Multi-Robot Path Planning for Mobile Sensing through Deep Reinforcement Learning

Yongyong Wei, Rong Zheng
Department of Computing and Software, McMaster University
1280 Main St. W., Hamilton, ON, Canada
Emails: {weiy49, rzheng}@mcmaster.ca

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488669

***Abstract*—Mobile sensing is an effective way to collect environmental data such as air quality, humidity and temperature at low costs. However, mobile robots are typically battery powered and have limited travel distances. To accelerate data collection in large geographical areas, it is beneficial to deploy multiple robots to perform tasks in parallel. In this paper, we investigate the Multi-Robot Informative Path Planning (MIPP) problem, namely, to plan the most informative paths in a target area subject to the budget constraints of multiple robots. We develop two deep reinforcement learning (RL) based cooperative strategies: independent learning through credit assignment and sequential rollout based learning for MIPP. Both strategies are highly scalable with the number of robots. Extensive experiments are conducted to evaluate the performance of the proposed and baseline approaches using real-world WiFi Received Signal Strength (RSS) data. In most cases, the RL based solutions achieve superior or similar performance as a baseline genetic algorithm (GA)-based solution but at only a fraction of running time during inference. Furthermore, when the budgets and initial positions of the robots change, the pre-trained policies can be applied directly.**

***Index Terms*—Mobile Sensing, Informative Path Planing, Multi-agent Reinforcement Learning**

I. Introduction

Traditionally, to collect spatial data, one needs to deploy static sensors with wireless communication capability at pre-selected locations to form a wireless sensor network (WSN) [1]. Although WSNs are ideal for long-term environmental monitoring, due to their huge infrastructure costs and setup overhead, they are not suitable for applications that only need to collect data *infrequently or on demand*. On the other hand, the past decade has witnessed remarkable advances in artificial intelligence and robotics, resulting in affordable mobile platforms such as autonomous underwater vehicles, drones, gliders, and rovers, etc [2]. These platforms can be utilized to collect environmental data, perform close reconnaissance, and carry out search and rescue missions on demand and autonomously. However, since they are typically battery-powered and have limited lifetime or travel distances, multiple units need to be dispatched to cover a large geographical area to complete the tasks in a timely manner.

In this paper, we investigate the problem of planning efficient paths for multiple mobile robots for spatial sensing in large areas. We model the underlying spatial phenomenon as a Gaussian Process (GP) [3]. GPs have been widely adopted to characterize spatial data distribution like humidity or temperature. With appropriate hyper-parameters, the models can be utilized to predict measurements give any location. Under GPs, the quality or *informativeness* of data collected can be characterized by information-theoretical measures such as mutual information and differential entropy [4], [5]. The battery capacity of mobile robots imposes budget constraints that limit the distances they can travel. The goal of multi-robot informative path planning (MIPP) is to find paths in a target area that maximize the informativeness of data collected subject to budget constraints.

The single-robot informative path planning (IPP) has previously been investigated in [5], [6]. MIPP is a non-trivial extension to IPP in that robots need to coordinate their path selections in order to maximize the total informativeness. There are many forms of cooperation among robots. On one extreme (non-cooperative), robots can plan their paths independently, completely disregarding the presence of others. On the other extreme, multiple robots can be viewed as a single robot so that in each step a joint decision is made regarding the actions of all robots. Between the two ends of the spectrum, robots can share information that guides cooperation but allows distributive path planning. Intuitively, non-cooperative path planning often leads to suboptimal decisions whereas the computation complexity of joint decision making grows exponentially with the number of robots. A fundamental question is thus, *is there a cooperation mechanism that is both near optimal and computationally efficient for MIPP?*

We formulate the MIPP problem as a multi-agent Markov Decision Process (MMDP) [7] and investigate multi-agent reinforcement learning (MARL) based solutions. For simplicity, we only consider Q-learning algorithms though other methods such as policy gradient or actor-critic can be adopted as well. The main challenge of MARL is that the joint action space for all agents is exponential with respect to the number of agents, which makes it hard to learn and execute a policy efficiently. For example, suppose that for a multi-agent system, the joint Q-function $Q_{jt}(s, \mathbf{u})$ is known, where s is the state of the system and $\mathbf{u}$ is the joint action. An optimal policy can then be found by selecting the joint action with the maximum Q-value in each state. If the number of agents is 6 and each agent has 20 actions available, one needs to evaluate $Q_{jt}(s, \mathbf{u})$ for $20^6 = 64,000,000$ times to find the optimal joint action for state s. Moreover, a large joint action space often leads to longer convergence time or suboptimal policies as we show

978-1-6654-0325-2/21/$31.00 ©2021 IEEE

experimentally in Section V.

To tackle the problem of joint action space explosion in MIPP, we develop two cooperative learning strategies. In the first strategy, called *independent Q-learning with credit assignment* [8], agents learn independent Q-functions. In MIPP, cooperation is achieved by sharing joint states and splitting one-step (team) reward among agents. How credits are assigned will affect the convergence and optimality of the resulting policies. In extreme cases, agents may be discouraged from learning good policies due to unfair assignments. In this work, we consider two credit assignment schemes, namely, equal split and difference rewards.

In the second strategy, we observe that in MIPP, robots only need to cooperate spatially instead of temporally. Therefore, one can transform MMDP to a single-agent decision problem by sequentially planning the robots, called *sequential rollout*. In this scheme, cooperation is achieved by taking into account subsequent agents' actions in updating the current agent's Q functions. The size of the action space in both strategies does not depend on the number of agents. In the previous example, to decide the best action in state s, one only needs to evaluate learned Q functions 120 times and 20 times in independent Q-learning and sequential rollout, respectively.

To evaluate the performance of the RL-based MIPP solutions, we consider the application of robotic indoor site survey for Wi-Fi Received Signal Strength (RSS), widely used in indoor positioning systems [9]. Real measurements from two indoor areas are used in evaluation. In policy training, we find that independent learning with difference rewards [10] has a comparable performance with sequential rollout based RL, and outperforms independent learning with equal split. Extensive path planning experiments are carried out using the RL policies for both homogeneous and heterogeneous robot teams. In most cases, the RL-based approaches achieve higher or comparable total awards as a non-learning genetic algorithm (GA) based approach at only a fraction of running time. Furthermore, one main advantage of the RL-based approaches is the ability to handle different budgets and initial positions of agents without retraining.

The rest of this paper is organized as follows. In Section II, we review related work in mobile sensing, informative path planning and multi-agent reinforcement learning. Next, the MIPP problem is formally formulated in Section III. We present the proposed solution in Section IV. Experimental results are shown in Section V, and we conclude our work and outline our future research direction in Section VI.

II. Related Work

In this section, we review some relevant works in mobile sensing and informative path planning, followed by a brief introduction to MARL.

