuristic domain knowledge; second, a *playout* can be done by sampling branching events along the base trajectory, which can be much more efficient than branching on every layer of search tree; third, the branching function can be viewed as modifying the base trajectory in the case of exceptions, which can also incorporate heuristic methods; last, the root base trajectory is the actual plan to be executed until a replanning is triggered at branching events. Thus the branching function is only used in the *playout* to estimate the value of a policy, which makes it less demanding for its optimality.

In Section 4.5, we will propose a heuristic reactive branching function. In Section 4.6, *playouts* based on Monte Carlo simulation are introduced. In Section 4.7, the process of modifying and optimising the root base trajectory, which equals the policy optimisation, will be explained. We will do a quantitative comparison with the method proposed in [49] in Section 7.1.4.

4.5. Heuristic reactive branching

We propose a heuristic reactive branching function $\chi^\circ = f_a(b^\circ, \chi)$. Let $K_t \in \Lambda_t$ be the set of known targets to be monitored along χ. The

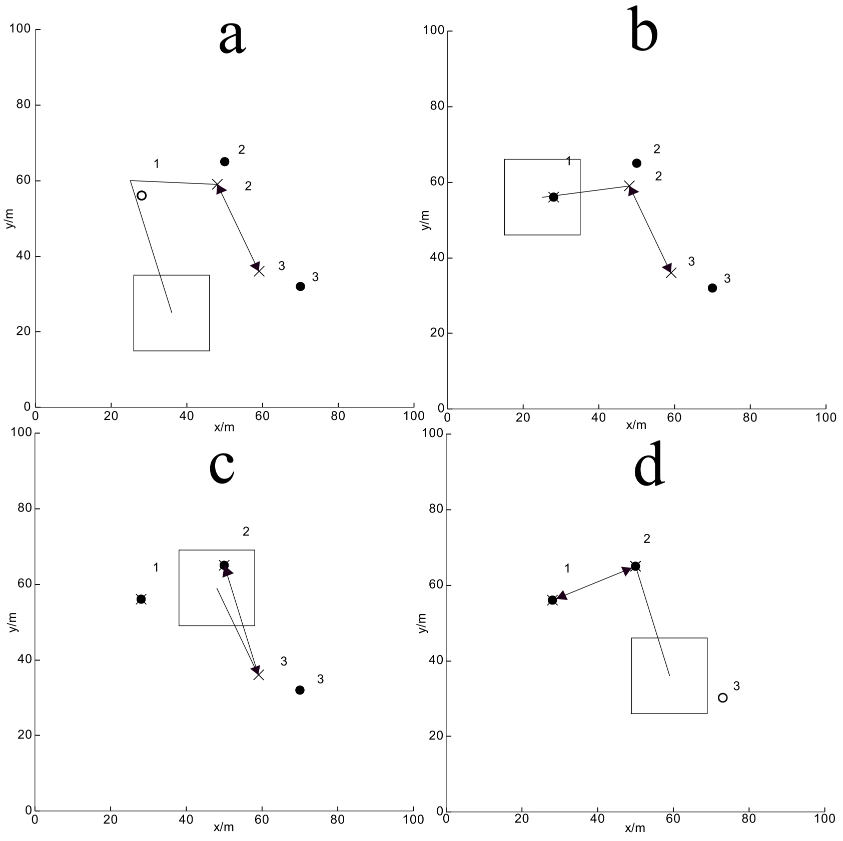

Fig. 6. Heuristic reactive branching function. **a**, **b**, **c** and **d** are four sequential moments in a *playout*.

vertices in χ that traverse known targets are called monitoring nodes. At non-branching states $b^{\bullet}$, the agent will keep following χ. Thus we only define the reactions to two cases of branching belief states b°:

(1) **Detecting a New Target**. If there is a detection of a new target λ at time t, then $K_t = K_t \bigcup \lambda$. The remaining part of χ is χ^r. We let $f_a(b^{\circ}, \chi) = \chi^r$ at this branching state, which does not change the original trajectory.

(2) **Losing a Known Target**. If a known target λ is lost at time t, then $K_t = K_t \setminus \lambda$, and the remaining part of χ is χ^r. χ^r is then refined in three steps:

   (a) *Prune*. We remove all the monitoring nodes from χ^r that would traverse the lost target λ;
   (b) *Straighten*. The possible monitoring nodes before and after each pruned node are connected by a straight line to replace the original segments connecting them. Thus χ^r is straightened to be χ^{rs};
   (c) *Complement*. The straightening may make χ^{rs} shorter than χ^r for a length of l_c. If $|K_t| > 1$, for the remaining monitoring nodes that are not pruned, 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}, which obtains χ^{rsc}. If $|K_t| = 1$, P_l only connects the end of χ^{rs} to the last remaining monitoring node. If $|K_t| = 0$, there is no complement.

$\hat{\pi}_s$ and $\hat{\pi}_r$ are the FSOA policies for the agent to follow χ^{rsc} or χ^r. The values $V_{\hat{\pi}_s}$ and $V_{\hat{\pi}_r}$ can be calculated deterministically. The rationale of $\hat{\pi}_s$ is to prune the monitoring nodes of the lost target to focus on later search and monitoring, while $\hat{\pi}_r$ maintains the old route, to avoid interrupting the original plan. Let $\chi^c = \chi^r$ if $V_{\hat{\pi}_r} > V_{\hat{\pi}_s}$, or $\chi^c = \chi^{rsc}$ if $V_{\hat{\pi}_s} > V_{\hat{\pi}_r}$, which compares and chooses between two options. We let $f_a(b^{\circ}, \chi) = \chi^c$ at this branching state.

The heuristic reactive branching function $f_a(b^{\circ}, \chi)$ can now be presented in Algorithm 2.

Algorithm 2: $\chi^{\circ} = f_a(b^{\circ}, \chi)$

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

The concept of the heuristic reactive branching function is explained in Fig. 6. As defined in Fig. 2, the box in Fig. 6 outlines the agent sensor footprint; the empty or solid circles denote unknown or known targets; the crosses are the estimated location of known targets; the polyline with arrows is the current base trajectory. In Fig. 6 **a**, when there is one unknown target and two known targets, the current base trajectory is to search for the hidden target and visit two known targets back and forth. In **b**, the unknown target 1 is detected at this moment, which is a branching event. The branching function adds target 1 in the set of known targets to be monitored and maintains the remaining part of the base trajectory. Up until this moment, there was no branching event such as detection or loss of a target, thus the current base trajectory was implemented with no branching. In **c**, target 2 is re-detected when the agent is trying to visit it. The estimated location of target 2 is updated (we did not follow assumption 6 in this illustration, for the ease of understanding), while no branching happens because this is also a non-branching event. In **d**, target 3 is lost when the agent is visiting its estimated location. This is a branching event. Target 3 is

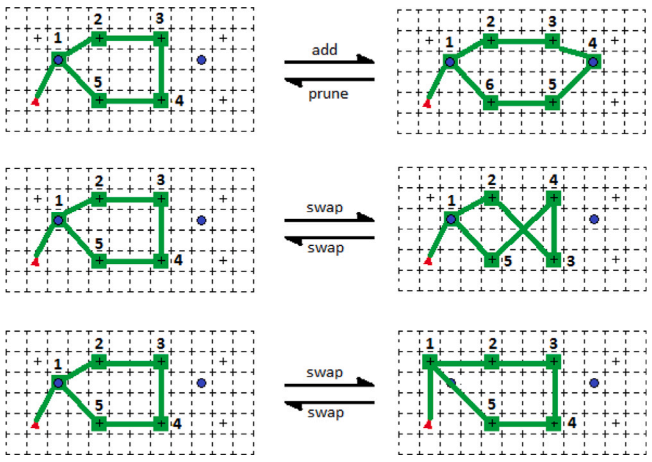

Fig. 7. Mutations on a trajectory. The red triangle is the current agent location. Green vertices and polylines indicate a candidate trajectory. The numbers show the sequence of vertices. The cells with a plus sign are search cells, and the cells with solid blue circles are monitoring cells. For an initial trajectory, mutations can be: add a search or monitoring cell to be a new vertex, or prune one vertex (top); swap two vertices along the trajectory (middle); swap a vertex with a search or monitoring cell (bottom).

removed from the monitoring list (step *prune*). Then, the remaining base trajectory is *straighten*ed into a shorter one along which the agent goes back to target 2. After arriving at 2, to make up for the shortened base trajectory and focus on monitoring the remaining set of known targets, the agent traverses between target 1 and 2, as shown in the graph (step *complement*). However, if $V_{\hat{\pi}_s} < V_{\hat{\pi}_r}$, the base trajectory shown in **c** would be maintained.

Let $\hat{\pi}_a$ denote the heuristic reactive policy that is of the structure defined in Algorithm 1 and contains branching function $f_a(b^\circ, \chi)$ defined in Algorithm 2. We will then prove the advantage of this policy formulation over the FSOA policy. In our application, the FSOA policy is the policy $a_t = \hat{\pi}_f(b) = \hat{x}_p(t+1) \in \chi$ that continues to move to the next position $\hat{x}_p(t+1)$ along a fixed χ regardless of any contingencies.

Theorem 4.3. *The optimal heuristic reactive policy, $\hat{\pi}_a^* = argmax_{\hat{\pi}_a} V_{\hat{\pi}_a}$, has a estimated value greater than or equal to that of the optimal FSOA policy, $\hat{\pi}_f^*$.*

Proof. See Appendix B. □

Theorem 4.3 shows that the heuristic reactive policy $\hat{\pi}_a$ can better approximate the optimal policy, compared with FSOA. Therefore, we let heuristic reactive policy $\hat{\pi}_a^*$ be the solution of our $\rho POMDP$.

For any root base trajectory χ combined with $f_a(b^\circ, \chi)$, we can estimate the value of policy $\hat{\pi}_a$ through *playouts*, which will be introduced in Section 4.6. The final step to work out $\hat{\pi}_a^*$, through optimising the root base trajectory χ, will be described in Section 4.7.

4.6. Monte Carlo estimation of value

We do m samples of *playouts*. In each sample, the agent applies the policy $\hat{\pi}_a$, and we simulate the defined stochastic environment. In each sample $i = 1, \dots, m$, the achieved hindsight value $V^i_{\hat{\pi}_a}$ can be computed based on the corresponding events that occurred. Thus, $V_{\hat{\pi}_a}$ can be approximated by:

$$V_{\hat{\pi}_a} = \sum_{i \in [1,m]} V^i / m \tag{7}$$

4.7. Optimisation of root base trajectory

With the Monte Carlo estimation designed in Section 4.6, we have obtained a mapping from a root base trajectory χ to the value of policy $\hat{\pi}_a$. Then the policy planning $\hat{\pi}_a^* = argmax_{\hat{\pi}_a} V_{\hat{\pi}_a(b,\chi)}$ is transformed into finding $\chi^* = argmax_\chi V_{\hat{\pi}_a(b,\chi)}$, which is a trajectory planning problem. To achieve the online policy improvement approach, we adopted a simulated-annealing-based algorithm for the planning of χ, which consists of a candidate mutation function and an optimisation algorithm.

