Multi-Agent Cooperative Target Search Based on Reinforcement Learning

Xudong Qin*, Xiaomao Li, Yuan Liu, Rui Zhou, Jiajia Xie

School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200444, China.

*Corresponding author e-mail: qinxdmall@foxmail.com

Abstract. This paper considers the problem of cooperative search for multiple stationary targets by multi-agent with limited sensing and communication capabilities. An integrated learning algorithm for cooperative search based on reinforcement learning is proposed. In particular, we consider a state containing local probability map and neighbouring agents map which provides the agent with information to plan routes and search collaboratively. In addition, a reward function consisting of target found reward, time consumption reward and guiding reward is designed to guide agents to explore and learn efficiently. To ensure the stability of training process, the policies of agents are frozen and shared periodically in a distributed training framework. The proposed method is tested under simulated scenarios compared with coverage control methods and random strategies. Multiple simulation results show considerable advantages.

1. Introduction

With the development of robotic technology, an increasing number of tasks are completed by single or multiple intelligent and autonomous robots. Compared with single-agent systems, multi-agent systems can accomplish more complex tasks such as search and rescue [1], harbour protection [2], but at the same time, it faces more technical problems. Multi-agent systems require communication and collaboration between agents and more complex control strategies.

We consider the problem of target search by multi-agent which has been studied by relevant scholars for some time. It is a traditional target search method to use the probability map to represent the surveillance region and update it by Bayesian rule after it is partitioned into cells [3, 4]. In [5], a distributed search algorithm is proposed which includes both map update and fusion procedure. Based on this algorithm, authors of [6] designed a consensus-like distributed scheme for cooperative target search including transformed map update, local map fusion and coverage control path planning. However, the search method based on coverage control has its drawbacks, including not applicable for fewer agents, easy to fall into local stability and not adapt to complex scenarios. Some recent studies have also explored some new methods for multi-agent cooperative search problems [7, 8].

In recent years, development of the reinforcement learning (RL) has provided an alternative way to deal with target search problems and RL is essentially suitable for this task for its learning from the interaction with the environment. Related research includes that [9] combined the RL algorithm with the Voronoi-based coverage control method and applies it to the area coverage control problems, but the RL algorithm is only used to approximate the control law. [10] developed a flocking control
framework based on DDPG algorithm which used the centralized training and distributed execution framework, it differs from this paper in both the task and the training framework. In this paper, we proposed an integrated learning algorithm for cooperative search based on reinforcement learning. State and observation representations transformed from probability map and reward function are designed to guide agents explore and learn. A relatively stable distributed training framework is adopted and tested.

The rest of this paper is organized as follows. In Section II, The basic idea of deep reinforcement learning method is described. The problem description and system modeling are introduced and formulated in Section III. Section IV details our reinforcement learning based search method and Section V validates the proposed method based on simulations. Section VI concludes our paper and layouts future work.

2. Deep Reinforcement Learning
RL allows agents to learn from interactions with the environment while considering tasks in which the agent interacts with an environment through a sequence of observation $o$, action $a$ and reward $r$. The goal of the agent is to maximise future rewards by selecting actions according to a policy $\pi$. In [11, 12], DQN algorithm demonstrated human-level performance in complex and high-dimension spaces. The basic idea is to estimate the Q-function denoted as $Q(s, a)$ by using the Bellman equations as an iterative update

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$$  \hspace{1cm} (1)

where $\alpha$ is the learning rate and $\gamma$ is a discount parameter. Instead of policy $\pi$, the agent selects actions according to Q-function and is trained to learning the optimal Q-function.

DQN uses deep neural network with parameters $\theta$ to approximate the Q-function to adapt to complex scenarios and uses optimization algorithms like ADAM [13] to optimize the Q-function. Then the lost function is defined as $L(s, a|\theta) = (y - Q(s, a|\theta))^2$, where $y = r + \gamma \max_{a'} Q(s', a')$.