A. Mobile Sensing with Robots

Autonomous mobile robots have been widely investigated as mobile sensing platforms for a variety of tasks. They have advantages of being able to navigate to fields where humans cannot. Comparing to statically deployed WSNs, they offer more flexibility and incur lower costs for environment monitoring at a scale. In [11], an autonomous indoor environment surveillance system is presented using mobile robots. In [12], the authors propose to use robots to collect Wi-Fi and magnetic fingerprints for indoor localization. Ishida et al. [13] survey existing work that use mobile robots for chemical substance sensing. This line of works mainly focus on specific applications of mobile robotic sensing and the design of the robots and sensors. In this paper, we focus on path optimization, which can be beneficial to all the relevant applications.

B. Informative Path Planning

IPP originated from the problem of informative sensor placement [4] in wireless sensor networks, where static sensors are placed at locations that maximizing the informativeness of data collected. Informative sensor placement though NP-hard can be approximated with a constant approximation ratio under additive or submodular utility functions. In contrast, since IPP requires to determine way-points in a target area as well as efficient paths among these way-points under a budget constraint, the problem is markedly harder. Therefore, heuristic strategies such as GA [5] are investigated for computation efficiency. In [6], way-points are added incrementally and a TSP solver is utilized to generate paths. In [14], the authors model the IPP problem as a sequential decision process and develop an RL based solution.

To shorten the data collection time, one solution is to use multiple robots concurrently. This poses further challenges to the IPP problem since one needs to plan paths and coordinate a team of robots. In [15], the authors propose a sequential allocation algorithm to the MIPP problem. Paths are generated one by one, and the utility function is updated by committing to previously selected paths. Since the spatial correlation of measurements generally decreases as the distances between the observations decrease, the authors in [16] approximated the correlation by only considering that among m steps along a path. A stage-wise dynamic programming algorithm is then proposed to optimize the paths based on the criteria of entropy or mutual information. In [17], the authors divided a target area using Voronoi diagram tessellation, where each robot is assigned to a partition to sense. With heterogeneous budgets or irregular terrains, it is difficult to find optimal partitions that can maximize team utility.

C. Multi-agent Reinforcement Learning

RL [8] experienced a resurgence of interests in recent years by using deep neural networks as a function approximator, known as Deep RL. The goal of RL is to learn a decision policy to maximize expected future reward. RL methods can be classified into value-based (e.g., Deep Q-Learning) and policy-based (e.g., policy gradient) solutions. Single agent RL has been successfully applied in areas such as game play [18], resource management [19] and robotics [20].

When it comes to multi-agent reinforcement learning, formulations differ for different application scenarios, including: whether agents can communicate with team members, whether agents can perceive the complete global state, whether agents are decentralized and whether the agents are cooperative or competitive. MARL problems are generally quite challenging and remain as an active research area in machine learning and robotics communities.

The most relevant MARL scenario to MIPP is the case of multi-agent fully cooperative learning, where all the agents perceive the same global states and reward signals. Agents learn to optimize long-term global reward cooperatively. This type of multi-agent system (MAS) is formulated as a multi-agent Markov Decision Process [7] or a fully cooperative stochastic game [21]. One key challenge of MARL is that the size of joint action space increases exponentially with the number of agents. To reduce the search space and make learning and decision making scalable with respect to the number of agents, one solution is to use independent learners [22], where each learner learns based on individual reward signals.

The term "independent learner" though commonly used in MARL literature is a bit of misnomer. Learners can in fact share observed states and their reward signals are coupled. To ensure their cooperation, one important question is how to split team rewards to individual agents. This is known as the *multi-agent credit assignment* problem [23]. It is non-trivial to find an optimal credit assignment strategy. Intuitively, if the reward an agent receives is not commensurable to its efforts (e.g., costs), the agent may be discouraged from cooperation, or may become a freeloader. In [24], it was shown for multi-agent repeated games, equal reward split can lead to convergence to a Nash equilibrium of the game if a suitable exploitative exploration strategy is adopted. Other works show that empirically difference-rewards based credit assignment (i.e., the reward to an individual agent is calculated as the difference between the team reward from the joint action and that by removing the agent's action) outperforms equal reward split.

To the best of our knowledge, this is the first work that considers MARL for IPP and investigates the learning dynamics of different credit assignment schemes.

III. Problem Formulation

We define the MIPP problem with a five-tuple $\langle G, \mathcal{C}, \mathbf{v}_s, \mathbf{b}, f_{\mathcal{D}}(\mathcal{P})\rangle$. Specifically,

- $G = (\mathcal{V}, \mathcal{E})$ is a graph that representing the layout and reachability of the target area, where $\mathcal{V}$ is the set of points of interests (POIs), and $\mathcal{E}$ represents the connectivity between the POIs.
- $\mathcal{C} \subseteq \mathcal{V}$ denotes the set of charging locations or depots. We assume that each charging location or depot has sufficient capacity to serve all robots in the system. Depots with limited capacity will be considered in our future work.
- $\mathbf{v}_s$ is a vector that denotes the initial locations of the team of robots in G. The start location of robot i satisfies $v_s^i \in \mathcal{C}$, i.e, it is initially located at one of the depots. Let $N = |\mathbf{v}_s|$ be the number of robots.
- $\mathbf{b}$ is a vector of dimension N denoting the available travel budget for each robot.
- $f_{\mathcal{D}}(\mathcal{P})$ is a function to evaluate the utility of a path set $\mathcal{P} = \cup_{i=1}^{n} \mathcal{P}_i$ traversed by the team of robots. Here, the subscript $\mathcal{D}$ represents historically collected sensing data, such as pilot data used to estimate the hyper-parameters of the GP. Formally, we define

$$\begin{aligned} f_{\mathcal{D}}(\mathcal{P}) &= \mathrm{MI}(\mathbf{y}_{\mathcal{V}}; \mathbf{y}_{\mathcal{P}} \cup \mathbf{y}_{\mathcal{D}}) \\ &= H(\mathbf{y}_{\mathcal{V}}) - H(\mathbf{y}_{\mathcal{V}} \mid \mathbf{y}_{\mathcal{P}} \cup \mathbf{y}_{\mathcal{D}}) \end{aligned} \tag{1}$$

  Here MI means mutual information and H denotes differential entropy. $\mathbf{y}$ represents sensor observations. Subscripts $\mathcal{P}$, $\mathcal{V}$ and $\mathcal{D}$ of $\mathbf{y}$ indicate the locations where measurements are taken. For example, $\mathbf{y}_{\mathcal{P}}$ is the collected data along the path set $\mathcal{P}$ from all robots[1]. (1) measures the decreased uncertainty of $\mathbf{y}_{\mathcal{V}}$ due to observations $\mathbf{y}_{\mathcal{P}}$.