4.7.1. Candidate trajectory mutation

According to assumption 7 in Section 4.3, the optimal base trajectory χ^* can be approximated by a trajectory traversing between search cells and monitoring cells. Let $\hat{\chi} = M(\chi)$ be the mutation function for a trajectory, which includes four kinds of mutations as inspired by [50]: 1. Add: at one position of χ, add a new vertex. 2. Prune: prune one vertex from χ. 3. Swap: swap the position of two vertices in χ or swap one vertex in χ with a new location. 4. Null: keep χ unchanged. Mutations are shown in Fig. 7.

The mutation function guarantees that an initial guess of χ can be modified incrementally towards χ^* through a sequence of mutations.

4.7.2. Optimisation algorithm based on simulated annealing

Our approach for trajectory optimisation is based on simulated annealing. Simulated annealing is widely used in trajectory planning and can effectively avoid local minima [51]. We have an initial guess of χ that can be a static trajectory in which the agent does not move. Then we impose the mutation function $\hat{\chi} = M(\chi)$ to obtain a neighbouring candidate solution from an initial χ. The reward of such a candidate solution can be acquired from the Monte Carlo simulation. A candidate solution would be accepted as the new solution if it shows a better reward than the initial one, or it may be accepted with a probability if it shows a worse reward. The same process repeats iteratively on the new candidate until the end of the iteration, then the probability of accepting a worse candidate drops in the next round of iterations. The detailed formulation of the optimisation algorithm based on simulated annealing is given in Algorithm 3.

Algorithm 3: Simulated Annealing Algorithm

```
initialization;
$\chi$,$T_e = T_{e0}$,$k_B = const$,$V_c = 0$
while $T_e \geq T_{default}$ do
    $\hat{\chi} = M(\chi)$. $V_p = -V_{\hat{\chi}}$, $E = |V_p - V_c|$
    if $V_p > V_c$ 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
        $V_c = V_p$, $\chi = \hat{\chi}$
    end
    Lower the temperature $T_e$
end
Output $\chi$
```

Note that the simulated annealing algorithm here is an independent component that can be achieved by any suitable algorithm. Thus, it is not our main focus.

4.8. Closed form of heuristic reactive policy planning

Now, the steps of policy planning of SSM can be shown below in a closed form.

(1) Propose an initial root base trajectory χ.
(2) Construct a candidate heuristic reactive policy $\hat{\pi}_a$ based on Algorithm 1, with a branching function $f_a(b^\circ, \chi)$ defined in Algorithm 2.
(3) Do Monte Carlo simulations and obtain $V_{\hat{\pi}_a}$, which is basically V_χ.
(4) Mutate χ with $\chi = M(\chi)$.
(5) Go back to Step 2 and repeat the whole process in a simulated-annealing manner.
(6) Find the optimal χ^* with the highest V_χ, then the optimal policy $\hat{\pi}^*_a(b, \chi^*)$ is obtained.

In a real application, revising and replanning will be introduced to better adapt to future contingencies. If a known target is updated, χ will be revised to cover the new target location. If a new target is detected or a known one is lost, the $\hat{\pi}_a$ will be re-planned to adapt.

In addition, when a known target is lost, if its probability distribution is still very certain, it may soon be recovered by the re-planned policy. However this is not considered in the above branching function. To compensate, when estimating the chance of detecting a known target, we expand the potential sensor footprint to $\tilde{O} = \{c_{i+a,j+b} | a, b \in \{-k-1, -k, \dots, 0, \dots, k+1\}\}$. This can encourage monitoring, with less fear of losing a known target.

5. Planning with entropy reduction method

As mentioned in the introduction, information gathering can be an alternative way of formulating relevant problems. Let $Og_t(c, \lambda) = \{target | no\ target\}$ denote the location distributions of all targets at time t, where $c \in \varsigma$ and $\lambda \in \Lambda$. It indicates whether a certain target is at a location. Then $H(Og_t)$ is the total entropy of Og_t, which is defined in Eq. (8) [15]:

$$H(Og_t) = -\sum_{\lambda \in \Lambda} \sum_{x_\lambda \in \varsigma} (Pr(x_\lambda|Z_t) log Pr(x_\lambda|Z_t) + (1 - Pr(x_\lambda|Z_t)) log(1 - Pr(x_\lambda|Z_t))) \tag{8}$$

The expected future total entropy can be updated by [15]:

$$\begin{aligned} H(Og_t|z_{t+1}) &= H(Og_t) - I(Og_t; z_{t+1}) \\ I(Og_t; z_{t+1}) &= H(z_{t+1}) - H(z_{t+1}|Og_t) \end{aligned} \tag{9}$$

where $I(Og_t; z_{t+1})$ denotes the mutual information of Og_t and potential measurement z_{t+1}. As $H(Og_t|z_{t+1})$ describes the expected uncertainty of target location distributions, reducing $H(Og_t|z_{t+1})$ can also be used in SSM. As mentioned in the introduction, the entropy reduction problem is difficult to solve with look-ahead planning and has generally been addressed using myopic methods [16–18].

From Eq. (9), it can be seen that to reduce the immediate $H(Og_t|z_{t+1})$, the agents should maximise $I(Og_t; z_{t+1})$. Because a perfect sensor assumption has been made in Section 4.3, we can find that $H(z_{t+1}|Og_t) = 0$. Then, the policy is to maximise $H(z_{t+1})$, which is the entropy of immediate measurement. Also, with the perfect sensor assumption, $H(z_{t+1})$ equalises the entropy of target location distributions within its immediate sensor footprint. Therefore, we design a greedy policy for the agent: it always flies to the immediate vicinity where the target uncertainty is the highest. We will compare this policy with our proposed heuristic reactive policy through simulation.

6. Cooperative policy planning for multiple agents

When there are multiple robots, cooperation is necessary to enable synergy between agents and avoid redundant efforts. In order to have a more scalable and robust system, many previous works formulate this problem as POSG or Dec-POMDP, and strive to achieve multi-robot collaboration in a decentralised way with little or no communication. One major challenge for an agent to plan a distributed policy is evaluating other agents' estimation about the environment and estimating their knowledge of each other's knowledge. The latter can be recursive infinitely, which is intractable.

Some works solves Dec-POMDP or POSG with offline planning [28,32,52]. Offline planning for each agent can be done in a centralised way, thus circumventing the difficulty of explicitly estimating each other's knowledge. However, as mentioned in Section 4.1, an offline solution is not practical for our problem, yet very few online solutions without strict assumptions exist [27]. Therefore, we first solve the problem under the assumption of full communication, where all agents share the same knowledge, then extend the method to cases with limited communication.

Although perfect communication has been assumed, we do not want a centralised planner to plan a joint strategy and assign it to each agent [53]. Since sharing the joint policy transfers more information and happens more frequently than only sharing the detection of targets, centralised policy planning will demand higher bandwidth communication and is less feasible and reliable in a non-ideal environment. In [54], a decentralised Monte-Carlo tree search (Dec-MCTS) approach has been applied on multi-robot information gathering problem. The collaboration has been considered in local policy planning of each agent, by sharing local search trees between robots. This method is also challenging on communication capability. Thus instead, we rely more on game-theoretical methods to achieve multi-robot cooperation, without sharing local plans.

In [52], the local policy of each agent is built as a deterministic finite state controller (FSC), and the best local policy is obtained through finding the Nash equilibrium among FSCs. Although it is solved offline, we can apply the concept of Nash equilibrium to online local policy planning.

6.1. Cooperative policy planning based on Nash equilibrium

To simplify the problem, we take the concept of partial open-loop feedback control [48], in which we assume that when doing planning, each agent considers the information and possible strategies of other agents, but the planned local policy will only react to its local observation and will ignore the future locations and measurements of other agents. With such an assumption in the policy planning, we can disentangle the coupling between the contingency reaction of each agent, but the coordination between robots can still be considered.

Assign π_γ as the local policy for agent γ. We let $\Pi_{-\gamma}$ denote the joint strategies of all agents except for γ. Thus for robot γ, its policy planning becomes

$$\pi^*_\gamma = \underset{\pi_\gamma}{argmax} V_{\{\Pi_{-\gamma}, \pi_\gamma\}} \tag{10}$$

This equation works symmetrically for every agent.

For the solution of the cooperative SSM, the key concept is to find a Nash equilibrium among the strategies of all agents [27,28]. In the cooperative case, such equilibrium should also be the optimal joint solution. We take the same approach, by letting each agent find a joint policy Π^* for all agents that can ensure the highest V_{Π^*}, and take $\pi_\gamma \in \Pi^*$ as its local policy. Because the information is shared among agents, we assume that the calculated optimal joint policy is the same for each agent, except for instances in which multiple equilibriums exist, i.e. when some agents are very close or the situation around them is symmetric. Therefore the overall optimal value can be achieved in most cases.

6.2. Local policy reconstruction

Since a local policy π_γ only reacts to local observations, it is of the same structure as the policy for single-agent SSM. According to Section 4.4 and Appendix A, such local policy can also be fully reconstructed with a base trajectory χ_γ and a branching rule. We let π_γ apply the same heuristic branching rule f_a as in Section 4.5. Then a local

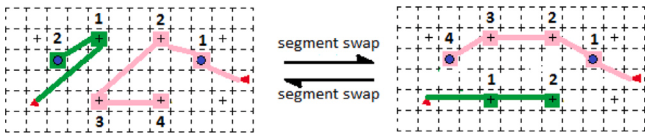

Fig. 8. Joint mutation on trajectories. Two base trajectories (green and pink polylines) can swap segments.

policy π_γ can be represented by a local root base trajectory χ_γ. Hence, Eq. (10) can be rewritten as

$$\chi_\gamma^* = \underset{\chi_\gamma}{argmax} V_{\{\Psi_{-\gamma},\chi_\gamma\}} \tag{11}$$

where $\Psi_{-\gamma}$ is the joint root base trajectory of all agents except for γ.

We let $\Psi^* = \{\Psi_{-\gamma}, \chi_\gamma\}$ be the optimal joint base trajectory. Before designing the detailed planning of Ψ^*, we first prove that the cooperation is achieved advantageously, compared with no cooperation. Let Ψ_o be the joint base trajectory if each agent plans its own policy without any cooperation, in the same way as in single-agent SSM.

Theorem 6.1. *The optimal joint base trajectory Ψ^* obtained through Eq. (11) can achieve an overall value greater than or equal to that of the non-cooperative joint base trajectory Ψ_o.*

Proof. Eq. (11) implies that $\Psi^* = argmax_\Psi V_\Psi$; thus, $V_{\Psi^*} \geq V_{\Psi_o}$. □

Theorem 6.1 demonstrates the advantage of having a distributed coordination for multi-agent SSM.

Since the agents share the same information, $V_{\{\Psi_{-\gamma},\chi_\gamma\}}$ can be estimated by each robot through Monte Carlo sampling in the same way as in the single robot scenario. Thus, we have obtained a mapping from a joint base trajectory Ψ to the overall value $V_{\{\Psi_{-\gamma},\chi_\gamma\}}$. Then, the computation of the Nash equilibrium has become a problem of finding an optimal joint base trajectory Ψ^*.

