Deep reinforcement learning approach for UAV search path planning in discrete time and space

Najoua Benalaya, Ichrak Amdouni, Cédric Adjih, Anis Laouiti, Leila Azouz Saidane

Deep Reinforcement Learning Approach for UAV Search Path Planning In Discrete Time and Space

Najoua Benalaya *, † Ichrak Amdouni*, Cedric Adjih ‡, Anis Laouiti †, Leila Azouz Saidane *

*ENSI, Tunisia, University of Manouba, surname.name@ensi-uma.tn
†Telecom SudParis, France, najoua_benalaya@telecom-sudparis.eu, anis.laouiti@telecom-sudparis.eu
‡INRIA Saclay, France, cedric.adjih@inria.fr

***Abstract*—Path planning for search missions carried out by Unmanned Aerial Vehicles (UAVs) is a challenging problem. This is due to UAV limited energy budget and the importance of time for search operations. The objective of this study is to come up with an approach to minimize the total search time required to locate a specific target. To achieve this, we deployed a deep reinforcement learning (DRL) model based on the Proximal Policy Optimization (PPO) algorithm to solve the combinatorial optimization problem of UAV search path planning within a minimized search time. A smart reward formulation is designed to achieve the learning goal, fulfill the search requirement, and encourage the agent to select search paths that minimize search time. In addition, we employed Optuna hyperparameter optimization framework to systematically select optimal parameters for the PPO model. Most importantly, thanks to the state representation we considered, the model is generalized and adaptable to various search environments. The PPO model succeeds to compute an accurate search path to be followed by the UAV searcher. Results of the model are compared with results previously obtained with a linear program. We found that the PPO achieves almost the same expected search time, which proves the great relevance of the reward design and the hyperparameters selection we made.**

***Index Terms*—Deep Reinforcement Learning, PPO, UAVs, Search Path Planning, Reward Design, Optuna, Hyperparameters Search**

I. Introduction

A. Context

Given their effectiveness and ease of deployment, unmanned aerial vehicles (UAVs) have been successfully used to execute a wide range of missions in several fields. In particular, UAVs are employed to carry out search operations [1]. Generally, a search mission is an operation where a searcher (e.g. human, UAV, robot, helicopter, etc) tries to localize a stationary or dynamic missing target. Stationary targets could be an injured person, victims, or any other item of interest (ship, mine, animal, etc). The technical capabilities of UAVs (sensors and high-resolution cameras) offer an aerial perspective during the early stages of a search and allow them to detect the target presence successfully.

To be able to autonomously localize targets, UAVs need to apply effective path planning strategies. One way to plan a search trajectory is to determine the sequence of sub-regions to be inspected inside a discretized area of interest (AOI) in order to maximize the chances of locating the target. However, the above task is complex and challenging especially when the UAV does not have knowledge about the AOI. Furthermore, as the onboard resources of the UAVs are limited (e.g battery), and because locating one target is usually a time constraint task, the search path must meet these delay constraints while increasing the chances of locating targets or individuals.

In the operational research literature, the problem of planning search paths is related to two families of problems: (i) coverage path planning problems (CPP) and (ii) optimal search path problems (OSP). These two types of problems are distinct but they are equally well suited to solve detection, search, and monitoring problems. Moreover, both are challenging and complex to solve. Trummel et al. [2] have proved that the OSP problem, for one searcher and a single stationary target, is one of the NP-complete optimization problems. Furthermore, CPP is a variant of the well-known Travelling Salesman (TSP) combinatorial optimization problem. It involves determining the sequence of waypoints that ensure optimal and complete coverage of an AOI, such as patrol and area exploration missions.

The literature published several studies on the optimization of the robot and UAV search trajectory and CPP problem. These studies have employed a variety of approaches to address the problem. One approach is the exact methods specifically the linear programming (LP) approach. The LP-based approach formulates mathematically the problem objective and constraints and outputs an optimal solution or trajectory of the investigated problem. For instance, Benalaya et al. [3] investigated the problem of UAV cattle search path planning and introduced a linear program capable of computing an optimal search path with minimum expected search time. Furthermore, the use of conventional methods such as the Boustrophedon motion, heuristic techniques (Greedy Search, Bio-inspired Algorithms),and most recently the deployment of Reinforcement Learning (RL) and Deep Learning (DL) models [4]. The RL and DRL models provide promising results in solving UAVs trajectory optimization problems [5] and assist in overcoming the limitations of the conventional methodologies [1] [6]. Generally, RL and DRL are widely deployed to address path optimization issues, and this is because these

[1]The conventional methods cannot be generalized to a larger scale

approaches provide suitable and adaptable models that can generalize to environment changes without model retraining from scratch.