Under GP assumptions, $f_{\mathcal{D}}(\mathcal{P})$ can be calculated analytically as

$$f_{\mathcal{D}}(\mathcal{P}) = \frac{1}{2}\ln|\Sigma_{\mathbf{y}_{\mathcal{V}}}| - \frac{1}{2}\ln|\Sigma'_{\mathbf{y}_{\mathcal{V}}|\mathbf{y}_{\mathcal{P}}\cup\mathbf{y}_{\mathcal{D}}}|, \tag{2}$$

where Σ is the covariance matrix of $\mathbf{y}_{\mathcal{V}}$, and Σ' denotes the conditional covariance matrix given the relevant data, which only depends on the kernel of the GP (estimated by the pilot data) and sensing locations. Note that the actual sensor measurements do not affect the utility. More details on the utility function based on mutual information can be found in [14], [15]. Hyper-parameters of the underlying GP can be estimated with data from a pilot study or from previously collected trials $\mathcal{D}$.

Therefore, the MIPP problem can be formulated as:

$$\begin{aligned} &\text{maximize } f_{\mathcal{D}}(\mathcal{P}), \\ &\text{s.t. } C(\mathcal{P}_i) \le b_i \text{ and } \mathcal{P}_i[-1] \in \mathcal{C}, \\ &\forall i \in \{1, ..., N\}, \end{aligned} \tag{3}$$

where $C(\mathcal{P}_i)$ is the cost of path $\mathcal{P}_i$, and $\mathcal{P}_i[-1]$ is the last vertex of the path. In other words, we seek to plan paths for the robots for mobile sensing such that the uncertainty of the target area (on vertices $\mathcal{V}$) is minimized. For each robot, it needs to return to one of the depots within its budget constraint. The initial locations and budgets of the robots may be the same or different. Clearly, MIPP is NP-hard since it is equivalent to IPP with identical start and terminal locations when $N = 1$ and $|\mathcal{C}| = 1$.

Fig. 1 shows a toy example of MIPP with 2 robots. In this example, each robot has a budget of one unit. The costs of all edges are one. In this case, MIPP can be viewed as a 2-agent matrix game. By varying the start locations or the locations of the pilot data (indicated by the green star), the game may have one or multiple equilibria, which can be found manually. Even for such a simple scenario, designing an MARL strategy

[1]Robots can collect data from both vertices and edges they traverse.

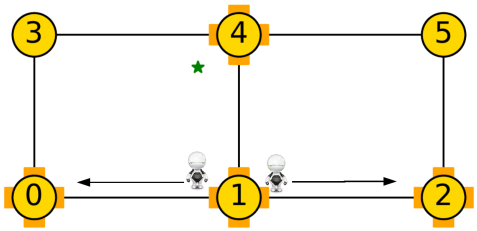


Fig. 1. A simple 1-step MIPP example. The green star sign represents the location that has been previously sensed, the orange bold plus shape denotes charging stations. Suppose each edge has unit length (1), if $\mathbf{v}_s = [1, 1]$ (vertex index) and $\mathbf{b} = [1, 1]$ (budgets), then the optimal paths for the two robots are $\mathcal{P}_1 = [1, 0]$ and $\mathcal{P}_2 = [1, 2]$, or $\mathcal{P}_1 = [1, 2]$ and $\mathcal{P}_2 = [1, 0]$ due to the existence of previously sensed locations. On the other hand, if $\mathbf{v}_s = [0, 2]$, the only solution is $\mathcal{P} = [[0, 1], [2, 1]]$ due to the budget and charging station constraints.

that converges to the optimal equilibrium is non-trivial for a repeated version of the game. As the number of the robots and budgets increase, for larger graphs, the search space becomes prohibitively vast for any manual solution to be feasible.

IV. RL Strategies to MIPP

Due to the NP-hardness of MIPP, one needs to devise heuristic solutions that can be computed in polynomial time. Conventional heuristic approaches such as GA require re-execution of their procedures whenever any parameter in $\langle G, \mathcal{C}, \mathbf{v}_s, \mathbf{b}, f_{\mathcal{D}}(\mathcal{P})\rangle$ changes since the corresponding problem instance is different. The advantage of an RL-based solution, on the other hand, lies in its ability to generalize to problem instances with different parameter settings with a single round of training.

Next, we first formalize MIPP from the prospective of Multi-agent Markov Decision Processes (MMDPs), and then discuss the proposed state encoding and reward function. Lastly, we present three different types of RL policies.

A. MMDP for MIPP

A multi-agent Markov Decision Process [7] is defined using a five-tuple $\langle S, \alpha, \{\mathcal{A}_i\}_{i\in\alpha}, Pr, R\rangle$, where

- S and α are finite sets of states and agents, and the states can be perceived by all agents,
- $\mathcal{A}_i$ is the set of actions available to agent i,
- $Pr : S \times \mathcal{A}_1 \times \cdots \mathcal{A}_N \to [0, 1]$ is the state transition probability,
- $R : S \times \mathcal{A}_1 \times \cdots \mathcal{A}_N \to \mathbb{R}$ is a reward function.

At any stage, each agent selects an action and executes it. The actions from the N agents form a joint action. Accordingly, the environment makes a state transition and provides a reward signal. An MMDP is also known as a stochastic game with identical interests (or a fully cooperative game) in game theory [21], where each player tries to maximize the same total pay-off.

In MIPP, each agent represents a data collection robot. Initially, agent i is at its start location v_s^i with an initial budget b_i. The available actions are the neighborhood vertices which agents can move to given remaining budgets. Execution of the joint action results in a single team reward. However, unlike the definition in [7], in MIPP, the state transitions are deterministic, i.e, given the current state and a joint action, the next state is unique. The goal of the agents is to plan paths that maximizing cumulative discounted team rewards.

B. States and Action Selection

We denote the state transition tuple at step t as $\langle s_t, \mathbf{u}_t, R_t, s_{t+1}\rangle$, where $\mathbf{u}_t = [u_t^1, ..., u_t^N]$ is the joint action, and R_t is the team reward. The state of robot i is encoded as (x_t^i, y_t^i, b_t^i), which consists of its coordinates in the target area and its remaining budget. Thus, the state of all agents at time t is given by $s_t = \{(x_t^i, y_t^i, b_t^i)_{i=1}^N\}$. The remaining budget b_t^i is updated as

$$b_t^i = b_{t-1}^i - c(v_{t-1}^i, v_t^i), \tag{4}$$

where $c(v_{t-1}^i, v_t^i)$ is the cost incurred by traversing the previous edge (v_{t-1}^i, v_t^i), and $b_0^i = b_i$ is the initial budget. Here, $v_t^i = (x_t^i, y_t^i)$ is the nodal position of agent i as the result of action u_t^i. Since historical actions (past paths) affect the current reward, we adopt a Recurrent Neural Network based on GRU [25] to encode the historical information as hidden states.