6.3. Joint root base trajectory planning

To find an optimal joint base trajectory Ψ, we extended the mutation function $\hat{\chi} = M(\chi)$ defined in Section 4.7.1 to the multiple trajectories case. We define a joint mutation function $\hat{\Psi} = M(\Psi)$. $\hat{\Psi} = M(\Psi)$ includes the four independent mutations on an individual $\chi \in \Psi$ that are defined in Section 4.7.1. $\hat{\Psi} = M(\Psi)$ also has a fifth joint mutation: 5. Segment Swap: two base trajectories $\chi_{\gamma_1}, \chi_{\gamma_2} \in \Psi$ swap segments, shown in Fig. 8.

After obtaining the extended mutation function $\hat{\Psi} = M(\Psi)$, the simulated annealing algorithm can also be applied to find the best Ψ.

6.4. Closed form of cooperative policy planning

In summary, the steps for an agent to do cooperative policy planning is presented below.

(1) Propose an initial joint root base trajectory Ψ for all agents.
(2) Construct a candidate local heuristic reactive policy $\hat{\pi} \in \hat{\Pi}$ for each agent based on Algorithm 1, with a branching function f_a defined in Algorithm 2.
(3) Do Monte Carlo simulations and obtain $V_{\hat{\Pi}}$, which is basically V_Ψ.
(4) Mutate Ψ with $\Psi = M(\Psi)$.
(5) Go back to Step 2 and repeat the whole process in a simulated-annealing manner.
(6) Find $\hat{\Pi}^*$ with the highest $V_{\hat{\Pi}}$, which is the optimal joint policy.
(7) $\hat{\pi}_\gamma^* \in \hat{\Pi}^*$ is the optimal local policy of agent γ

In this way, an agent can plan its local optimal policy and implement it until replanning.

6.5. Limited communication range

Even though the cooperative planning is done locally on each agent, communication with unlimited range is still required, and all fellow agents must be planned for when an agent is finding the Nash equilibrium. Therefore, the above approach is still not fully distributed. We can impose a range limit of communication, only within which an agent can be aware of the existence of fellow agents and receive their current measurements, while the belief states or past observations will not be shared. An agent maintains and updates its own belief state based on its measurements and the observations it receives.

This range may either be the result of physical limitations or simply an artificial threshold. It limits the number of fellow agents to be considered in the Nash equilibrium. Therefore, as long as the agents are sparsely scattered, we expect an upper bound on the computational cost for each agent. We call this cooperative planning semi-Dec-POMDP. Its computational efficiency is guaranteed indirectly.

In [55], a Dec-POMDP with communication constraints is studied. In this work, Dec-POMDP is formulated with distributed value functions (DVFs). With DVF, an agent approximates the influence of other agents on its own reward, then plans a local policy independently. The cooperation is achieved in an indirect and decoupled way. Thus even when the communication breaks, an agent does not need to explicitly estimates other agents' knowledge, which is robust and efficient.

In our problem, however, collaboration is intensive thus knowledge of each other's knowledge is necessary. The divergence of belief states between each agent is inevitable. We therefore take the following approach: each agent only considers the neighbouring robots with which it is in communication; It assumes that these agents have the same belief state as itself. Under this assumption, a robot still tries to find a Nash equilibrium with the robots within communication range. The local policy can then be planned by each agent in the same way as in Section 6.4.

Not considering the difference of information held by neighbouring robots should cause a decrease in performance, compared with comprehensively estimating the knowledge of other agents. We take it as a trade-off between scalability and performance, and will evaluate it through simulation.

7. Simulation & experiment

7.1. Simulation of single pursuer SSM

7.1.1. Case study

Consider a 100 m $\times$ 100 m square environment ς that is discretised into 25×25 cells. The agent sensor can cover 5×5 cells. There are five unknown targets and one robot scattered in the environment. For each time step $dt = 0.2$ s, there will be $p_s = 80\%$ probability that a target will stay within the current location. The robot can move at speed $V = 20$ m/s. The agent will plan and execute the proposed heuristic reactive policy π_a for the SSM task, with a time horizon $T_h = 10$ s. When a contingency belief state b° is reached, or when it has been more than T_p time since the last planning, a replanning will be triggered. We set $T_p = 5s < T_h$ to make the planning more adaptive to environmental changes. $\tilde{B}_l = 30\%$ is the lower threshold of *belief probability* for losing a known target. The initial target probability distribution $Pr(x_\lambda|Z_t)$ is uniform within the environment, and targets are randomly scattered.

Figs. 9–11 are the snapshots of one simulation.

The polylines with arrows are the planned root base trajectories. As defined in Section 4.4, the root base trajectory is the default trajectory plan to be executed until a branching event happens. Thus, we use such a fixed trajectory to represent the reactive policy behind it. We can see from Fig. 9 that when there are areas with a high probability distribution of unknown targets, the agent will sweep across these areas to find them. Fig. 10 shows that when some known targets are close,

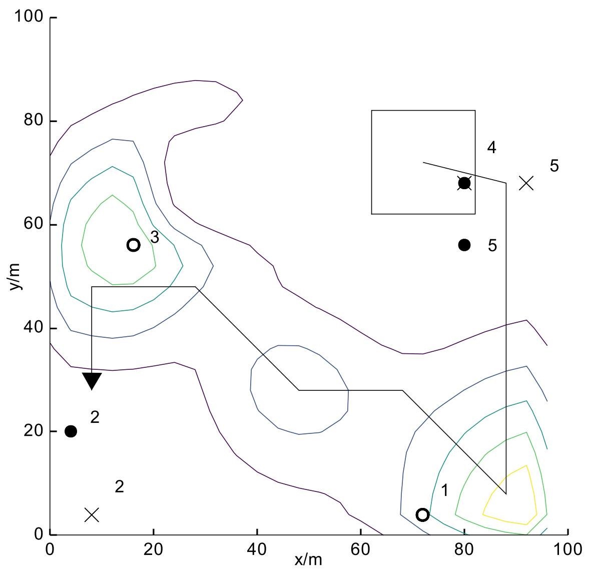

Fig. 9. Searching for unknown targets. As shown by the contour, there are two regions with high confidence of unknown target presence, where targets 1 and 3 exist. The agent chooses to sweep over the two regions to find them.

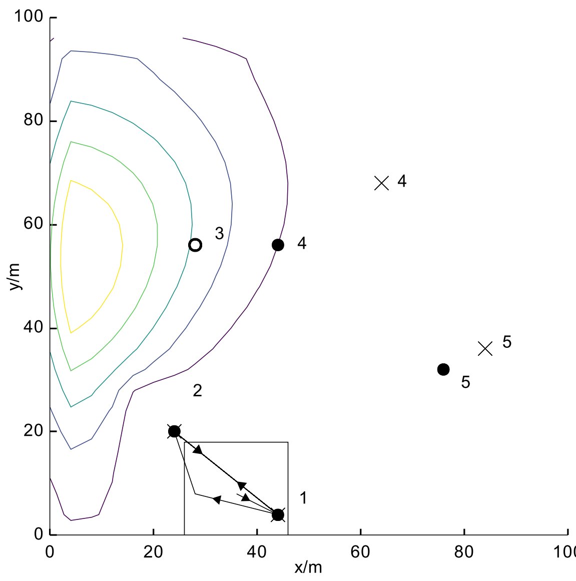

Fig. 11. Monitoring known targets. Two known targets are nearby. However, other known targets and possible areas with unknown targets are far away. In such a case, the agent focuses on monitoring neighbouring known targets by traversing between them.

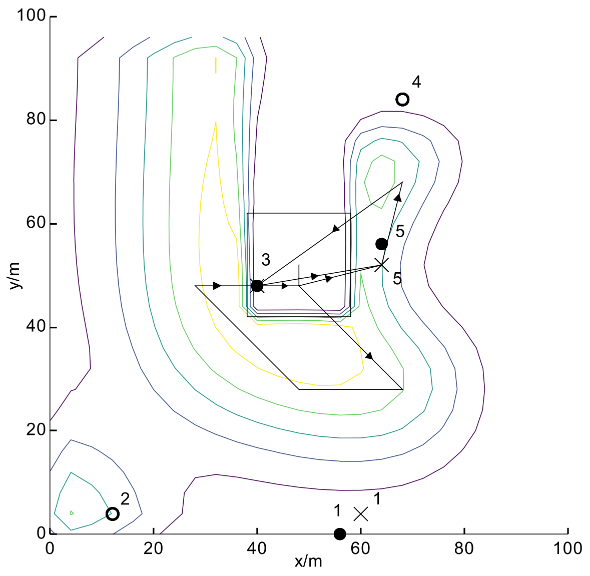

Fig. 10. Combined search and monitoring. In this case, targets 3 and 5 are known and nearby. There is also high probability distribution of unknown targets in the vicinity. The agent thus plans a route to intermittently traverse the unknown area while visiting targets 3 and 5 back and forth.

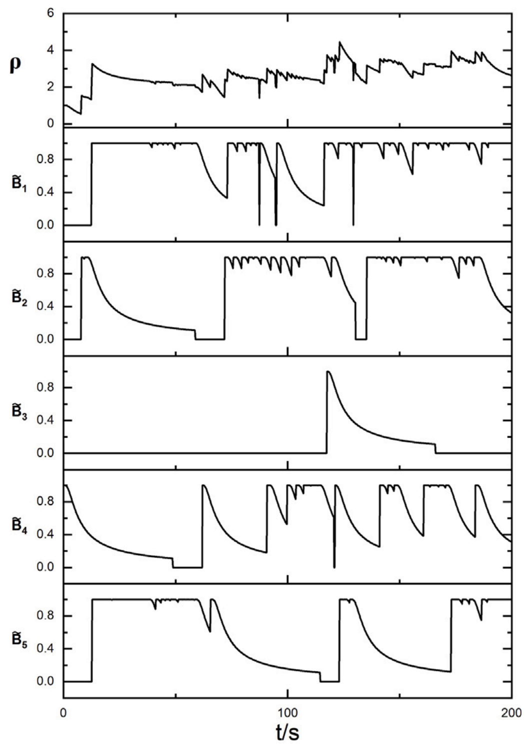

Fig. 12. *Belief probability* maintenance in simulation, with one robot. $p_s = 80\%$. $\tilde{B}_\lambda$ denotes the *belief probability* of target λ. ρ is the total *belief probability*, which is the reward of SSM at each time step. $\tilde{B}_\lambda = 0$ when target λ is unknown, and $\tilde{B}_\lambda = 1$ when it is currently being measured. This graph shows the development of the *belief probability* of each target during one simulation of 200 s.

together with hidden targets possibly in the neighbouring area, the default plan is to explore the surrounding unknown area and traverse the nearby known targets recurringly, thus merging search and monitoring in the same trajectory. Fig. 11 shows that when the monitoring effort is saturated, which happens when some known targets are within reach, but all other targets are far away, the agent would focus on traversing nearby known targets back and forth. In this case, a greedier effort to reach the farther area may lose what is currently being monitored.