Then the parameter is updated using back-propagation:

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} L(s, a|\theta)$$  \hspace{1cm} (2)

3. Problem Description and System Modeling
The problem addressed in this paper is cooperatively searching for multiple stationary targets by a group of unmanned intelligent vehicles with limited sensing and communication capabilities. Unmanned intelligent vehicles are collectively called agent in the following content. The system modelling method mainly follows the method used in [6].

The surveillance region $G \in \mathbb{R}^2$ is assumed to be a rectangular plane region representing a battlefield or a sea surface. It is assumed that several targets are present in the surveillance region from the beginning of the search process and remain stationary throughout. In order to locate the targets, the whole surveillance region is partitioned into $G$ cells where each cell is associated with a probability of target existence within the cell, which constitutes a probability map for the whole region. Each cell is identified with its center $g = (x, y)$, where $x$ and $y$ are the coordinates of its center. For each cell, if target is present, then $\theta_g = 1$, otherwise, $\theta_g = 0$. The probability of target existence within cell $g$ is modeled as a Bernoulli distribution, i.e., $\theta_g = 1$ with probability $P(\theta_g = 1)$ and $\theta_g = 0$ with probability $1 - P(\theta_g = 1)$.

Suppose there are $N$ agents, all agents are assumed to move in the surveillance region, so the position of each agent can be described by its projection onto $G$, which is denoted as $\mu_{i,k}$ meaning the position of agent $i$ at time $k$. Agents have limited observation ability by sampling within its sensing region. The sensing region $S_{i,k}$ of agent $i$ at time $k$ is defined by sensing radius $R_s$, where $S_{i,k} = \{g \in G: \|g - \mu_{i,k}\| \leq R_s\}$ and $\|\cdot\|$ denotes the 2-norm for vectors. A cell is assumed to be wholly within $S_{i,k}$ if its center is within $S_{i,k}$. The sampling process is modeled by $d$(detection probability) and $f$(false alarm probability).
$z_{i,g,k}$ denotes the sample result of cell $g$ by agent $i$ at time $k$. The sample results can be positive ($z_{i,g,k} = 1$), or negative ($z_{i,g,k} = 0$). Then sampling process can be expressed as $P(z_{i,g,k} = 1|\theta_g = 1) = d$ and $P(z_{i,g,k} = 1|\theta_g = 0) = f$. On the other hand, agents’ communication ability is limited by communication radius $R_c$, which allows the agents to collaborate during searching. The neighbors of agent $i$ at time $k$ is defined as $M_{i,k} = \{j : \|\mu_{j,k} - \mu_{i,k}\| \leq R_c\}$ (including $i$ ). In Section IV, communication makes the probability map to be fused across all agents, improving the stability of search process. The system modeling is illustrated in Figure 1.

4. Method

Reinforcement learning allows agents to learn from interactions with the environment which can be established according to system modeling. In this section, the key ingredients of reinforcement learning based cooperative search method is introduced. Specifically, we begin with introducing the representation method of state in RL and the observation representation transformed from state. Then the reward function is described which is designed to guide the agents to learn the most efficient search policy. The learning process of agents implemented based on DQN is described in the end.

![Figure 1. Target search by agents with limited sensing and communication capabilities.](image)

4.1. State and Observation Representation

The policy expected to be learned by agents is essentially a mapping from observation to action, and the observation should contain the information needed to guide the agent. Therefore, we consider a state containing two pieces of information. One is the probability map which is the representation of target position information obtained by sampling. For the multi-agent system discussed here is distributed, each agent $i$ keeps an individual probability map $P_{i,k} = \{p_{i,k,g} \in [0, 1]: g \in \mathcal{G}\}$, where $p_{i,k,g}$ denotes the estimation of probability of target existence within cell $g$ at time $k$ by agent $i$. The initial value of $p_{i,k,g}$ is 0.5 meaning there is no information and should be updated during the search process. The commonly used method of updating the probability map by sampling is based on the Bayesian rule [5]. After a nonlinear transformation of $p_{i,k,g}, w_{i,g,k} \triangleq \ln \left(\frac{1}{p_{i,g,k}} - 1\right)$, the update can be simplified as

$$w_{i,g,k} = w_{i,g,k-1} + u_{i,g,k} \quad (3)$$

where
\[
\begin{align*}
v_{i,g,k} &= \begin{cases} 
\ln \left( \frac{L}{d} \right), & \text{if } z_{i,g,k} = 1, \\
\ln \left( \frac{1-f}{1-d} \right), & \text{if } z_{i,g,k} = 0.
\end{cases}
\end{align*}
\]