The available actions at t to agent i are the neighboring vertices of its current location. Since agents have different budgets, some may terminate earlier than others. To handle this situation, we include a dummy action to the action space, which is available when an agent arrives at a charging station and has insufficient budget to visit other locations. When all the agents arrive at charging stations, one training episode terminates. Furthermore, since an agent must reach one of the charging stations, we further prune the available actions of an agent as follows. Let the current position of agent i be v_t^i. With respect charging station $c_j \in \mathcal{C}$, its valid actions are given by

$$\mathcal{A}_t^i(v_t^i, c_j) = \{v \in \text{Nbr}(v_t^i) : c(v_t^i, v) + \text{LCP}(v, c_j) \le b_t^i\}, \tag{5}$$

where Nbr represents adjacent vertices, and LCP denotes the Least Cost Path from v to v_t that can be computed using the Dijkstra algorithm. Thus, the valid action set for agent i at v_t^i is,

$$\mathcal{A}_t^i(v_t^i) = \bigcup_{j=1}^{|\mathcal{C}|} \mathcal{A}_t^i(v_t^i, c_j). \tag{6}$$

In implementation, we define a mask vector based on (6) and add it to the Q-values to remove invalid actions, so that agents are guaranteed to reach some charging stations within budget constraints.

C. Team Reward

The team reward R_t is calculated as the reduction in the uncertainty of $\mathbf{y}_{\mathcal{V}}$ due to the incremental data collected by

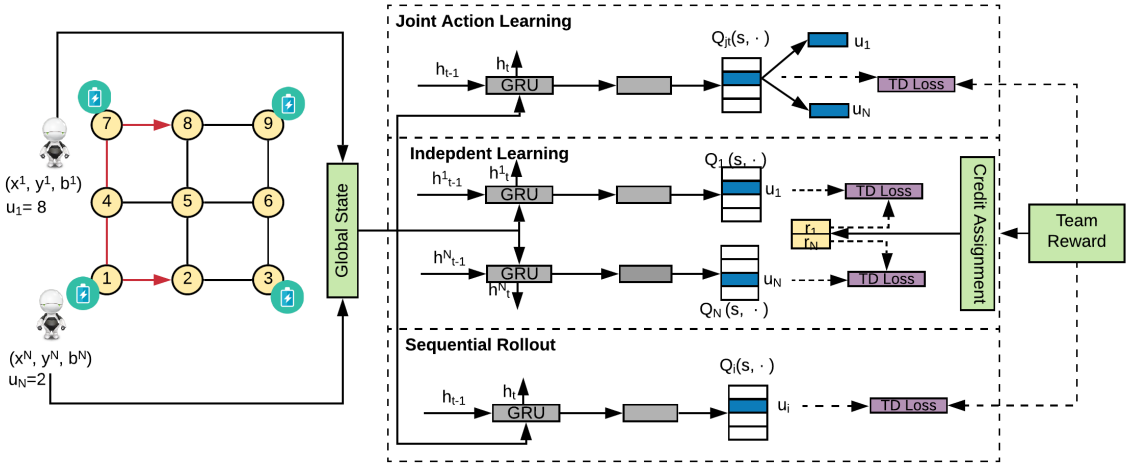


Fig. 2. State encoding and three learning schemes

all robots at step t. Recall from Section III, uncertainty only depends on sensing locations under the GP assumption, which can be determined from the sampling frequency and the moving speed of robots. Formally,

$$R_t = H(\mathbf{y}_{\mathcal{V}} \mid \mathbf{y}_{\mathcal{P}_{t-1}} \cup \mathbf{y}_{\mathcal{D}}) - H(\mathbf{y}_{\mathcal{V}} \mid \mathbf{y}_{\mathcal{P}_t} \cup \mathbf{y}_{\mathcal{D}}), \quad (7)$$

where $\mathbf{y}_{\mathcal{P}_{t-1}}$ and $\mathbf{y}_{\mathcal{P}_t}$ are the sensor data collected by the robot team through the previous and current steps, respectively. Clearly, $\sum_t R_t = f_{\mathcal{D}}(\mathcal{P})$ is exactly the optimization goal defined in (3). As a result, the team of robots are incentivized to explore informative paths through the reward signal.

D. Learning Schemes

To make the learned policy generalizable to different start locations and initial budgets for all agents, during the training stage, we treat $\mathbf{v}_s$ and $\mathbf{b}$ as variables. For each training episode, we randomly sample the start location of each robot from the set $\mathcal{C}$. Similarly, the budget of each robot is sampled uniformly from $[B_{\min}, B_{\max}]$. Next, we present three types of training schemes based on Q-learning, each representative of a mechanism that agents cooperate (Fig. 2).

1) Joint Action Learning: Joint Action Learning (JAL) treats the MMDP as a single agent MDP, and encodes a joint action as a single action. For instance, in a two-agent system, a pair of action (u_1, u_2) can be encoded as a joint action index $u_1 \cdot (1 + |\mathcal{V}|) + u_2$ (note the existence of the dummy action). Conversely, any joint action index can be mapped to a pair of actions. The reward signal can be thus seen as the reward to the encoded single action. By doing so, existing single-agent Q-learning algorithm like DQN [18] can be applied. We call this approach DQN-JAL. The main disadvantage of JAL is that it can only handle small problem sizes with limited number of agents and a small action set.

2) Independent Q-learning: One way to reduce the action space of the MMDP is by assigning credits to agents based on their respective actions and the team reward in each step, and train the agents separately. Formally, a credit assignment strategy is defined as a function $\eta : S \times A \times \mathbb{R} \to \mathbb{R}^N$ that maps $\{s_t, \mathbf{u}_t, R_t\}$ to $[r_t^1, ..., r_t^N]$. We consider two different forms of $\eta(s_t, \mathbf{u}_t, R_t)$ for MIPP. The first approach is to split reward R_t equally among robots regardless of their actions with the exception of robots who have terminated, which will receive zero awards. The second approach is based on difference rewards [10], [23], which considers the marginal contribution of an agent. In particular, r_t^i can be calculated as

$$r_t^i = H(\mathbf{y}_{\mathcal{V}} \mid \mathbf{y}_{\mathcal{P}_t^{-i} \cup \mathcal{D}}) - H(\mathbf{y}_{\mathcal{V}} \mid \mathbf{y}_{\cup_{j=1}^N \mathcal{P}_t^j \cup \mathcal{D}}), \quad (8)$$

where $\mathcal{P}_t^{-i}$ denotes the set of paths from teammates, excluding the path from agent i. Since $\sum_{i=1}^N r_t^i \neq R_t$, we further normalize r_t^i with $\frac{R_t}{\sum_{i=1}^N r_t^i}$ to obtain the final reward.

Based on the outcome of credit assignment, the training procedure of independent learning is outlined by Algorithm 1. We adopt Q-learning as the RL algorithm for good sample efficiency. The optimization goal is to minimize the temporal-difference (TD) error of the Q-function network,

$$\mathcal{L}_{\text{TD}}(s_t, u_t^i, r_t^i, s_{t+1}; \boldsymbol{\theta}_i) = (Q_i(s_t, u_t^i; \boldsymbol{\theta}_i) - y_{\text{target}}^i)^2, \quad (9)$$

where $y_{\text{target}}^i = r_t^i + \gamma \max_{u_{t+1}^i} Q_i(s_{t+1}, u_{t+1}^i; \boldsymbol{\theta}_i^-)$. For every C episodes, we save the model parameters $\boldsymbol{\theta}$ (the set of parameters for all agents). We call this approach IQL-EQUALSPLIT or IQL-DIFFREWARDS depending on the credit assignment approach used.