Fig. 12 illustrates the *belief probability* of each target, $\tilde{B}_\lambda$, and the overall reward of the SSM mission $\rho = \sum_{\lambda\in\Lambda} \tilde{B}_\lambda$, at each time step of a case study. The *belief probability* of a target increases to 1 when it is detected and drops to 0 when it is lost. The *belief probability* degrades gradually when the target is not measured. In the graph, when a known target is lost, its *belief probability* can be lower than $\tilde{B}_l = 30\%$. This is because the potential sensor footprint for detecting a known target has been expanded, as mentioned in Section 4.8, making the estimated chance of detection higher than the actual *belief probability*. It can be seen that every target can be detected during the simulation. Most of them can be maintained at a high *belief probability* for several non-continuous periods and can be re-detected intermittently after being

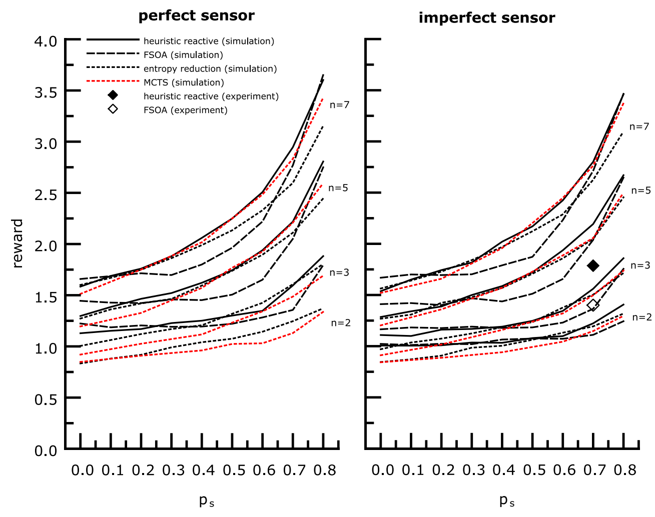

Fig. 13. Reactive policy vs. fixed sequence of actions vs. entropy reduction vs. Monte Carlo Tree Search. The lines in different style and colour show the average reward achieved by the heuristic reactive policy, the FSOA policy, the greedy policy of entropy reduction, and MCTS, through simulation in different scenarios. The solid and empty diamonds show the one-scenario experimental results of the heuristic reactive policy and the FSOA policy.

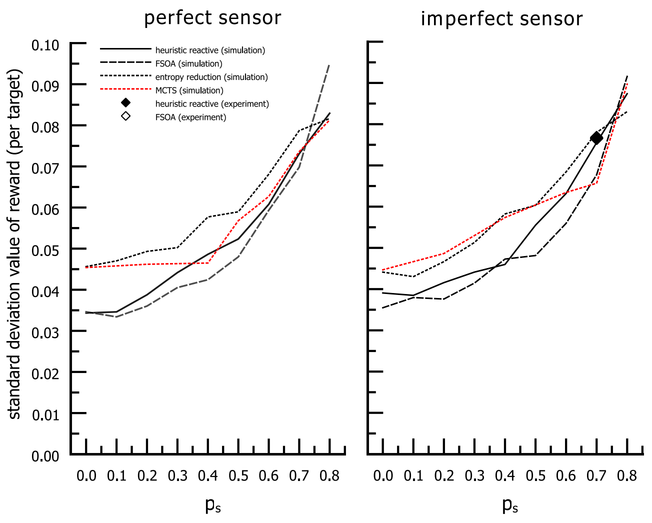

Fig. 14. Standard deviation value of reward (per target) in each scenario.

unattended. The negative spikes show that the targets may be lost when the agent tries to re-detect them, but they will soon be retrieved. The overall reward is increased shortly after the simulation starts and is kept above a certain level with fluctuations. It appears that the total reward is increasing over time after $t > 100$ s. However, after studying large quantities of simulations, we have found that there is no fixed pattern for the development of total reward, since it is highly dependency on the environment, which is dynamic and uncertain.

The case study qualitatively shows that, after search and monitoring are dynamically combined, the agent can efficiently search for hidden targets and preserve the *belief probability* of as many targets as possible. The search and monitoring efforts on each target are well scheduled.

7.1.2. Comparison with FSOA

We ran a quantitative study of the SSM and compared the performance of the proposed heuristic reactive policy planning with the FSOA policy. Scenarios with n = 2, 3, 5 and 7 targets were studied, and with $p_s = 0, 10, 20, 30, 40, 50, 60, 70, 80\%$. For each scenario, we did 100 cases of simulation for 200 s each, with $T_p = 1$ s. In each case, the initial locations of the agent and targets are randomly scattered. The reward of a scenario is the average reward of every time step in every case, and the average computation time of each planning is recorded as well. We also consider the cases with imperfect sensors, where at each time step, for the sensing of each target, there would be a 0.05% chance of a false positive or a 1% chance of a false negative. Fig. 13 shows the performances in each scenario. Each simulation is done by one core of the E5 2650V2 processor (2.6 GHz).

In most scenarios, the reward of heuristic reactive policy is better than that of the FSOA policy. It proves that if the future contingencies and corresponding reactions are considered during planning, the agent can make a better decision about future actions, which is consistent with Theorem 4.3. Given the end of Section 4.2, this comparison also proves that our synergistic SSM can achieve higher performance than conventional task assignment methods. When $p_s \leqslant 20\%$, the target motion is very uncertain, thus affecting assumption 2 and 6. Therefore in these scenarios, the heuristic reactive policy may not out-perform the policy of FSOA, which sets the boundary of our method.

Besides the average reward, in Fig. 14, we also show the standard deviation value of reward. In comparison with the differences of average reward between policies, the differences in the standard deviation values are much smaller. This suggests that all polices show similar consistency when solving this uncertain problem.

We studied the following case to explain the advantage of the heuristic reactive policy. In a situation where there are only two known targets, and $p_s = 80\%$, we planned the policy using both proposed heuristic reactive policy planning and the FSOA policy. The root base trajectories planned by both methods are shown in Fig. 15

In this case, the FSOA policy directs the robot to follow only one target, with an estimated value of 1.70. In contrast, the heuristic reactive policy planning drives the robot to go back and forth between two known targets, with a better value of 1.92. The FSOA policy does not choose the back-and-forth route because it assumes that the agent would still go back and forth if one target is lost, and the remaining target would always have a chance to escape between each visit. However, if the policy is reactive, the planner would know that the agent would focus on monitoring the remaining target if the other one is lost, which is a more reasonable strategy. Thus, while the FSOA policy tends to be conservative when there is a risk, the heuristic reactive policy allows the agent to make more sensible decisions.

Fig. 13 also shows that, in the case of an imperfect sensor, there will be a decrease in the performance of both approaches. However, this decreased performance can be mitigated by introducing sensor filtering to reduce the influence of false measurements.

According to the simulation, each planning of heuristic reactive policy takes 0.2 s on average. It is much slower than the FSOA policy, which takes 0.01 s on average. Nevertheless, the speed of heuristic reactive policy planning is still practical for real-time implementation.

7.1.3. Comparison with entropy reduction

Besides FSOA, we also compared our approach with the greedy entropy reduction method mentioned in Section 5. The result is also shown in Fig. 13. It can be seen that, although being almost instantaneous in planning speed, the performance of greedy entropy reduction is still suboptimal compared with the heuristic reactive policy. This proves that our POMDP formulation, although complicated, can achieve better performance in the SSM problem than the myopic entropy reduction approach.

7.1.4. Comparison with Monte Carlo tree search

In [49], one major challenge is to design a particle filter to predict the belief state throughout the progression of decision tree. In our problem, with the assumptions made in Section 4.3, the progression of belief state can be predicted straightforwardly with little computational cost.

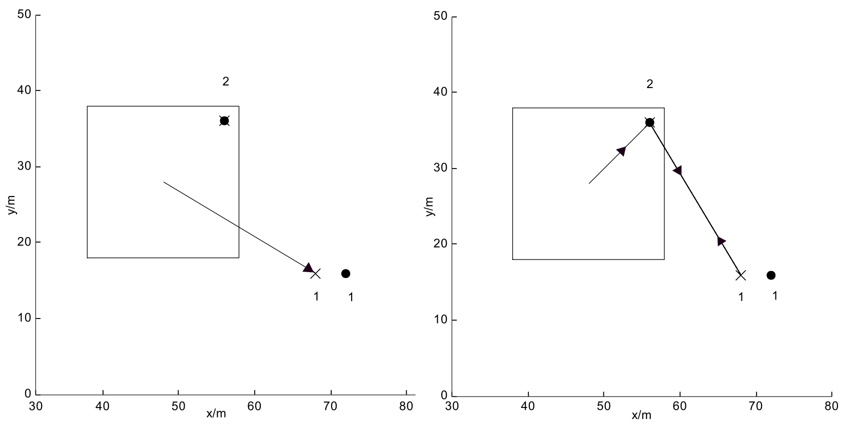

Fig. 15. A snapshot of root base trajectory planned in the same situation, by the FSOA policy (left) or heuristic reactive policy (right). There are only two targets in the situation, both known with the same location and belief.

Thus we applied the MCTS-based $\rho POMCP$ algorithm in [49] to SSM without using particle filters. The result is also shown in Fig. 13. After finding a good balance between exploration and exploitation, the average planning time for MCTS is 0.5 s (compared with 0.2 s for heuristic reactive policy planning), however the performance is still suboptimal to reactive heuristic policy in almost every scenario. Although state-of-the-art MCTS-based approaches are versatile and promising, this SSM is still too large to be solved online fast enough through conventional tree search. When there are 7 targets and the planning horizon is 10 s, even if we do 1-second macro-steps in rough planning and only consider detection/no detection in measurement, there can be at most 9 actions and $2^7 = 128$ observations at each step, with 10 steps maximum. Thus the worst case number of terminal leaf nodes in the search tree can be approximated as: $(9 \times 128)^{10} = 4.11 \times 10^{30}$, which appears to be a formidable size of a search tree.

The simulation comparison proves that, for the problem of SSM with a deep search tree and large branching factors, the heuristic reactive policy can explore the search tree more efficiently and develop a better policy.

7.2. Experiment with a single pursuer SSM

We also did an experimental study of the single-robot SSM, as illustrated in Fig. 1. We studied the scenario with n = 5 and p_s = 70%. We did 50 samples of experiments for both the heuristic reactive policy and the FSOA policy. The development of the *belief probabilities* in one experiment of heuristic reactive policy is shown in Fig. 16. It can be seen that the *belief probabilities* maintenance behaviour in the experiment is similar to that in the simulations. The average rewards of the two methods are included in Fig. 13. There is a decrease in performance compared with the simulations. This decrease is caused by the robot dynamics that incurs a delay in following the flight command. However, it still shows that the heuristic reactive policy has a dramatic advantage over FSOA in the tested scenario, which supports our major claims. In Fig. 14, the standard deviation values of rewards of two polices are almost identical, showing similar consistency against uncertainties.