(4)

After defining w-map \( W_{i,k} \triangleq \{ w_{i,g,k} \in (-\infty, +\infty): g \in g' \} \), we use \( W_{i,k} \) instead of \( P_{i,k} \) as state for more efficient calculations.

Due to the limited sensing capability, it is difficult for the agent to obtain global information in a short time. Making w-map fused after every sampling between agents within sensing region helps improve this situation. The fused w-map can be formulated as

\[
\tilde{W}_{i,g,k} = \sum_{j=1}^{N} \alpha_{i,j,k} w_{j,g,k}
\]

where \( \alpha_{i,i,k} = 1 - \left( |M_{i,k}| - 1 \right) / N \), \( \alpha_{i,j,k} = 1 / N \) for \( j \in M_{i,k} (j \neq i) \) and \( \alpha_{i,j,k} = 0 \) for \( j \notin M_{i,k} \). The observation inputted into the policy network is a local w-map centered on coordinates of agent, denoted as \( \tilde{W}_{i,k} = \{ \tilde{w}_{i,g,k}: x_{g} - x_{\mu_{i,k}} \leq R_s \text{ and } y_{g} - y_{\mu_{i,k}} \leq R_s \} \), where \( x \) and \( y \) denote the \( x \)-coordinate and \( y \)-coordinate respectively. An illustration of \( \tilde{W}_{i,k} \) is showed in Figure 2a.

Another part of state is the coordinates of other agents which agent can adjust its policy according to. The agents can obtain a bit of cooperation ability by this information. For example, adding distance bonus when design reward makes agents tend to converge to a dispersed search policy.

The information of coordinates can be overwhelmed by w-map if it is inputted into the policy network directly. Therefore we map the coordinates of neighbouring agents within the scope of communication range to a matrix of size \( 2R_c \times 2R_c \) called c-map. The matrix of agent \( i \) at time \( k \) is denoted as \( C_{i,k} = \{ c_{i,k,g} \in \mathbb{R}: \| g - \mu_{i,k} \| \leq R_c \} \), where \( c_{i,k,g} \) is the superposition of Gaussian distributions centered on the neighbouring agent coordinates. \( C_{i,k} \) can be used as another observation which is demonstrated in Figure 2b. Just like the observation of w-map, we use a convolutional network to extract its features and separate it from each other by branch structure.

4.2. Reward Function
The reward given by environment according to state and action of agent is one of the key ingredients of the method, it assumes the role of guiding agents to explore and learn. The reward function designed here consists of four parts, namely target found reward, time consumption reward and guiding reward.
Target found reward $r^t$ motivates the agents to find as many targets as possible while ensuring a certain accuracy. A target is believed to be found when $P_{i,g,k} \geq \bar{p}$, where $\bar{p}$ is the preset threshold. Agent gets a positive reward $r^{t1}$ every time a target is found. Target number $M$ is assumed to be fixed, so the episode ends when $M$ targets are found and there will be a large positive reward $r^{t2}$ when all target locations are correct.

$$r^{t1}_{i,k} = \beta_1 \sum_{g \in G} 1_{p_{i,g,k} \geq \bar{p} \text{ and } p_{i,g,k-1} < \bar{p}} \quad (6)$$

$$r^{t2}_{i,k} = \beta_2 \prod_{g \in (g \in G; \theta_g = 1)} 1_{p_{i,g,k} \geq \bar{p}} \quad (7)$$

Time consumption reward $r^c$ motivates the agents to find the most efficient searching policy which is designed to be a piecewise linear function.

$$r^c = \beta_3 (k - k_e + \gamma_1 \max(k_1 - k_e, 0) + \gamma_2 \max(k_2 - k_e, 0)) \quad (8)$$

Where $\gamma_1$ and $\gamma_2$ is the additional weight for the current segment, and normally set to 1, $k_e$ is the episode step number, $k_1$, $k_2$ are preset segmentation points and $k$ is the preset maximum number of steps when the episode is forced to end. Obviously, time consumption reward is the same for all agents.