RL based strategies enable an agent to learn how to behave or how to execute a sequence of optimal decisions in an interactive environment. The RL environment can be modeled as a "Markov Decision Process" (MDP) defined by the tuple $\mathcal{H} = \langle S, A, P, R, \gamma \rangle$, where S is the set of states, A is the set of actions an agent could perform, P is the state transition probability, R is the reward gained as a consequence of the executed action, γ is a discount factor $\in [0, 1]$ that indicates how valuable immediate rewards are over delayed ones. Given a set of rewards and penalties, the RL agent learns an optimal policy $\pi^*(a|s)$ that allows it to select the best action.

Many algorithms implement the RL strategies. For our case, we propose a Deep Reinforcement Learning (DRL) approach based on the Proximal Policy Optimization (PPO) algorithm [7]. We consider a grid-world AOI divided into cells. The grid has a probability distribution, that captures the probability of each cell to contain the target. During training phases, the UAV learns the optimal path to find the target within the shortest total expected search time. Besides, the model is generalized; it enables the UAV agent to find the target on any map.

B. Contributions

The major contributions of this conducted research are:

- Design of a novel UAV control method for UAV search path planning based on the PPO Algorithm.
- Designing a smart reward function that fulfills the major learning objective (minimum expected search time).
- An extensive search of optimal hyper-parameters of the PPO model using the Optuna framework.
- Developing the model and comparing it with the results given by a linear program. Results show that our model is very close to the optimal.

The remaining sections of the paper are structured as follows. Section II reviews studies about RL and DRL algorithms for solving combinatorial optimization problems related to trajectory optimization. Section III describes the system model while section IV introduces the PPO model for UAV search path planning problem. The simulation settings and results are given in Section V. Section VI summarizes the paper.

II. Related Work

Table I recaps the research studies based on RL and DRL methods to solve trajectory optimization problems, namely the CPP and the TSP problems. The classic trajectory optimization problems can be modeled as graph optimization problems. In this area, researchers used the RL to learn heuristics via training neural networks designed to operate on graph-structured data and known as Graph Neural Networks (GNN). The paper [9] tackled Vehicle Routing problem (VRP) and TSP where authors used Graph Attention network (GAT) based on encoder-decoder structure and trained with REINFORCE algorithm. The authors created a generalized learning framework capable of generating effective heuristics for diverse optimization problems within the realm of routing.

In another work, [8] proposed to train Pointer Networks (Ptr-Net) models using policy gradient RL methods to solve TSP. This type of network generates a sequence of cities in the order according to which they should be visited. Recent works on two variants of TSP problem are given in [13] and [15]. These two references focused on TSP with varying traffic conditions and customer demands while considering the need of refueling. In the first study the authors deployed two tabular methods of SARSA and Q-learning RL algorithms which do not generally perform well with high dimensional state and action space. However, in the second study the authors deployed a GNN network and trained it with policy gradient algorithm.

For computing an sub-optimal search path, [10] developed a Q-Learning model to solve the CPP problem in a maritime environment. This work used a probability distribution map that presents the probability of target containment in the AOI. The proposed model outperforms other classic algorithms like A star. However, it cannot be generalized, which means that a new training is required when a new probability distribution is adopted. Furthermore, RL algorithms were applied in target searching in general, including the visual localization. For instance, [11] deployed a Convolution Neural Network (CNN) and DQN for visual feature extraction and direction learning.

III. Problem Description and System Model

A. Problem Description and Assumptions

In this paper, we investigate the problem of planning an efficient search trajectory followed by a UAV searcher to locate a stationary target in discrete time and space. The objective is to determine a path that guarantees the target localization within the minimum expected search time. Due to the limited computational power of the UAV, DRL model was loaded onto the UAV's onboard system but was not fully trained onboard.