3) Sequential Rollout: In MIPP, intuitively, agents can maximize team reward by spatially spreading out in the target area. Their actions can be performed in an asynchronous manner. For example, given two paths, collecting data using two robots concurrently achieves the same utility as running one robot after the other sequentially along the paths since utility only depends on the locations of data collection (though the completion time is different). However, unlike sequential allocation [26] that treat robots independently, when planning the path of the first robot, one should be cognizant of the initial

Algorithm 1: IQL based on Credit Assignment

Input : $\langle G, \mathcal{C}, f_{\mathcal{D}}(\mathcal{P}), B_{\text{min}}, B_{\text{max}} \rangle$, model save cycle C
Output: model parameter set Θ

initialize replay memory $\mathcal{M}$
initialize $\boldsymbol{\theta}$ randomly and set $\boldsymbol{\theta}^- = \boldsymbol{\theta}$
for episode $e = 1, 2, ...$ **do**
  set $\mathbf{v}_s$ and $\mathbf{b}$ by sampling from $\mathcal{C}$ and $[B_{\text{min}}, B_{\text{max}}]$
  get initial global state s_0
  for step $t = 1, 2, ..., T$ **do**
    for agent $i = 1, 2, ..., N$ **do**
      with probability ϵ select a random action u_t^i from $\mathcal{A}_t^i(v_t^i)$
      otherwise select $u_t^i = \arg\max_{u_t^i} Q_i(s, u_t^i)$
    end
    take action $\mathbf{u}_t$ and calculate R_t
    store transition $(s_t, \mathbf{u}_t, R_t, s_{t+1})$ to $\mathcal{M}$
    sample a batch of $(s_j, \mathbf{u}_j, R_j, s_{j+1})$ from $\mathcal{M}$
    calculate the credit assignment $[r_j^1, ..., r_j^N]$
    update $\boldsymbol{\theta}_i$ by minimizing the loss $\mathcal{L}_{\text{TD}}(s_j, u_j^i, r_j^i, s_{j+1}; \boldsymbol{\theta}_i)$ for each agent
  end
  set $\boldsymbol{\theta}^- = \boldsymbol{\theta}$ with some period K
  if $e \bmod C = 0$ **then**
    save parameter $\boldsymbol{\theta}$ to Θ
end
return Θ

locations and budgets of the remaining robots. Motivated by this key insight, we propose a new cooperative mechanism called *sequential rollout* by converting the MMDP to a single-agent decision process. We fix the order of the robots. At each time step t, the agent makes decision for one robot, and all the other robots take the dummy action, i.e, they stay at the same location. When the current robot arrives at a charging station and is out of budget, the agent switches to the next robot on the list.

In sequential rollout, the full state includes the status of all robots to predict the total discounted future reward. However, the one-step reward only comes from the active robot and its action. The main training procedure is similar to Alg. 1 except that there is only a single agent, and no credit assignment is needed. This approach also does not suffer from the problem of exponential joint action space. One potential disadvantage is that the time step horizon is longer from T steps to NT steps. In other words, it trades spatial with time. This may make the prediction of the total future reward less accurate compared with a shorter horizon. We call this approach SEQ-ROLLOUT-DQN with DQN as the learning algorithm.

E. Path Planning

Once the model has been trained, it can be utilized for path planning. Given any $\mathbf{v}_s$ and $\mathbf{b}$, a greedy policy with respect to the Q-values can be used to select actions. Note that except for JAL-DQN, the next action of a robot only takes $O(|\mathcal{V}|)$ time.

To mitigate local optimality, we execute the path planning algorithm multiple times with different model parameters output by Algorithm 2, and select the path set with the maximum reward. The path planning procedure is outlined in Algorithm 2 for independent learning. For sequential rollout, paths are generated one by one sequentially.

Algorithm 2: Path Planning Procedure (with IQL)

Input : $\Theta, \mathbf{v}_s, \mathbf{b}$
Output: best path $\mathcal{P}^*$

initialize a candidate path set $\mathcal{P}_\Theta = \{\}$
for $\boldsymbol{\theta} \in \Theta$ **do**
  get initial global state s_0
  for step $t = 1, 2, ..., T$ **do**
    for agent $i = 1, 2, ..., N$ **do**
      select $u_t^i = \arg\max_{u_t^i} Q_i(s, u_t^i)$ based on $\boldsymbol{\theta}$
    end
    take action $\mathbf{u}_t$
  end
  add the generated path to $\mathcal{P}$
end
return $\mathcal{P}^* = \arg\max_{\mathcal{P} \in \mathcal{P}_\Theta} f_{\mathcal{D}}(\mathcal{P})$

V. PERFORMANCE EVALUATION

In this section, we compare the performance of different strategies for MIPP using real-world traces. For evaluation, we consider the task of Wi-Fi RSS collection. RSS is widely used in indoor localization solutions due to its pervasiveness and ease of measurements [9], [27]. However, it is notoriously time consuming to collect RSS measurements in large indoor areas. Mobile robotic sensing offers a promising alternative to manual site survey. Given limited time budgets, collecting RSS along informative paths can lead to a better estimation of the RSS distribution of a target area, which is instrumental to reducing localization errors [5].

A. Implementation and Environment Setup

Two indoor areas, a squared-shaped one and a T-shaped one, are selected for experiments. In the two areas, there are some previously collected RSS data used to estimate the hyper-parameters of GPs. The layouts, entropy heatmaps and the constructed graphs of the two areas are shown in Fig. 3 and Fig. 4, respectively.

To train RL agents, a simulator is developed in Python and used as the environment. Graphs are represented using Networkx [28]. The simulator is able to evaluate the total reward of any path on a graph. Both the moving speed of robots and sensing frequency can be configured, which may lead to different sensing locations along a path.

During training, the neural networks in DQN are implemented using PyTorch [29]. RMSProp [30] is utilized as the optimizer for parameter estimation with a learning rate of

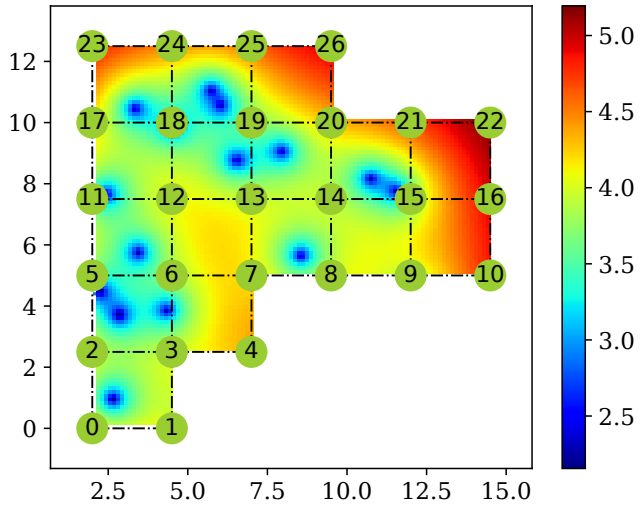


Fig. 3. Entropy heatmap in Area One and the corresponding graph. Large values indicate higher uncertainty.