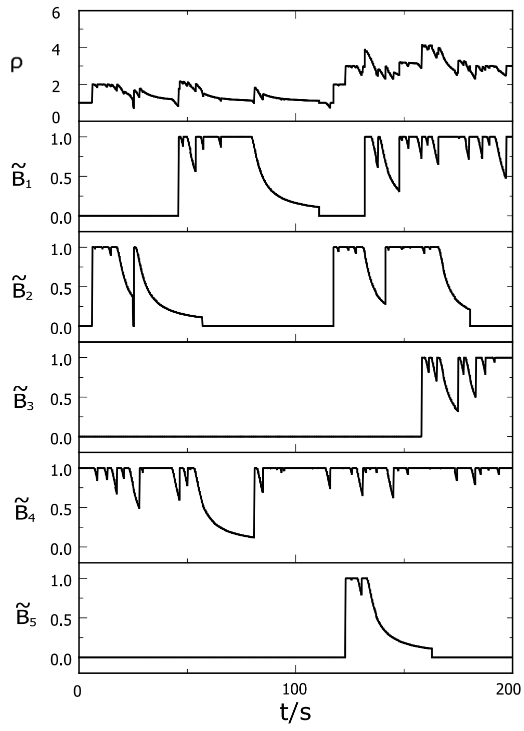

Fig. 16. *Belief probability* maintenance in experiment, with one robot. $p_s = 70\%$.

7.3. Simulation of multiple pursuer SSM

7.3.1. Case study

We then did a case study with the same setup of the environment and the same robot and target properties. Instead of having only one agent, three agents cooperate in the SSM, each planning and applying the local policy $\pi_\gamma \in \Pi$. The replanning is triggered at the same conditions as in the single robot scenario, and replanning happens for all robots simultaneously.

Figs. 17 and 18 are two snapshots of the case simulation.

In Fig. 17, only one target is known. The agent currently covering the target chooses to focus on monitoring and searches in the nearby area in the meantime. The other two agents divide the unknown areas and plan non-overlapping trajectories for search. In Fig. 18, when all targets have been detected, each agent goes back and forth to monitor nearby targets, and there is no redundant effort between the robots. Note that the collaboration of SSM is achieved in a distributed way, without any communication about plans. The case study shows that, with exchanging only sensing data, the agents can partition the tasks without overlap and do the SSM cooperatively.

Fig. 19 shows the development of the *belief probability* and the overall reward of the SSM mission at each time step of a case study. We can see that all targets can be detected soon after the beginning and can be maintained at a high *belief probability* for most of the times. Some sporadic negative spikes show that although sometimes the targets may be lost in monitoring, they will still be found right after. The overall

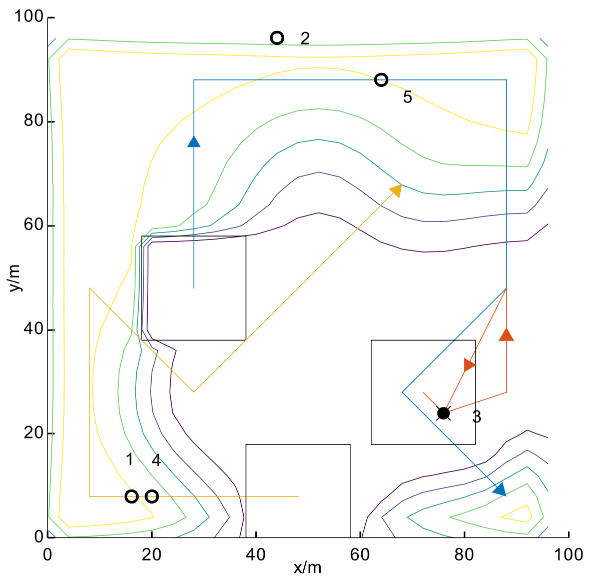

Fig. 17. Cooperative search. When three agents are available, and only one target has been found, two agents divide the unknown area to search, while the remaining agent plans to search the vicinity then return to check the known target.

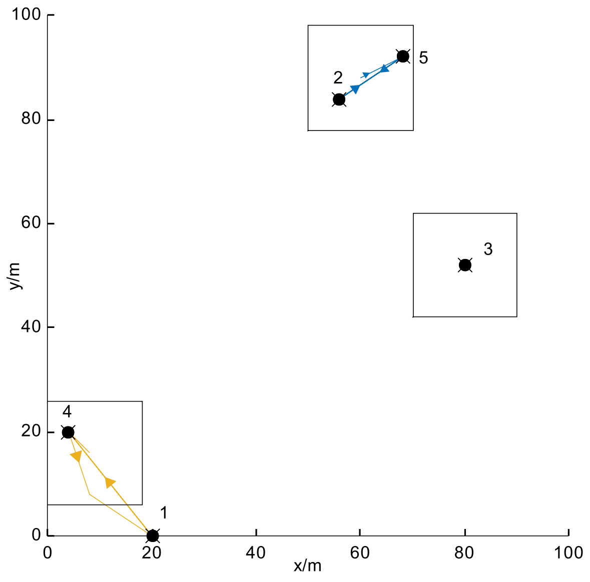

Fig. 18. Cooperative monitoring. When all targets have been detected, they are partitioned by three agents to be monitored.

award is increased quickly initially and is kept right under the highest level for most of the simulation. A comparison with Fig. 12 shows that having multiple agents brings substantial performance improvement.

7.3.2. Comparison with unlimited range of communication

We also ran a quantitative study, as in the single-agent case, to compare the performance of our proposed cooperative SSM with the non-cooperative SSM. In non-cooperative SSM, the measurements are received and belief states are maintained in the same way as cooperative SSM. However an agent plans its own policy while ignoring other agents, as in single-agent SSM. For the non-cooperative SSM and the following cooperative SSM with limited communication, we assume that there is a master agent that does nothing except for receiving full measurements from all other agents. The master agent updates its belief state based on the joint measurements, and the reward of SSM is based on this belief state.

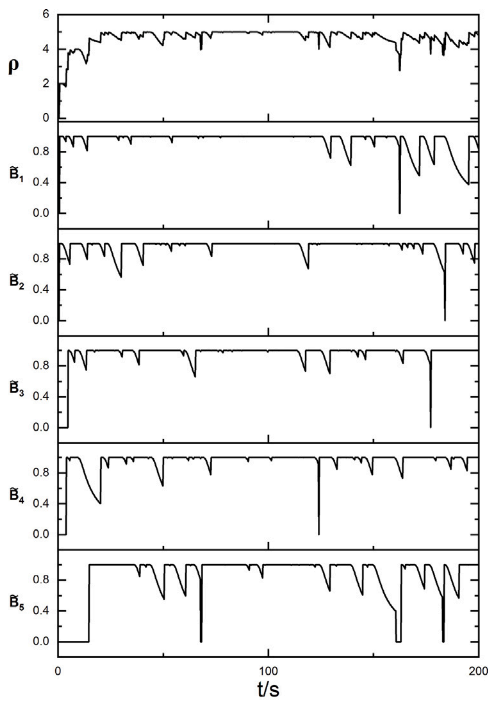

Fig. 19. *Belief probability* maintenance in simulation, with three robots. $p_s = 80\%$.

Table 1
Computation time for cooperative policy planning with unlimited communication.

Number of agents	Number of targets			
	2	3	5	7
2 agents	0.41 s	0.55 s	0.70 s	0.72 s
3 agents	–	0.86 s	1.21 s	1.30 s
5 agents	–	–	2.22 s	2.80 s

We maintained our assumption of an unlimited range of communication in this section. Scenarios with n = 2, 3, 5 and 7 targets were studied, with $p_s = 60, 70, 80\%$, and with m = 2, 3 and 5 robots. Each scenario was simulated for the same number of cases and length of time as in the single robot simulation. Fig. 20 shows the performances in each scenario.

Fig. 20 shows that in every scenario, there is a dramatic performance improvement after cooperation is considered. It proves that, in our distributed cooperative strategy planning, with only communication of measurement, each agent can independently plan its own strategy that considers the cooperation with other agents and can thus achieve a better overall reward compared with non-cooperative SSM. This validates Theorem 6.1.

The computation time for each scenario is shown in Table 1. It can be seen that, even though the planning is local, the computation time still grows with the number of agents. With more robots in the cooperation, more agents need to be planned for in the computation of the Nash equilibrium.

7.3.3. Comparison with limited range of communication

To make our method more scalable in bigger and more complex problems, we impose the range limit of communication, as mentioned in Section 6.5. We investigate the cases with communication ranges L_c = 50 m and 30 m. The performances are shown in Figs. 21 and 22.

Figs. 21 and 22 show that a reduced communication range does undermine the performance of cooperative SSM. However, the advantage of collaboration is still maintained. The computation time for scenarios with limited communication is in Tables 2 and 3. Comparing these tables with Table 1, it shows that the range limit largely reduces the

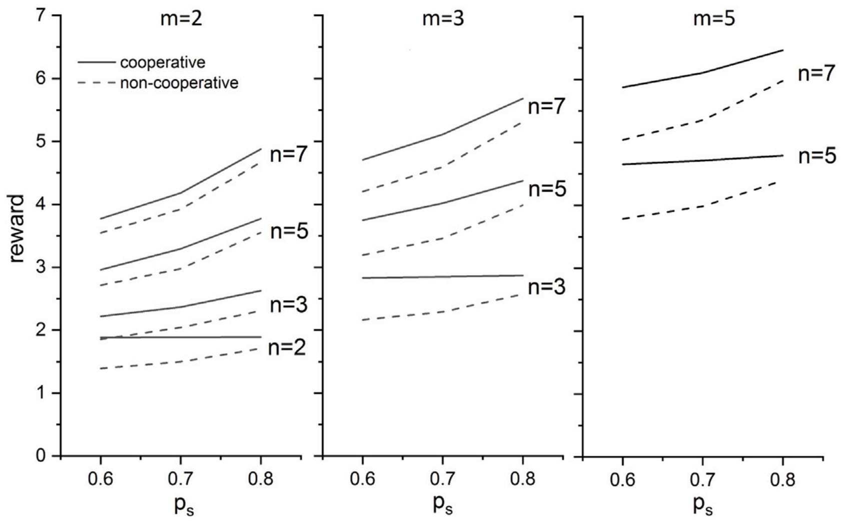

Fig. 20. Cooperative vs. non-cooperative. The solid and dashed lines are the reward achieved through simulation in different scenarios, by cooperative SSM and non-cooperative SSM.

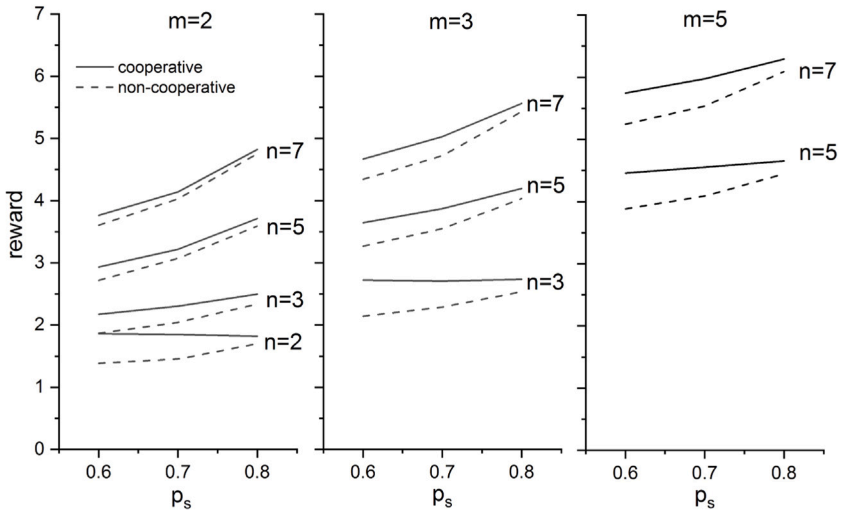

Fig. 21. Cooperative vs. non-cooperative when $L_c = 50$ m.