To solve this issue we consider the following assumptions:

A1: We consider a two-dimensional search space decomposed based on the regular grid decomposition technique (each sub-area is equivalent to a cell).
A2: There is only one stationary target inside each cell.
A3: The searcher is one autonomous UAV.
A4: The searcher's movement is constrained.
A5: The search time is discrete. The time is divided into multiple time periods.
A6: The UAV is equipped with perfect sensors enabling it to detect the target if both of them are in the same cell at the same time.

B. Search Environment Model

In search problems-related literature, researchers often partition the search space into smaller sub-regions, which is a crucial step for autonomous agents where the searcher makes decisions. This holds significance in both OSP and

TABLE I: Summary of Studies on Path Optimization Problems with RL and DRL

Reference	Problem	Objective	Model
Bello et al. [8]	TSP, VRP	Maximizing the route cost	Pointer Networks and policy gradient RL
Kool et al. [9]	TSP, VRP	Maximizing the route cost	GAN based encoder-decoder architecture and the REINFORCE RL algorithm
Ai et al. [10]	CPP in maritime	Maximizing the cumulative probability of success among the traversed Route	Tabular RL: Q-Learning
Liu et al. [11]	Target localization	Robot search path planning for a trapped person	Prioritized DQN algorithm based on CNN
Shurrab et al. [12]	Target localization	UAV search path planning for radiation source	CNN +PPO
Ottoni et al. [13]	TSP with refueling	Minimizing refueling cost and distance	Q-learning and SARSA
Zhang et al. [14]	Dynamic TSP (varying traffic conditions)	Minimizing the route distance cost	GNN trained with policy gradient method

CPP problems by making it more convenient to establish the sequence for searching or covering cells [16].

Our system model aligns with CPP assumptions. Hence, we consider a grid world search environment $\mathcal{G} = [0, g] \times [0, g]$ which consists in a finite set of grids or cells $\mathcal{J} = \{1, \dots J\}$. Each cell is designed to match the drone footprint, assuming perfect detection when the UAV is at the cell center. The search theory uncertainty is addressed with a **probabilistic distribution** P known priory by the searcher and represents a key input for the model. Indeed, $P = \{p_j, j \in \mathcal{J}\}$, where each cell j has a probability p_j that indicates the probability that this cell contains the target. Since the target is stationary the equation (1) is valid. Consequently, it is guaranteed that the target can always be found, albeit potentially requiring a visit to every cell in the worst-case scenario.

$$\sum_{j \in \mathcal{J}}^{J} p_j = 1 \tag{1}$$

The UAV initially sweeping the AOI could be a method to estimate the prior probability distribution.

C. UAV Model

The search mission takes place in a limited set of cells over a finite set of time periods $\mathcal{T} = \{1, \dots T\}$. In the first time period the UAV could be located in any initial cell j. It inspects the initial cell. Then, for each time period, it autonomously navigates through the cells of the computed search trajectory (given by the established model). It inspects the cell it currently occupies to locate a target. Once the target is located, the search mission is over. Due to the UAV kinematics constraints, we assume that the drone movement inside the search space is constrained. In other words, if the UAV is in cell j at the time period t it is allowed to visit only one of the neighbor cells in the following time period $t + 1$. Another significant assumption that we have considered is that the UAV has perfect sensors that allow it to detect perfectly the target if both of them are in the same cell. Thus, if the drone scans cell j at time period t without finding the target, then the probability of target containment p_j in $t + 1$ time period is updated to 0.

IV. PPO based Model for UAV Search Path Planning

Due to the limitations of tabular-based methods in the case of large state and action space, researchers integrated neural networks to approximate the optimal policy. This can be achieved either by directly learning a parameterized policy from experiences via policy-based methods: proximal policy optimization (PPO), deep deterministic policy gradient (DDPG) or by learning an intermediate function via value-based methods: Deep Q-networks (DQN), Double DQN, from which the policy can be deduced [17].

A. Overview about PPO

PPO wich was introduced by [7], belongs to the class of policy gradient methods and is designed to optimize the policy of the RL agent. Generally, policy gradient methods revolve around iteratively refining the policy of an agent denoted π, maximizing the cumulative reward within a given environment using a policy objective function L_{PG} given by Equation 2:

$$L_{\text{PG}}(\theta) = \mathbb{E}_t \left[\log \pi_\theta(a_t|s_t) \hat{A}_t \right] \tag{2}$$

Where θ is the vector of policy parameters to be updated and $\hat{A}_t$ is the estimated advantage.