The rewards mentioned above are too sparse to provide guidance in the early stages of training, so guiding reward is designed to be a dense reward to equip agent with a better exploration ability. The decrease of global uncertainty is a good choice for this purpose. The global uncertainty is defined as

$$\eta_{i,k} = \frac{1}{G} \sum_{g \in G} \eta_{i,g,k} \text{, where } \eta_{i,g,k} = e^{-\beta_\eta |w_{i,g,k}|} \text{, } \beta_\eta \text{ is a constant.} \quad (9)$$

$$r^g_{i,k} = \beta_4 \frac{\eta_{i,k-1} - \eta_{i,k}}{\eta_{i,k-1}}$$

$\beta_1$, $\beta_2$, $\beta_3$ and $\beta_4$ are the weights of each part of reward and can be set empirically as 0.5, 5.0, 0.01, 1.0. $\beta_\eta$ is set as 2.0 in the simulations.

Finally, the reward function of agent $i$ at time $k$ is composed as

$$r_{i,k} = r^{t1}_{i,k} + r^{t2}_{i,k} + r^c + r^g_{i,k} \quad (10)$$

### 4.3. Integrated Algorithm

Training together with multiple agents can lead to instability in the training process, which is a classic problem for multi-agent reinforcement learning. The reason is that for one of the agents, the environment is changing as other agents are part of the environment. The method adopted here is to maintain the policies of other agents while training one agent, thereby ensuring the stability of the environment, and the policy is shared among all agents after a certain number of training steps, so that all agents can progress together. Unlike the algorithms of the centralized training and
Distributed execution framework, the replay buffer here stores only the experience of the currently trained agent. This approach makes the training process more stable and easier to converge, but cannot fully utilize the experience of all agents, resulting in increased training time.

Based on the state and observation representation and the reward function mentioned above, we would like to introduce a learning algorithm for cooperative search based on RL. DQN and DDPG are both classic reinforcement learning algorithms which are value-based and policy-based methods respectively. Both algorithms can be easily integrated into our method. In the experiments, we tried these two methods, and finally adopted DQN for better convergence.

5. Simulations

In the simulations, the RL algorithms are implemented based on the Tensorflow and the environment is built using Python. The implementation of the algorithm proposed runs on a computer with i5-9400 CPU and GTX 750 graphic card. The neural network designed here is relatively shallow, so it does not require a lot of computing power, and the general office computer can satisfy it.

In all simulations, agents are placed in a $25 \times 25$ surveillance region. Agents’ initial positions are evenly distributed on the left side of the region according to the number. At each moment, the agent samples, fuses, and moves. The sensing radius $R_s = 6$ and the communication radius $R_c = 8$. The detection and false alarm probabilities are set as $p = 0.9$ and $q = 0.3$.

Figure 3a shows the decreasing curve of the uncertainty of the RL based search (RLBS) method and random strategy in single-agent scenario. Figure 3b shows the decreasing curve of the uncertainty of the RLBS method and coverage control method in multi-agent scenario. Simulation results show that the RLBS method can effectively reduce the uncertainty of the surveillance region. Coverage control method is not suitable for single-agent scenario, so it is replaced by a random strategy. Table 1 shows searching performance (Average steps of episodes and accuracy) of RLBS method, Coverage control method.