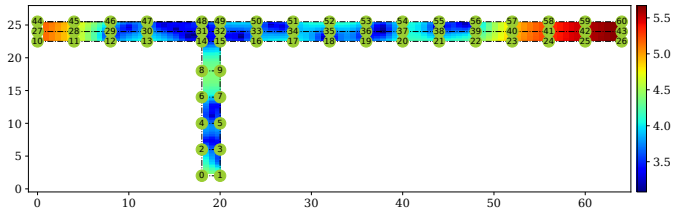


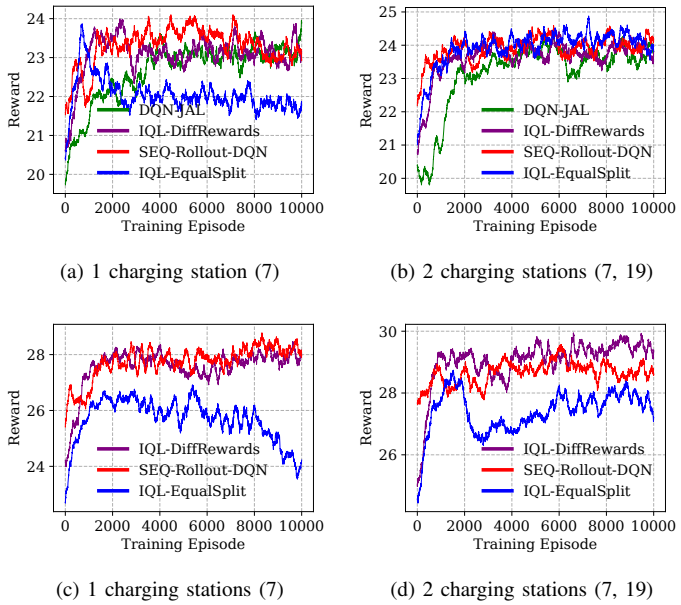
Fig. 4. Entropy heatmap in Area Two and the corresponding graph. Large values indicate higher uncertainty.

0.0001 for all the learning schemes. The minibatch size is set to 32 for each training iteration. The size of experience buffer is 5000. The discounter factor γ for long-term returns is set to 0.99. Different settings are considered, including different numbers of robots, charging stations and budgets. In each setting, we train 10000 episodes on the graph of Area One, and 20000 episodes on the graph of Area Two.

B. Training and Convergence

Extensive experiments are carried out in the two areas to evaluate different learning schemes. In the first area, policies are trained for the cases of 2 or 3 robots. In each case, 1 or 2 charging stations are included. In the second area, due to its elongated shape, more robots (3 – 4), and charging stations (3 – 4) are deployed. Robots in the first and the second areas have budget ranging from 15 to 30 units and 30 to 50 units, respectively. The cost of a single edge in the first and the second areas are 2.5 units and 2 units, respectively. During training, budgets are sampled from these ranges for all robots as shown in Alg. 1.

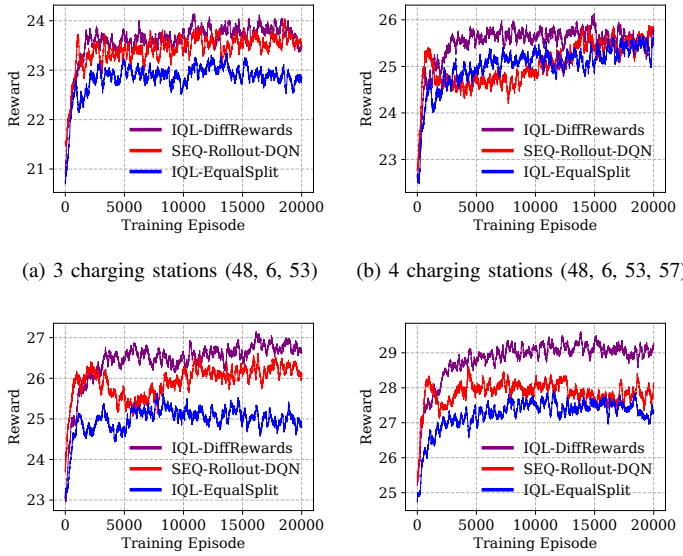
Fig. 5 shows the average reward during training per episode for robots in Area One. When there are only 2 robots, we also evaluated DQN-JAL, which has a joint action space of size $28 \times 28 = 784$. Due to its poor scalability, as the graph size



(a) 1 charging station (7) (b) 2 charging stations (7, 19)

(c) 1 charging stations (7) (d) 2 charging stations (7, 19)

Fig. 5. Average reward per episode during training for Area One, with different setting of charging stations and number of robots. (a) and (b): **2** robots, (c) and (d): **3** robots.



(a) 3 charging stations (48, 6, 53) (b) 4 charging stations (48, 6, 53, 57)

(c) 3 charging stations (48, 6, 53) (d) 4 charging stations (48, 6, 53, 57)

Fig. 6. Average reward per episode during training for Area Two. (a) and (b): **3** robots, (c) and (d): **4** robots.

grows (in Area Two) or the number of robots increases, it is no longer computationally feasible to run DQN-JAL. From the figure, we see that in the first area, IQL-DIFFREWARDS and SEQ-ROLLOUT-DQN have a comparable performance, and are both better than IQL-EQUALSPLIT. On the other hand, DQN-JAL converges slowly and reaches similar rewards as IQL-DIFFREWARDS and SEQ-ROLLOUT-DQN after 10,000 episodes of training.

Fig. 6 shows the rewards during the training process with 3 and 4 robots in Area Two. Similar to the first Area, it can be observed that IQL-EQUALSPLIT has inferior performance overall, and IQL-DIFFREWARDS achieves the best result among all three schemes. Moreover, the performance of SEQ-ROLLOUT-DQN degrades as more robots are available. As discussion previously in Section IV, sequential rollout has a longer time horizon since the total number of steps increases linearly with the total budget of the robots. In contrast, with independent learning, all robots are trained concurrently.