PPO is designed to tackle some instability issues related to earlier policy gradient methods through the use of a "clipped" objective function. Initially, PPO operates by collecting a batch of experiences through interaction with the environment. Then it uses these experiences to compute an estimate of the policy gradient. The clipped objective function is then optimized to update the policy in a way that encourages actions with high rewards and discourages actions with low rewards while staying close to the previous policy. It limits the change in the policy during each update, by forcing the probability ratio $r_t(\theta)$ between old and new policies to stay within a small interval around 1, precisely $[1 - \epsilon, 1 + \epsilon]$, where ϵ is a hyper-parameter.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right] \tag{3}$$

The strengths of PPO are in its simplicity and effectiveness for a wide range of environments, making it a popular choice for researchers and practitioners in the field of reinforcement

learning. Its stability and relatively simple implementation have contributed to its widespread adoption and use in various applications, ranging from robotic control to game-playing. It is worth to mention the PPO input is the state of the environment and the output is a probability distribution of the possible actions given the fitted state.

B. Proposed Model

This section is devoted to the establishment of the PPO based model for UAV search path planning.

1) State and Action Representation: As we mentioned before, the objective of deploying a PPO model is to learn the search path followed by the UAV that minimizes the total expected search time and guarantees an accurate detection of the target. The environment is described by means of a probability distribution map which illustrates the target probability of containment. This probability distribution is the key map used by the RL agent to decide which cell to inspect in the next step.

Fitting the RL agent with a global observation of its environment is a major factor in the agent success in learning the optimal policies [17]. Hence, in the context of search path planning issue, the state should be represented in a way that allows the agent to be aware of its position inside the environment, and also aware of its surroundings which means the probability distribution of all the cells of the AOI.

$$S_t = [x_t, y_t, p_1, \ldots, p_J] \tag{4}$$

Based on this intuition, we present the state as a unified layer including the UAV coordinates (x, y) in the corresponding time step t and the probability distribution of target containment in all the cells. The probability distribution is updated according to the agent action. If the agent selects to inspect cell j in time step t then p_j become null in the following time steps.

As we mentioned before in the assumptions, the agent movement in the 2D grid world is constrained. For each action taken, we assume that the agent navigates at a constant speed. The actions are as follows:

$$A_t = \{Left, Right, Up, Down\}$$

2) Reward Design: The reward function serves as the guiding principle for learning. A well-crafted reward function plays a crucial role in discerning the advantageous and disadvantageous actions of the agent throughout the learning process. By appropriately assigning rewards to various actions, the agent can efficiently work towards achieving its objectives in the shortest possible time.

Optimizing the search path involves addressing multiple learning sub-goals to meet the major learning goal, which is minimizing the total expected search time. We carefully identify sub-goals as :

- Prioritize the exploration of cells with higher probabilities of containment.
- Avoid going out of the AOI bounds and revisiting cells.
- Visit all the cells with $p_j \neq 0$.
- Avoid revisiting cells.

The challenging nature of these learning goals poses significant challenges in formulating an effective reward function. In a previous work [3], we formulated the problem of UAV cattle search path aiming at minimum search time as a mathematical optimization problem and solved it via Linear Programming. The paper introduced a MILP (Mixed Integer Linear Programming) where the objective function is given by formula 5. Our purpose was to minimize this function respecting its multiple constraints.

$$\text{Expected search time} = \sum_{j \in \mathcal{J}} p_j \cdot t_j \tag{5}$$

Inspired by the formulation (5) we designed a reward formulation that guides the agent to minimize the total expected search time. We adopted the concept of goal reward which involves assigning one final reward in the final step of each episode. Consider $\mathcal{S}$ is the sequence of cells that are visited during an entire episode, the final reward is formulated as follows:

$$R_{\text{final}} = -\sum_{j \in \mathcal{S}} p_j \cdot t_j - \left[\beta \cdot (T+1) \cdot (1 - \sum_{j \in \mathcal{S}} p_j)\right] \tag{6}$$

Compared to formula 5, formula 6 introduces :