**Figure 3. Uncertainty of the RL based search (RLBS) method, coverage control method and random strategy.**

**Table 1. Performance of methods.**

| No. agents | No. targets | Fused | RLBS          | Coverage control | Random          |
|------------|-------------|-------|---------------|------------------|-----------------|
| 2          | 1           | No    | 151(83.2%)    | /                | 315(67.2%)      |
| 2          | 1           | Yes   | 200(99.6%)    | 229(100%)        | 453(80.4%)      |
| 5          | 5           | Yes   | 329(100%)     | 338(100%)        | 874(80.76%)     |
method and random strategy at different scenarios. The specific setups of each scenario are expressed as number of agents, number of targets and whether there is a probability map fusion. Coverage control method is not suitable for scenarios without probability map fusion. As we can see in the table, RLBS method can find the target faster with similar value of accuracy, and has stronger robustness.

6. Conclusion
In this paper, an integrated learning algorithm for cooperative search based on reinforcement learning is proposed. The advantage of RL based method is that it can automatically find the optimal search policy through interaction with the environment with less manual design and can adapt to different environments through training. In order to integrate the reinforcement learning algorithms into the multi-agent search method, we made the following efforts: 1) we implemented state and observation representations transformed from probability map. 2) we designed a reward function to guide agents explore and learn including sparse and dense rewards. 3) we adopted a relatively stable distributed training framework. The performance and advantage of the proposed method are demonstrated in the simulation results. At the same time, there are some problems with this method, such as increased training time caused by training tricks. Potential future research avenues of the current work include reducing training time while keep stability and exploring adaptability of the method in complex environments.

Acknowledgments
This work was supported in part by Self-Organizing Network Marine Environment Multi-Parameter Measuring Instrument (2018YFF01013400), Key Technology and Demonstration Application of Unmanned Vessels in Complex Marine Environment Detection (17DZ1205001) and Research and Application Demonstration on Key Technologies of Unmanned Emergency Rescue Equipment (2017YFC0806700).

References
[1] Liu, Yugang , and G. Nejat . "Multirobot Cooperative Learning for Semiautonomous Control in Urban Search and Rescue Applications." Journal of Field Robotics 33.4 (2016):512-536.
[2] Miah, Suruz , et al. "Nonuniform Deployment of Autonomous Agents in Harbor-Like Environments." Unmanned Systems 02.04 (2014):377-389.
[3] Polycarpou, Marios M. , Y. Yang , and K. M. Passino . "A cooperative search framework for distributed agents." Intelligent Control, 2001. (ISIC '01). Proceedings of the 2001 IEEE International Symposium on IEEE, 2001.
[4] Yang, Yanli , A. A. Minai , and M. M. Polycarpou . "Decentralized cooperative search by networked UAVs in an uncertain environment." American Control Conference IEEE, 2005.
[5] Millet, Travis , et al. "Multi-agent Decentralized Search of a Probability Map with Communication Constraints*." Aiaa Guidance, Navigation, & Control Conference 2013.
[6] Hu, Jinwen, et al. "Multiagent information fusion and cooperative control in target search." IEEE Transactions on Control Systems Technology 21.4 (2012): 1223-1235.
[7] Bera, Titas, Rajarshi Bardhan, and Sundaram Suresh. "An Integrated Decision and Control Theoretic Solution to Multi-Agent Co-Operative Search Problems." arXiv preprint arXiv:1704.07158 (2017).
[8] Otte, Michael , M. Kuhlman , and D. Sofge . "Competitive target search with multi-agent teams: symmetric and asymmetric communication constraints." Autonomous Robots (2017).
[9] Adepegba, Adekunle A., Suruz Miah, and Davide Spinello. "Multi-agent area coverage control using reinforcement learning." The Twenty-Ninth International Flairs Conference. 2016.
[10] Zhao Xu, Yang Lyu, Quan Pan, Jinwen Hu, Chunhui Zhao, Shuai Liu. "Multi-vehicle Flocking Control with Deep Deterministic Policy Gradient Method." 2018 IEEE 14th International Conference on Control and Automation (ICCA). 2018.
[11] Mnih, Volodymyr , et al. "Playing Atari with Deep Reinforcement Learning." Computer Science
[12] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529.

[13] Kingma, Diederik P., and J. Ba. "Adam: A Method for Stochastic Optimization." Computer Science (2014).