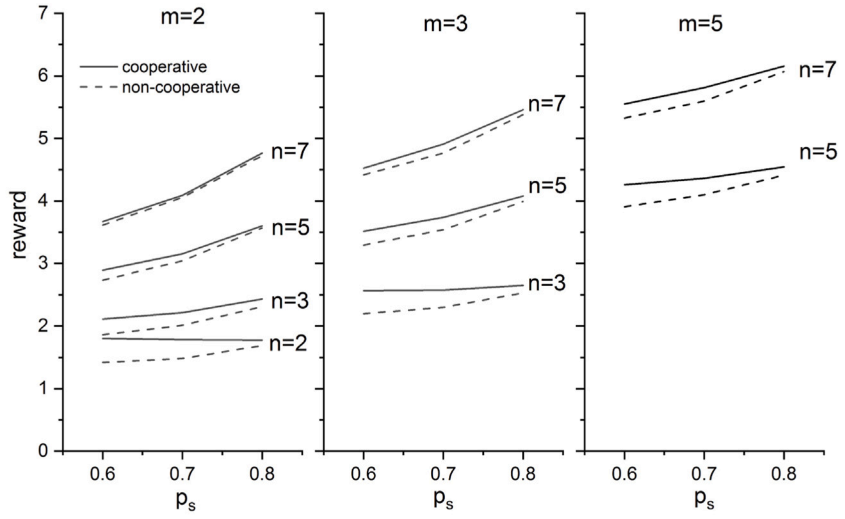

Fig. 22. Cooperative vs. non-cooperative when $L_c = 30$ m.

Table 2
Computation time for cooperative policy planning when $L_c = 50$ m.

Number of agents	Number of targets			
	2	3	5	7
2 agents	0.27 s	0.44 s	0.43 s	0.55 s
3 agents	–	0.48 s	0.62 s	0.77 s
5 agents	–	–	1.11 s	1.50 s

Table 3
Computation time for cooperative policy planning when $L_c = 30$ m.

Number of agents	Number of targets			
	2	3	5	7
2 agents	0.23 s	0.30 s	0.38 s	0.38 s
3 agents	–	0.46 s	0.55 s	0.61 s
5 agents	–	–	0.87 s	1.00 s

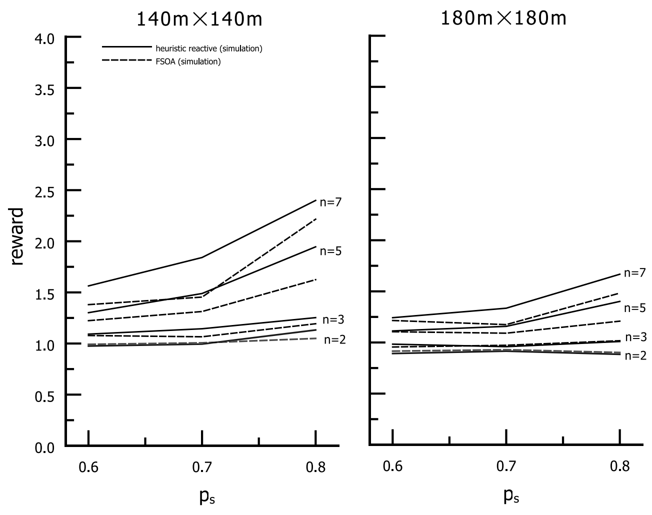

Fig. 23. Reactive policy vs. fixed sequence of actions, with a single agent. The sizes of arenas are 140 m × 140 m and 180 m × 180 m.

Table 4
Computation time for heuristic reactive policy planning, with a single agent.

	140 m × 140 m	180 m × 180 m
Computational time per planning	0.51 s	0.42 s

computation cost. Although the planning efficiency still degrades with the number of agents, the communication range imposed should help with the scalability of semi-Dec-POMDP, as long as the agents are not densely deployed.

7.4. Study of scalability

To further study the scalability of our approaches, we look into SSM with bigger size of environment. Simulations with arenas in a size of 140 m×140 m and 180 m×180 m are conducted. Fig. 23 shows the study on single-agent SSM. In Figs. 24 and 25, multi-agent SSM with 30 m of communication range are also studied. Tables 4 to 6 are the average computational time for the above scenarios.

From Figs. 23 to 25, it shows that the reward in every scenario decreases with the size of environment. This is very natural. In a bigger arena, the targets are more sparsely scattered, making it more difficult for the agent to acquire and maintain target information, thus the drop of reward is inevitable.

Comparing Table 4 and the 0.2 s average computational time in the original arena. The time cost more than doubled when the size of arena = 140 m×140 m, then dropped slightly when the size of arena = 180 m× 180 m. When the size expands, with the same planning horizon, time

Table 5
Computation time for cooperative policy planning when $L_c = 30$ m. The size of arena is 140 m × 140 m.

Number of agents	Number of targets			
	2	3	5	7
2 agents	0.32 s	0.35 s	0.42 s	0.45 s
3 agents	–	0.52 s	0.59 s	0.64 s
5 agents	–	–	0.99 s	1.03 s

Table 6
Computation time for cooperative policy planning when $L_c = 30$ m. The size of arena is 180 m × 180 m.

Number of agents	Number of targets			
	2	3	5	7
2 agents	0.29 s	0.31 s	0.34 s	0.38 s
3 agents	–	0.46 s	0.50 s	0.54 s
5 agents	–	–	0.82 s	0.85 s

step, and number of targets, the increase of possible player motions makes the planning more complex. However the number of targets that an agent can cover decreases, making the planning easier. This counteraction between two factors may explain why the time cost first increased then dropped.

Comparing Tables 3, 5 and 6, the increase of time cost is much smaller when the size of arena = 140 m × 140 m, compared with that of single-agent SSM. The time cost also dropped when the size of arena = 180 m × 180 m. Nonetheless, we can still observe the time cost increasing with the number of agents or targets.

According to the above results, the computational cost may increase with the size of environment or the number of targets and agents. Fortunately, the planning time reduces after a certain size of arena, and a smaller communication range can also reduce time cost. Thus one possible direction of future study is trying to find a theoretical upper bound of planning time, given a range limit of communication and a maximum density of players, therefore guaranteeing the conditional scalability of our approach.

8. Conclusion

A novel and essential simultaneous search and monitoring problem is proposed and studied in this work. Compared with methods of task assignment that are conventional and intuitive, or entropy reduction, which is simpler, we combine the search and monitoring as a united mission in a synergistic solution and strive for look-ahead planning. Scenarios with both single and multiple agents are addressed. The complex interconnection between search and monitoring makes the combined problem a $\rho POMDP$ or Dec-POMDP for single or multiple agents.

A united value function was first proposed to solve the $\rho POMDP$ for a single robot. It sets synergistic search and monitoring instead of a simplistic trade-off. The concept of policy improvement is then applied to allow for real-time online planning.

An original policy reconstruction method is designed to build a structure to approximate the optimal policy. We have proved that any optimal policy can be fully reconstructed in such a way. Compared with conventional tree search approaches, it not only makes it easy to formulate a good candidate policy but also makes it more efficient to do *playout* and policy improvement in each iteration. As part of policy reconstruction, we then designed a heuristic reactive branching function. We have proved that such a heuristic reactive policy is better than the conventional FSOA policy. With the policy reconstruction, the policy planning is transformed into finding the best root base trajectory, which is solved by the simulated annealing method.

We then extend the heuristic reactive policy planning to scenarios with multiple robots. The planning for a single agent is combined with the concept of a Nash equilibrium. A cooperative SSM strategy can thus

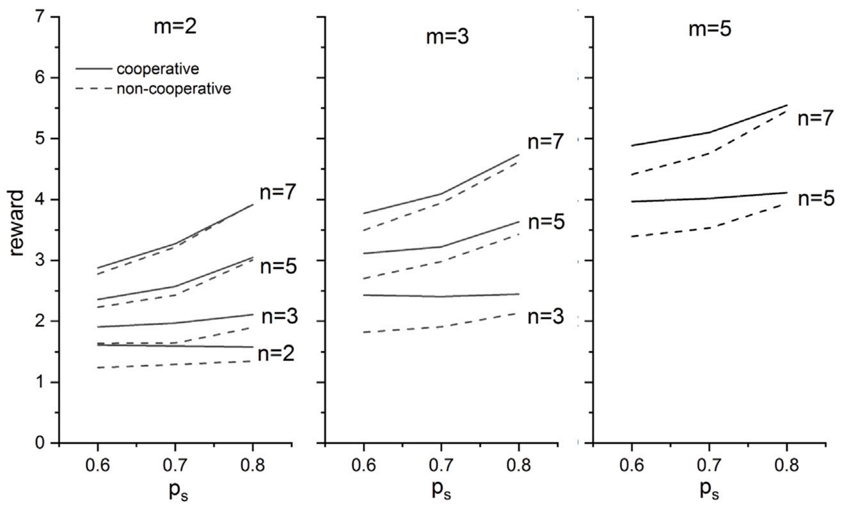

Fig. 24. Cooperative vs. non-cooperative when $L_c = 30$ m. The size of arena is $140\ \text{m} \times 140\ \text{m}$.

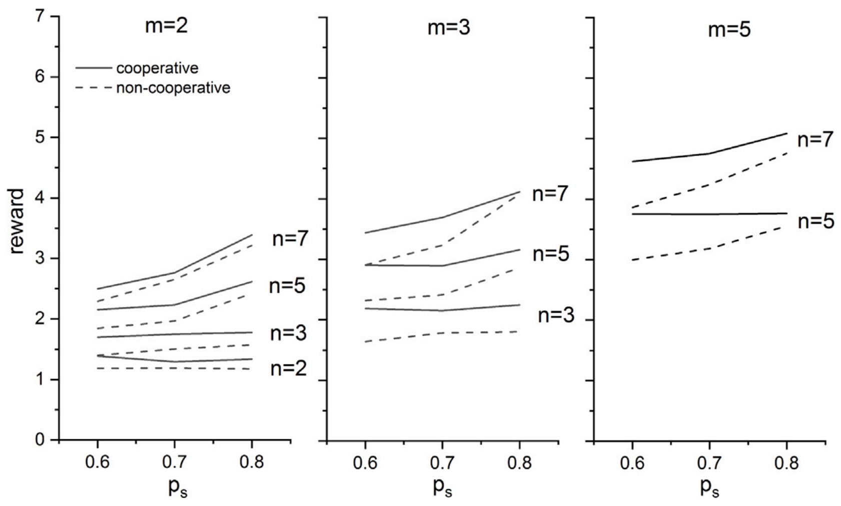

Fig. 25. Cooperative vs. non-cooperative when $L_c = 30$ m. The size of arena is $180\ \text{m} \times 180\ \text{m}$.

be planned and executed locally by each agent. We also proved that the cooperative SSM has better performance than the non-cooperative SSM.