- $\left[\beta \cdot (T+1) \cdot (1 - \sum_{j \in \mathcal{S}} p_j)\right]$: represents a penalty that is given to the UAV when it finishes an episode without visiting all cells. In other words, this penalty term adjusts the final reward based on unvisited cells and episode length, ensuring that incomplete grid coverage leads to a worst reward. β is a parameter that is equivalent to the number of additional episodes to perform in case of non fill coverage.
- $(1 - \sum_{j \in \mathcal{S}} p_j)$ is the sum of the probability of the unvisited cells during the episode. If the agent visits all the cells once, then this value becomes null and the final reward is reduced to the expected search time.
- $(T+1)$ represents the episode length (in terms of number of time steps) plus one. It represents the time dedicated to the search for additional episodes.

Because our objective is to minimize a value which is the total expected search time, we added the negative sign so the reward becomes a negative value.

V. Simulation

We considered a grid-world as the AOI and the environment of RL algorithm. The size of the grid as selected to 4x4. The agent moves inside this grid-world by selecting which direction it takes (among the four directions). The UAV is trained during 3125000 episodes. Each episode has a fixed length of 16 time steps to give the agent the ability to visit all cells and achieve a probability of detection equal to 1. For each new training episode, we set a new map and a new initial position for the UAV. This important feature allows us

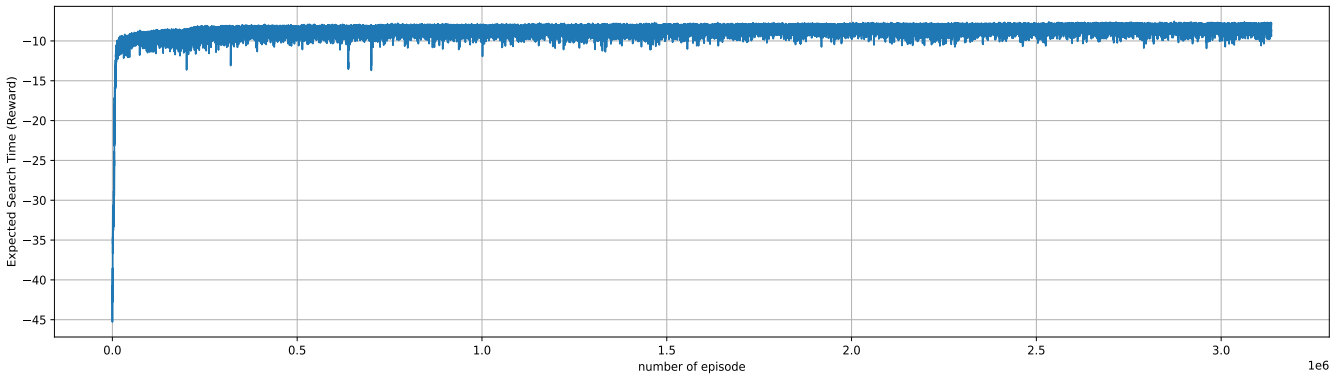


Fig. 1: The reward received during the training

to have a **generalized model**. The model was implemented in Python using the Stable-Baselines3 on a machine with Intel (R) Xeon (R) 2.93GHz CPU, RAM 8 GB with 4 cores. For the generation of the probability distribution, we implemented a python map generator model responsible for generating random and specific probability maps. [2]

A. Optuna Hyper-parameters Search

We employed Optuna, an open-source hyper-parameter optimization framework [18], to systematically search for the optimal hyper-parameters of our PPO model. Optuna optimization process involves setting the search space which is represent a valid range of values for each hyper-parameter and the definition of an objective function, the core of the hyper-parameter tuning process. The objective function is used to assess the performance of different hyper-parameter configurations. Optuna incorporate a state-of-the-art algorithms for efficiently searching large spaces and pruning unpromising trials for faster results. The Tree-Parzen Estimator (TPE) algorithm is used to sample the hyper-parameters and adaptively select the set of hyper-parameters to be tried next based on the history of trials. The Asynchronous Successive Halving (ASH) algorithm is used to prune unpromising trials based on intermediate target values, reducing the computational resources required to optimize the hyper-parameters [18]. We initiated a total of 300 trials within Optuna to identify the hyper-parameter configuration that yielded the best results for our specific task. We select a particular hyper-parameters that largely affect the learning efficiency. By leveraging Optuna, we were able to efficiently explore a wide range of hyper-parameter combinations and select the configuration that optimized our model performance and achieved the best cumulative reward average. Table II illustrates the Optuna optimal hyper-parameters selected for our model.