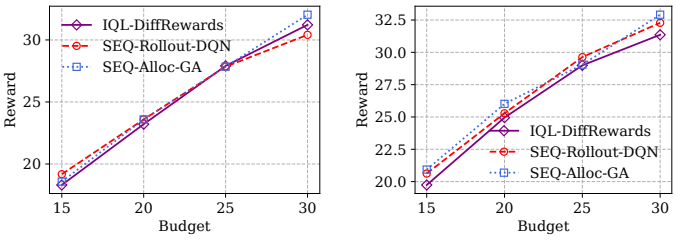
Finally, the search space grows when the number of robots or the number of charging stations increases. For instance, when there is only one charging station, the paths of all the robots form a tour. Adding an extra charging station means that robots have more available paths to explore since they do not have to return to where they started. In both two areas, rewards at convergence time increase with more robots or charging stations as expected.

C. Path Planning Performance

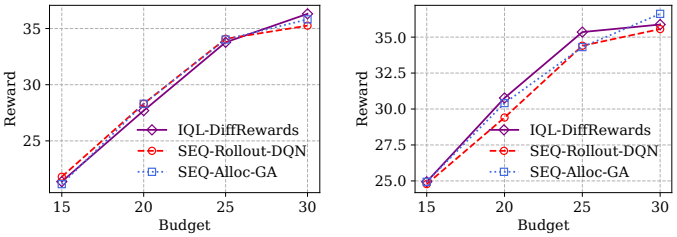
The learned policies encode knowledge of informative paths and can be used to plan paths with different initial budgets and starting locations. To evaluate the informativeness of the predicted paths of the proposed schemes, we also implement a non-learning sequential allocation method inspired by [15], which plans paths one after another independently using a single-robot IPP algorithm. We adopt the GA from [5] as the single-robot IPP algorithm. In GA, paths are encoded as chromosomes, cross over and mutation operators are designed to increase the diversity of the population. The best path among the final evolution generation is chosen as the output. Due to the existence of multiple charging stations, the GA in [5] needs to be adapted to handle situations that robots can terminate at *any* of the charging stations. We refer to this approach as SEQ-ALLOC-GA in subsequent comparisons. In experiments, the population size is set to 100 and 50 generations are evolved. Note that SEQ-ALLOC-GA needs to be re-executed any time the budgets of robots or their initial positions change.

For RL based solutions, we select policies trained by IQL-DIFFREWARDS and SEQ-ROLLOUT-DQN in path planning, since they have fast and better convergence as seen in the previous section. Next, we present the results of two scenarios, namely, homogeneous budgets and heterogeneous budgets among the robots. In the first scenario, every robot has the same budget, while in the second case, at least one robot has a different budget from the others.

1) Homogeneous Budgets: Fig. 7 shows the rewards of different path planning schemes in Area One. In the experiments, we vary the budgets, the number of robots and charging stations. It can be seen that the RL-based path planning methods achieve similar rewards as SEQ-ALLOC-GA. As the budgets increase or the number of robots increases, high reward can be achieved by all schemes. Introduction of more charging stations also improves the rewards as more feasible paths are available to all robots.

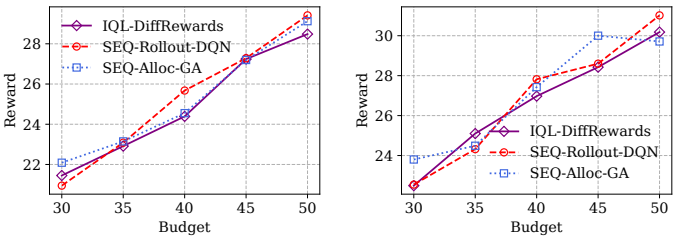


(a) 1 charging station (7) (b) 2 charging stations (7, 19)

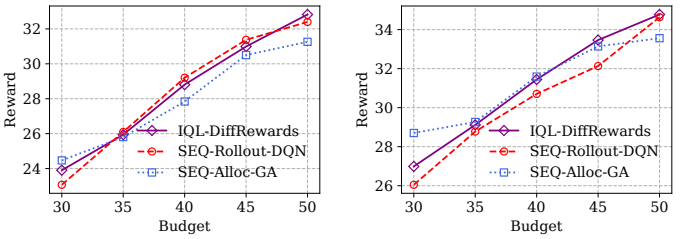


(c) 1 charging station (7) (d) 2 charging stations (7, 19)

Fig. 7. Path planning performance in Area One for homogeneous robots. The X-axis shows the budget for each robot for a single experiment. (a) and (b): **2** robot team, (c) and (d): **3** robot team. In (b), the initial locations $\mathbf{v}_s = [7, 19]$, in (d), $\mathbf{v}_s = [7, 7, 19]$.



(a) 3 charging stations (48, 6, 53) (b) 4 charging stations (48, 6, 53, 57)

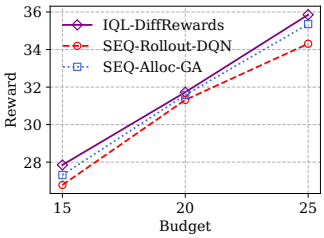


(c) 3 charging stations (48, 6, 53) (d) 4 charging stations (48, 6, 53, 57)

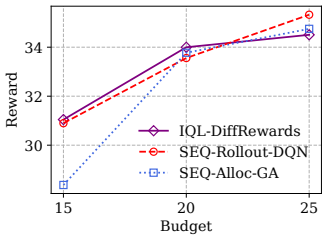
Fig. 8. Path planning performance in Area Two for homogeneous robots. (a) and (b): **3** robot team, (c) and (d): **4** robot team. In both (a) and (b), the initial locations $\mathbf{v}_s = [48, 6, 53]$, in (c), $\mathbf{v}_s = [48, 6, 53, 48]$, in (d), $\mathbf{v}_s = [48, 6, 53, 57]$.

Similar observations can be made for Area Two as shown in Fig. 8 for 3 and 4 robots. All methods have competitive performance.

2) Heterogeneous Budgets: Fig. 9 and Fig. 10 show the results of robots with heterogeneous budgets in Area One and Area Two, respectively. In this scenario, SEQ-ALLOC-

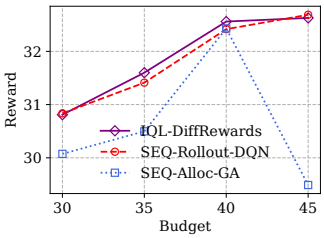


(a) 1 charging station (7) nodes

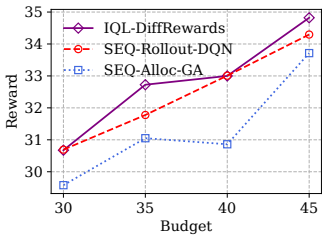


(b) 2 charging stations (7, 19)

Fig. 9. Path planning performance in Area One for **3** heterogeneous robots. One robot has a fixed budget $b = 30$, and X-axis shows budgets for the other two robots. In (b), the initial locations $\mathbf{v}_s = [7, 19, 7]$.