The simulations and experiments show that our proposed heuristic reactive policy can effectively search for hidden targets in an initially unknown environment and maintain their surveillance with moderate computational cost. Whenever the monitoring capability is not saturated, the agent will try to find more targets without losing current known ones. In the multi-agent case, the robots can divide the tasks with no overlap, thus cooperating without redundant effort. In the comparative studies, we validated that the proposed heuristic reactive policy works better than the conventional method of FSOA and that the cooperation between agents is advantageously achieved. The superiority of the heuristic reactive policy over FSOA also proves that the concept of synergistic search and monitoring is more suitable for SSM than a task assignment approach. When compared with the strategy of entropy reduction, our approach showed higher performance in most scenarios. This further proves that, in this problem, look-ahead stochastic planning has better potential than a simple entropy reduction method. The comparative study with the MCTS-based state-of-the-art well demonstrates the contribution of this work, proving that for a problem like SSM, the presented concept of policy reconstruction can help build a better reactive policy more efficiently than basic MCTS.

The density of players and range of communication can both help to improve scalability of our approach. However, finding an upper bound of computational cost is still open for future study.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

Appendix A. Proof of Theorem 4.2

Proof. Assume that there is an arbitrary deterministic policy $\pi(b_t)$. Applying such a policy from the initial belief state b_{t_0} until the terminal time t_f, if no branching event happened, which is when $\{b_t | t = t_0 + 1, \dots, t_f\} \in \Delta^{\bullet}$, let $\{a_t^{\bullet} | t = t_0, \dots, t_f, a_t^{\bullet} = \pi(b_t)\}$ denote the corresponding sequence of actions in such a case. Given known policy $\pi(b_t)$, the

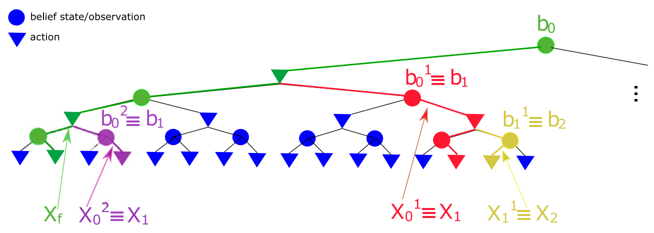

Fig. 26. Example of branching tree. b^x_{k-1} or b_k denotes the kth branching belief state in the history along the branching tree. χ^x_{k-1} or χ_k denotes the respective base trajectory after the kth branching.

sequences $\{b_t|t = t_0 + 1, \dots, t_f\} \in \Delta^{\bullet}$ and $\{a^{\bullet}_t|t = t_0, \dots, t_f, a^{\bullet}_t = \pi(b_t)\}$ can be determined. Let $\chi = \{a^{\bullet}_{t_0}, \dots, a^{\bullet}_{t_f}\}$.

If, during the execution of policy $\pi(b_t)$, a branching event happened at time t_1, then the immediate action taken is $a^{\circ}_{t_1} = \pi(b_{t_1})$. Let $\{a''_t|t = t_1 + 1, \dots, t_f, a''_t = \pi(b_t)\}$ denote the corresponding sequence of actions for a later sequence of non-branching states. Let $\chi^{\circ} = \{a^{\circ}_{t_1}, a''_{t_1+1}, \dots, a''_{t_f}\}$.

It can be seen that after iteratively applying this process, all the possible states and the corresponding actions in the policy tree can be reconstructed by the combination of χ and all χ°s, which means that $\pi(b_t)$ can be fully reconstructed by $\hat{\pi}(b_t, \chi)$ defined in Algorithm 1. According to Lemma 4.1, there exists an optimal policy $\pi^*(b_t)$; thus, it can be reconstructed by $\hat{\pi}^*(b_t, \chi)$, which is also optimal. □

Appendix B. Proof of Theorem 4.3

Proof. Assume that there is an optimal policy of FSOA $\hat{\pi}^*_f(b_t, \chi_f)$. We build a heuristic reactive policy $\hat{\pi}_a(b_t, \chi_f)$, which takes χ_f as the initial base trajectory.

For an agent applying policy $\hat{\pi}_a$, branchings may be triggered recursively. We let b_k be the kth branching belief state in the history of one *playout* and let χ_k denote the respective base trajectory after the kth branching. t_k is the timing for b_k. Without loss of generality, the backtrace history before χ_k, b_k and t_k is omitted from the notation. Assume that there can only be at most $M \le T_h/dt$ layers of branchings within the time horizon. Thus, $0 \leqslant k \leqslant M$. Along χ_k, assume that N_k possible branching events can happen. The structure of the branching tree is illustrated in Fig. 26.

Let t^x_k and p^x_k denote the time instant and probability of the xth possible branching event to occur along χ_k, and b^x_k is the corresponding branching belief state, where $0 \leqslant x \leqslant N_k$. p^0_k denotes the probability that no branching event happens along χ_k. Let $V_h(t_a, t_b|\chi_k)$ denote the hindsight value from time t_a to t_b, given no branching event happens along χ_k. Let $V^f_{\chi_f}(b_{t_0}) = V_{\hat{\pi}^*_f(b,\chi_f)}(b_{t_0})$ be the value of applying policy $\hat{\pi}^*_f$ from the initial belief state b_{t_0} until the finishing time t_f, and let $V^a_{\chi_f}(b_{t_0}) = V_{\hat{\pi}_a(b,\chi_f)}(b_{t_0})$ be the value of applying heuristic reactive policy $\hat{\pi}_a$. Then $V^a_{\chi_f}(b_{t_0})$ can be constructed recursively in the following way.

$$V^a_{\chi_f}(b_0) = p^0_0 V_h(t_0, t_f|\chi_f) + p^1_0(\delta^1_0 V_h(t_0, t^1_0|\chi_f)$$
$$+ (1-\delta^1_0)V^a_{\chi^1_0}(b^1_0)) + \dots + p^{N_0}_0(\delta^{N_0}_0 V_h(t_0, t^{N_0}_0|\chi_f)$$
$$+ (1-\delta^{N_0}_0)V^a_{\chi^{N_0}_0}(b^{N_0}_0));$$

...

$$V^a_{\chi_1}(b_1) = p^0_1 V_h(t_1, t_f|\chi_1) + p^1_1(\delta^1_1 V_h(t_1, t^1_1|\chi_1)$$
$$+ (1-\delta^1_1)V^a_{\chi^1_1}(b^1_1)) + \dots + p^{N_1}_1(\delta^{N_1}_1 V_h(t_1, t^{N_1}_1|\chi_1)$$
$$+ (1-\delta^{N_1}_1)V^a_{\chi^{N_1}_1}(b^{N_1}_1));$$

...

$$V^a_{\chi_k}(b_k) = p^0_k V_h(t_k, t_f|\chi_k) + p^1_k(\delta^1_k$$
$$V_h(t_k, t^1_k|\chi_k) + (1-\delta^1_k)V^a_{\chi^1_k}(b^1_k)) + \dots + p^{N_k}_k$$
$$(\delta^{N_k}_k V_h(t_k, t^{N_k}_k|\chi_k) + (1-\delta^{N_k}_k)V^a_{\chi^{N_k}_k}(b^{N_k}_k));$$

$\dots k \in [1, M-1]$

where $\delta^x_k = (t^x_k - t_k)/(t_f - t_k)$. $\chi^x_k = f_a(b^x_k, \chi_k)$, which is the new base trajectory after the xth branching along χ_k. By abuse of notation, we let $b_{k+1} = b^x_k$ and $\chi_{k+1} = \chi^x_k$, which also omits the backtrace history.

Because there will be no more branching after b_M, then $V^a_{\chi_M}(b_M) = V^a_{\chi^x_{M-1}}(b^x_{M-1}) = V^f_{\chi^x_{M-1}}(b^x_{M-1}) = V^f_{\chi_M}(b_M)$. Based on the definition of $f_a(b,\chi)$, $V^a_{\chi_M}(b_M) = V^f_{\chi^x_{M-1}}(b^x_{M-1}) \ge V^f_{\chi_{M-1}}(b^x_{M-1})$. Thus,

$$V^a_{\chi_{M-1}}(b_{M-1}) = p^0_{M-1}V_h(t_{M-1}, t_f|\chi_{M-1}) + p^1_{M-1}(\delta^1_{M-1}$$
$$V_h(t_{M-1}, t^1_{M-1}|\chi_{M-1}) + (1-\delta^1_{M-1})V^a_{\chi^1_{M-1}}(b^1_{M-1}))+$$
$$\dots + p^{N_{M-1}}_{M-1}(\delta^{N_{M-1}}_{M-1}V_h(t_{M-1}, t^{N_{M-1}}_{M-1}|\chi_{M-1})+$$
$$(1-\delta^{N_{M-1}}_{M-1})V^a_{\chi^{N_{M-1}}_{M-1}}(b^{N_{M-1}}_{M-1}))$$
$$\ge p^0_{M-1}V_h(t_{M-1}, t_f|\chi_{M-1}) + p^1_{M-1}(\delta^1_{M-1}V_h(t_{M-1}, t^1_{M-1}|\chi_{M-1})$$
$$+ (1-\delta^1_{M-1})V^f_{\chi_{M-1}}(b^1_{M-1})) + \dots +$$
$$p^{N_{M-1}}_{M-1}(\delta^{N_{M-1}}_{M-1}V_h(t_{M-1}, t^{N_{M-1}}_{M-1}|\chi_{M-1})+$$
$$(1-\delta^{N_{M-1}}_{M-1})V^f_{\chi_{M-1}}(b^{N_{M-1}}_{M-1})) = V^f_{\chi_{M-1}}(b_{M-1})$$
$$= V^f_{\chi^x_{M-2}}(b^x_{M-2}) \ge V^f_{\chi_{M-2}}(b^x_{M-2})$$

Applying the same process iteratively, it can be seen that

$$V^a_{\chi_k}(b_k) \ge V^f_{\chi_k}(b_k) \ge V^f_{\chi_{k-1}}(b^x_{k-1})$$

...

$$V^a_{\chi_f}(b_{t_0}) \ge V^f_{\chi_f}(b_{t_0})$$

Thus, given an optimal FSOA policy $\hat{\pi}^*_f(b_t, \chi_f)$, there will always be a heuristic reactive policy $\hat{\pi}_a(b_t, \chi_f)$ to achieve a higher or at least equal value, which proves the theorem. □

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2023.104544.

References

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

[2] Lawrence D. Stone, Johannes O. Royset, Alan R. Washburn, Optimal search for moving targets.