B. Experiments and Results

Figure 1 illustrates the reward evolution during the training. The plot shows that in the beginning of training the reward increases linearly until it converges after 200000 episodes, which proves that PPO converges fast. To validate the performance of the proposed model we compare the accumulated reward for the PPO model and the optimal solution given by the linear program that we have developed in a previous work [3]. Figure 2 shows that for 9 different probability maps and starting from different initial positions the search trajectory computed by the linear program achieved a slightly better-expected search time with maximum difference ratio 3% for the initial position 9. The small difference ratio proves that the proposed model is efficiently capable of determining a sub-optimal search path. Figure 3 presents the computed search trajectory by the PPO model and the linear program using the same probability map and the same UAV initial position. For instance, in Figure 3.(a) and Figure 3.(b), the cells with the highest probability of containment are 14, 2, 15 then 7. Both programs reach the cell 14 by in $t = 3$, cell 15 at $t = 4$, and cell 7 at $t = 8$. However, in PPO trajectory cell 2 is visited one time period later than the linear program. This highlights that PPO model select almost the optimal sequence of cells.

TABLE II: Optuna Best Hyper-parameters

Hyper-parameters	**Value**
Learning Rate	0.00043951622909747524
Clip range	0.22189614603726104
γ	0.9997438077898142
Entropy coef	0.050649945359319534
GAE Lambda	0.8615743551574223
Network Architecture	4 Hidden layers 64 unit
Batch Size	624

VI. Conclusion and Future Scope

The paper introduces a PPO based model to solve a combinatorial optimization issue denoted UAV search path planning. The problem at hand revolves around designing an efficient search path that guarantees the target localization within the minimum expected search time. A deep observations allowed us to design a smart reward function that encourages the agent to learn the search trajectory with the minimum search time. The model is fitted with a probability map of target

[2]We can refer to the probability distribution by probability map

[3]POC is the probability of target containment in each cell in ascending order

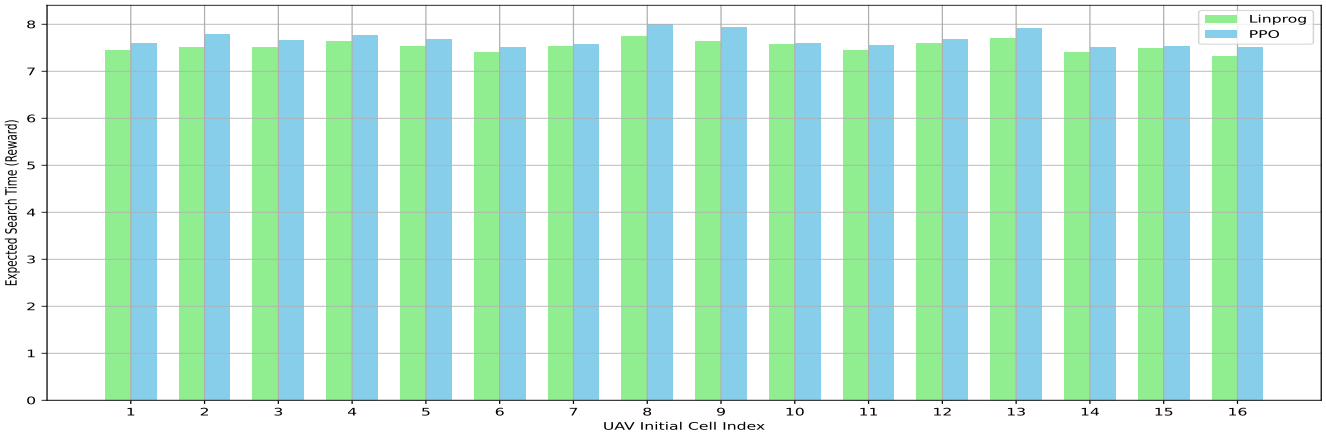
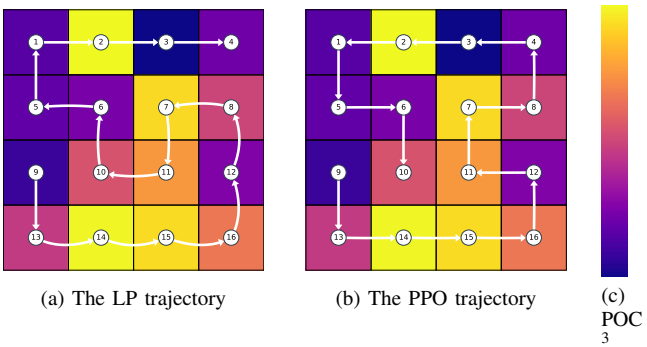