(a) 3 charging stations (48, 6, 53)



(b) 4 charging stations (48, 6, 53, 57)

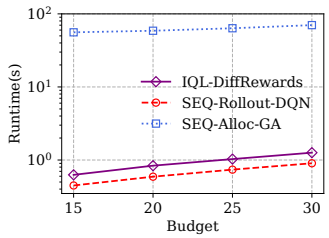
Fig. 10. Path planning performance in Area Two for **4** heterogeneous robots. Two robots have a fixed budget of 50, and X-axis shows the budgets of the remaining two robots. In (a), the initial locations $\mathbf{v}_s = [48, 6, 53, 48]$ and (b), the initial locations $\mathbf{v}_s = [48, 6, 53, 57]$.

GA has inferior performance than the RL-based solutions in most cases. This is because SEQ-ALLOC-GA is a greedy solution. It can only make use of the current robot's budget information, and plan the robots independently. When the robots have different budgets, the order which robot is planned first matters. To understand this, consider the case of two robots, one with a large budget and one with a small budget. To maximize the total reward, it is desirable to plan the robot with the smaller budget first as the other robot has more flexibility to maneuver to regions with higher rewards given the decision of the first robot. The RL-based schemes, by design, take into account of the (future) decisions of all robots in updating Q-functions and thus are agnostic to the orders.

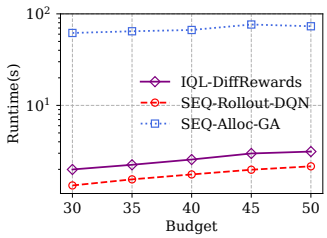
Lastly, it can be observed from Fig. 9 and Fig. 10 that with the exception of SEQ-ALLOC-GA, all schemes achieve higher rewards with more average budgets and more charging stations in both areas. IQL-DIFFREWARDS has the best performance in most cases.

D. Computation Efficiency

In addition to the total rewards, an important evaluation metrics is the computation time. Without runtime limitation, one can even adopt brute force search to find paths with the highest rewards. For GA, one can infinitely increase the population size or the number of evolution generations to approach global optimal.



(a) Area One, 2 robots



(b) Area Two, 3 robots

Fig. 11. Approximate run time of different solutions for the two areas on a desktop (16GB memory, Intel Core i7).

For RL-based MIPP schemes, we focus on the computation time of path planning (or inference), since the policies are trained beforehand without the knowledge of specific budgets or initial locations of robots. The computation complexity of IQL-DIFFREWARDS and SEQ-ROLLOUT-DQN is both $O(N|\mathcal{V}|B_{\max}|\Theta|)$, where $B_{\max}$ is the maximum budget of the robots and $|\Theta|$ is the size of the model parameter sets. The compexity of JAL-DQN is $O(|\mathcal{V}|^N B_{\max}|\Theta|)$.

Fig. 11 shows the path planning run time of different approaches on a desktop PC with Intel Core i7 and 16GB memory for the two target areas. It can be seen that both RL-based path planning solutions are efficient, and can finish within seconds even in the worst case. Since in each time step, IQL-DIFFREWARDS needs to compute the action of each agent, it is slightly slower than SEQ-ROLLOUT-DQN. We error on the optimistic side to run GA for only 50 generations though it is far from reaching the optimal solution. Even so, SEQ-ALLOC-GA takes much longer time than the RL-based solutions.

VI. CONCLUSION AND FUTURE WORK

In this paper, we formulated cooperative spatial sensing among multiple robots as the MIPP problem and developed efficient RL-based solutions. To mitigate the exponential growth of the joint action space with more robots, we devised two solutions, namely, individual learning based on credit assignment and sequential rollout based reinforcement learning. With both schemes, the learned policy is utilized to plan paths and achieves higher or similar rewards compared with baseline solutions in most cases. More importantly, the RL-based MIPP solutions are much more efficient since they can handle different parameter settings during inference.

As part of our future work, we plan to analyze the learning dynamics of different multi-agent learning schemes, and develop policies that work with actions in continuous space.

REFERENCES

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[2] G. Cook and F. Zhang, *Mobile Robots: Navigation, Control and Sensing, Surface Robots and AUVs*. John Wiley & Sons, 2020.

[3] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[4] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[5] Y. Wei, C. Frincu, and R. Zheng, "Informative path planning for location fingerprint collection," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1633–1644, 2020.

[6] K.-C. Ma, L. Liu, and G. S. Sukhatme, "Informative planning and online learning with sparse gaussian processes," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4292–4298.

[7] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, 1996, pp. 195–210.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[9] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019.

[10] D. H. Wolpert and K. Tumer, "Optimal payoff functions for members of collectives," in *Modeling complexity in economic and social systems*. World Scientific, 2002, pp. 355–369.

[11] D. Di Paola, A. Milella, G. Cicirelli, and A. Distante, "An autonomous mobile robotic system for surveillance of indoor environments," *International Journal of Advanced Robotic Systems*, vol. 7, no. 1, p. 8, 2010.

[12] C. Li, C. Frincu, Z. Gong, Q. Xu, and R. Zheng, "Robot-assisted fingerprint-based indoor localization," *Microsoft Indoor Localization Competition, Tech. Rep.*, 2017.

[13] H. Ishida, Y. Wada, and H. Matsukura, "Chemical sensing in robotic applications: A review," *IEEE Sensors Journal*, vol. 12, no. 11, pp. 3163–3173, 2012.

[14] Y. Wei and R. Zheng, "Informative path planning for mobile sensing with reinforcement learning," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 864–873.

[15] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[16] N. Cao, K. H. Low, and J. M. Dolan, "Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 7–14.

[17] A. Dutta, A. Bhattacharya, O. P. Kreidl, A. Ghosh, and P. Dasgupta, "Multi-robot informative path planning in unknown environments through continuous region partitioning," in *The Thirty-Second International Flairs Conference*, 2019.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[19] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for v2v communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[20] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[21] T. Raghaven, T. S. Ferguson, T. Parthasarathy, and O. Vrieze, *Stochastic games and related topics: In honor of professor LS Shapley*. Springer Science & Business Media, 2012, vol. 7.

[22] E. Wei and S. Luke, "Lenient learning in independent-learner stochastic cooperative games," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2914–2955, 2016.

[23] D. T. Nguyen, A. Kumar, and H. C. Lau, "Credit assignment for collective multiagent rl with global rewards," in *Advances in Neural Information Processing Systems*, 2018, pp. 8102–8113.

[24] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.

[25] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[26] A. Singh, A. Krause, C. Guestrin, W. J. Kaiser, and M. A. Batalin, "Efficient planning of informative paths for multiple robots." in *IJCAI*, vol. 7, 2007, pp. 2204–2211.

[27] C. Li, Q. Xu, Z. Gong, and R. Zheng, "Turf: Fast data collection for fingerprint-based indoor localization," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2017, pp. 1–8.

[28] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[30] Y. Bengio and M. CA, "Rmsprop and equilibrated adaptive learning rates for nonconvex optimization," *corr abs/1502.04390*, 2015.