[3] Zhijun Tang, Umit Ozguner, Motion planning for multitarget surveillance with mobile sensor agents, IEEE Trans. Robot. 21 (5) (2005) 898–908.

[4] Scott A. Miller, Zachary A. Harris, Edwin K.P. Chong, A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking, EURASIP J. Adv. Signal Process. 2009 (1) (2009) 1–17.

[5] Mac Schwager, Daniela Rus, Jean-Jacques Slotine, Decentralized, adaptive coverage control for networked robots, Int. J. Robot. Res. 28 (3) (2009) 357–375.

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

[7] Junfei Xie, Luis Rodolfo Garcia Carrillo, Lei Jin, Path planning for UAV to cover multiple separated convex polygonal regions, IEEE Access 8 (2020) 51770–51785.

[8] Rene Vidal, Omid Shakernia, H Jin Kim, David Hyunchul Shim, Shankar Sastry, Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation, IEEE Trans. Robot. Autom. 18 (5) (2002) 662–669.

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

[10] Andreas Kolling, Stefano Carpin, Cooperative observation of multiple moving targets: an algorithm and its formalization, Int. J. Robot. Res. 26 (9) (2007) 935–953.

[11] Sara Bernardini, Maria Fox, Derek Long, Combining temporal planning with probabilistic reasoning for autonomous surveillance missions, Auton. Robots 41 (2017) 181–203.

[12] Yan Jin, Yan Liao, Ali A. Minai, Marios M. Polycarpou, Balancing search and target response in cooperative unmanned aerial vehicle (UAV) teams, IEEE Trans. Syst. Man Cybern. B 36 (3) (2005) 571–587.

[13] Eric W. Frew, Jack Elston, Target assignment for integrated search and tracking by active robot networks, in: 2008 IEEE International Conference on Robotics and Automation, IEEE, 2008, pp. 2354–2359.
[14] Jack Elston, Eric W. Frew, Hierarchical distributed control for search and tracking by heterogeneous aerial robot networks, in: 2008 IEEE International Conference on Robotics and Automation, IEEE, 2008, pp. 170–175.
[15] Thomas M. Cover, Joy A. Thomas, Elements of Information Theory, John Wiley & Sons, 2012.
[16] Gabriel Hoffmann, Steven Waslander, Claire Tomlin, Distributed cooperative search using information-theoretic costs for particle filters, with quadrotor applications, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2006, p. 6576.
[17] Louis K. Dressel, Mykel J. Kochenderfer, Efficient and low-cost localization of radio signals with a multirotor UAV, in: 2018 AIAA Guidance, Navigation, and Control Conference, 2018, p. 1845.
[18] Ben Grocholsky, James Keller, Vijay Kumar, George Pappas, Cooperative air and ground surveillance, Robot. Autom. Mag. IEEE 13 (3) (2006) 16–25.
[19] Cindy Leung, Shoudong Huang, Ngai Kwok, Gamini Dissanayake, Planning under uncertainty using model predictive control for information gathering, Robot. Auton. Syst. 54 (11) (2006) 898–910.
[20] Louis Dressel, Mykel J. Kochenderfer, Efficient decision-theoretic target localization, in: Twenty-Seventh International Conference on Automated Planning and Scheduling, 2017.
[21] James N. Eagle, The optimal search for a moving target when the search path is constrained, Oper. Res. 32 (5) (1984) 1107–1115.
[22] Hiroyuki Sato, Johannes O. Royset, Path optimization for the resource-constrained searcher, Nav. Res. Logist. 57 (5) (2010) 422–440.
[23] Dimitris J. Bertsimas, Garrett Van Ryzin, A stochastic and dynamic vehicle routing problem in the Euclidean plane, Oper. Res. 39 (4) (1991) 601–615.
[24] Cheng Song, Lu Liu, Gang Feng, Yong Wang, Qing Gao, Persistent awareness coverage control for mobile sensor networks, Automatica 49 (6) (2013) 1867–1873.
[25] Pablo Lanillos, Eva Besada-Portas, Gonzalo Pajares, José J Ruz, Minimum time search for lost targets using cross entropy optimization, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 602–609.
[26] Sara Bernardini, Maria Fox, Derek Long, Chiara Piacentini, Deterministic versus probabilistic methods for searching for an evasive target, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[27] Rosemary Emery-Montemerlo, Geoff Gordon, Jeff Schneider, Sebastian Thrun, Approximate solutions for partially observable stochastic games with common payoffs, in: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1, IEEE Computer Society, 2004, pp. 136–143.
[28] Iadine Chades, Bruno Scherrer, François Charpillet, A heuristic approach for solving decentralized-pomdp: Assessment on the pursuit problem, in: Proceedings of the 2002 ACM Symposium on Applied Computing, ACM, 2002, pp. 57–62.
[29] Christos H. Papadimitriou, John N. Tsitsiklis, The complexity of Markov decision processes, Math. Oper. Res. 12 (3) (1987) 441–450.
[30] Daniel S Bernstein, Robert Givan, Neil Immerman, Shlomo Zilberstein, The complexity of decentralized control of Markov decision processes, Math. Oper. Res. 27 (4) (2002) 819–840.
[31] Richard D. Smallwood, Edward J. Sondik, The optimal control of partially observable Markov processes over a finite horizon, Oper. Res. 21 (5) (1973) 1071–1088.
[32] Eric A. Hansen, Daniel S. Bernstein, Shlomo Zilberstein, Dynamic programming for partially observable stochastic games, in: AAAI, Vol. 4, 2004, pp. 709–715.
[33] E. Walraven, Mtj Spaan, Point-based value iteration for finite-horizon POMDPs, J. Artificial Intelligence Res. (2019).
[34] Mauricio Araya, Olivier Buffet, Vincent Thomas, Françcois Charpillet, A POMDP extension with belief-dependent rewards, Adv. Neural Inf. Process. Syst. 23 (2010).
[35] Yash Satsangi, Shimon Whiteson, Frans A Oliehoek, Matthijs TJ Spaan, Exploiting submodular value functions for scaling up active perception, Auton. Robots 42 (2) (2018) 209–233.
[36] Hyeong Soo Chang, Robert Givan, Edwin K.P. Chong, Parallel rollout for online solution of partially observable Markov decision processes, Discrete Event Dyn. Syst. 14 (3) (2004) 309–341.
[37] M. Ahmadi, U. Rosolia, MD Ingham, R.M. Murray, A.D. Ames, Risk-averse decision making under uncertainty, 2021.
[38] Stéphane Ross, Joëlle Pineau, Sébastien Paquet, Brahim Chaib-draa, Online planning algorithms for POMDPs, J. Artificial Intelligence Res. 32 (2008) 663–704.
[39] Haoyu Zhang, Sandor Veres, Andreas Kolling, Simultaneous search and monitoring by unmanned aerial vehicles, in: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), IEEE, 2017, pp. 903–910.
[40] Kamil Dedecius, Diffusion estimation of state-space models: Bayesian formulation, in: Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on, IEEE, 2014, pp. 1–6.
[41] Zhijun Tang, 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, Vol. 1, IEEE, 2003, pp. 19–24.
[42] Martin L. Puterman, Markov decision processes. Discrete stochastic dynamic programming MVspa, 2005.
[43] Matthijs T.J. Spaan, Tiago S. Veiga, Pedro U. Lima, Decision-theoretic planning under uncertainty with information rewards for active cooperative perception, Auton. Agents Multi-Agent Syst. 29 (6) (2014) 1157–1185.
[44] Eric A. Hansen, Solving POMDPs by searching in policy space, in: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1998, pp. 211–219.
[45] Pascal Poupart, Craig Boutilier, VDCBPI: an approximate scalable algorithm for large POMDPs, in: Advances in Neural Information Processing Systems, 2005, pp. 1081–1088.
[46] E.K.P. Chong, C. Kreucher, A.O. Hero III, POMDP approximation methods based on heuristics and simulation, Found. Appl. Sensor Manage. 8 (2007) 95–120.
[47] W. Ding, L. Zhang, J. Chen, S. Shen, EPSILON: An efficient planning system for automated vehicles in highly interactive environments, IEEE Trans. Robot.: Publ. IEEE Robot. Autom. Soc. (2) (2022) 38.
[48] Dimitri P. Bertsekas, Dynamic Programming and Optimal Control, Vol. 1, Athena Scientific Belmont, MA, 1995.
[49] Vincent Thomas, Gérémy Hutin, Olivier Buffet, Monte Carlo information-oriented planning, 2021, arXiv preprint arXiv:2103.11345.
[50] Songhwai Oh, Shankar Sastry, 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, IEEE, 2005, pp. 2197–2202.
[51] P.J. van Laarhoven, E.H. Aarts, Simulated Annealing: Theory and Applications, Vol. 37, Springer Science & Business Media, 2013.
[52] Yang You, Vincent Thomas, Francis Colas, Olivier Buffet, Solving infinite-horizon dec-POMDPs using finite state controllers within JESP, in: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2021, pp. 427–434.
[53] Ebtehal Turki Alotaibi, Shahad Saleh Alqefari, Anis Koubaa, LSAR: Multi-UAV collaboration for search and rescue missions, IEEE Access 7 (2019) 55817–55832, http://dx.doi.org/10.1109/ACCESS.2019.2912306.
[54] Graeme Best, Oliver M Cliff, Timothy Patten, Ramgopal R Mettu, Robert Fitch, Dec-MCTS: Decentralized planning for multi-robot active perception, Int. J. Robot. Res. 38 (2–3) (2019) 316–337.
[55] Laëtitia Matignon, Laurent Jeanpierre, Abdel-Illah Mouaddib, Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 26, 2012, pp. 2017–2023.

Haoyu Zhang is a former Ph.D. Student at the Department of Automatic Control and Systems Engineering at the University of Sheffield. His research interests include robot search and pursuit evasion, planning and decision making, and flight dynamics and control. After receiving his Ph.D. degree, he is currently working in the field of autonomous driving of commercial road vehicles.

Sandor M. Veres had been Professor of Autonomous Control Systems at the Department of Automatic Control and Systems Engineering at the University of Sheffield, UK, where he had been the director of the Autonomous Systems and Robotics Group until his retirement in 2022. In the past four decades he published in a variety of research areas such as information theory, system identification, adaptive control, active sound and vibration control, satellite formation flying, verification of decision systems on underwater vehicles and drones, methods of reconfigurable autonomy, decision making among distributed agents, verifiable software architectures for autonomous robots and recently on human robot interactions with building of trust, overall in about 300 refereed papers and 6 books.

Andreas Kolling is a principal applied scientist at Amazon Robotics working on autonomous mobility. Previously, Andreas Kolling worked as a senior principal scientist for iRobot, developing technologies for planning, navigation, and mapping for the Roomba i7+, which received numerous awards as the most advanced consumer robot. Prior to this, he was an assistant professor at the University of Sheffield, England, and a postdoctoral research fellow at the Robotics Institute at Carnegie Mellon University. His research interests include planning, mapping, AI, machine learning, multi-robot systems, human–robot interaction and robot software. He has published more than sixty peer-reviewed articles, served as general co-chair for DARS 2016, and as associate editor for ICRA and IROS since 2014.