Fig. 2: The average expected search time given different initial positions for the Linear program and the PPO model



(a) The LP trajectory (b) The PPO trajectory (c) POC 3

Fig. 3: The computed search trajectories

containment in the AOI and outputs the search path. The model is generalized, we are not obliged to retrain the model if the UAV hovers inside a new map. In addition, a hyper-parameters tuning framework was deployed to optimize the hyper-parameters of PPO model. The model was trained with the best hyper-parameters and it provided a successful performance for learning an efficient search trajectory. The search trajectories of the PPO model are compared with the optimal search trajectories computed via a linear programming. The results were remarkably very close with a difference rate up to 3%. The model we proposed is generalized but it does not scale to larger probability maps. Hence, our perspective is to address the scalability issue with RL for UAV search path planning problem.

Acknowledgment

This research was conducted under the project PHC-Maghreb ANGEL 24MAG18.

References

[1] Mingyang Lyu, Yibo Zhao, Chao Huang, and Hailong Huang. Unmanned aerial vehicles for search and rescue: A survey. Remote Sensing, 2023.

[2] KE Trummel and JR Weisinger. The complexity of the optimal searcher path problem. Operations Research, 1986.

[3] Najoua Benalaya, Cedric Adjih, Anis Laouiti, Ichrak Amdouni, and Leila Saidane. Uav search path planning for livestock monitoring. In 2022 IEEE 11th IFIP International Conference on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN), pages 1–6. IEEE, 2022.

[4] Tan et al. A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. IEEE Access, 2021.

[5] Kurunathan et al. Machine learning-aided operations and communications of unmanned aerial vehicles: A contemporary survey. IEEE Communications Surveys & Tutorials, 2023.

[6] Qi Wang and Chunlei Tang. Deep reinforcement learning for transportation network combinatorial optimization: A survey. Knowledge-Based Systems, 2021.

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[8] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940, 2016.

[9] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! arXiv preprint arXiv:1803.08475, 2018.

[10] Bo Ai, Maoxin Jia, Hanwen Xu, Jiangling Xu, Zhen Wen, Benshuai Li, and Dan Zhang. Coverage path planning for maritime search and rescue using reinforcement learning. Ocean Engineering, 241:110098, 2021.

[11] Yanglong Liu, Zuguo Chen, Yonggang Li, Ming Lu, Chaoyang Chen, and Xuzhuo Zhang. Robot search path planning method based on prioritized deep reinforcement learning. International Journal of Control, Automation and Systems, 20(8):2669–2680, 2022.

[12] Mohammed Shurrab, Rabeb Mizouni, Shakti Singh, and Hadi Otrok. Reinforcement learning framework for uav-based target localization applications. Internet of Things, 23:100867, 2023.

[13] André LC Ottoni, Erivelton G Nepomuceno, Marcos S de Oliveira, and Daniela CR de Oliveira. Reinforcement learning for the traveling salesman problem with refueling. Complex & Intelligent Systems, 8(3):2001–2015, 2022.

[14] Zhang et al. Solving dynamic traveling salesman problems with deep reinforcement learning. IEEE Transactions on Neural Networks and Learning Systems, 2021.

[15] Zhang et al. Path following control for uav using deep reinforcement learning approach. Guidance, Navigation and Control, 1(01), 2021.

[16] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In Field and service robotics, pages 203–209. Springer, 1998.

[17] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. Foundations and Trends® in Machine Learning.

[18] Takuya Akiba, Shotaro Sano, Takeru Yanase, Toshihiko Ohta, and Masanori Koyama. Optuna: A hyperparameter optimization framework. https://optuna.org/, 2019.