In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS 2020), October 2020

Deep R-Learning for Continual Area Sweeping

Rishi Shah[*1,2], Yuqian Jiang[*1], Justin Hart[1], Peter Stone[1,3]
{rishihahs, jiangyuqian}@utexas.edu, {hart, pstone}@cs.utexas.edu

***Abstract*— Coverage path planning is a well-studied problem in robotics in which a robot must plan a path that passes through every point in a given area repeatedly, usually with a uniform frequency. To address the scenario in which some points need to be visited more frequently than others, this problem has been extended to non-uniform coverage planning. This paper considers the variant of non-uniform coverage in which the robot does not know the distribution of relevant events beforehand and must nevertheless learn to maximize the rate of detecting events of interest. This *continual area sweeping* problem has been previously formalized in a way that makes strong assumptions about the environment, and to date only a greedy approach has been proposed. We generalize the *continual area sweeping* formulation to include fewer environmental constraints, and propose a novel approach based on reinforcement learning in a Semi-Markov Decision Process. This approach is evaluated in an abstract simulation and in a high fidelity Gazebo simulation. These evaluations show significant improvement upon the existing approach in general settings, which is especially relevant in the growing area of service robotics. We also present a video demonstration on a real service robot.**

I. Introduction

Consider a service robot operating in an office or home. When a user requests that the robot bring a cold beverage or pick up the mail, the robot must reason about not only the static facts, such as the locations of rooms, but also the locations of objects in its environment which can change over time. As the occupants in this environment may move objects around, efficiently servicing these requests requires continually surveying the area to keep up to date on the objects' locations.

The problem of *continual area sweeping* was introduced by Ahmadi and Stone [1] as one motivated by building maintenance tasks in which some areas of the building see higher traffic and messier activities and therefore must receive more attention. A robot performing such tasks needs to service trash cans and restrooms more frequently than closets. For example, in a cleaning setting, the robot acts optimally when the time between the appearance of a mess and cleaning it up is minimal. Ahmadi and Stone [1] formalize a process of visiting areas of the map in a gridworld in which "events" (representing dirt and messes) appear non-uniformly throughout the robot's environment. Unlike more classical approaches, the distribution of these events is neither known nor constant, and thus must be learned online. Performance is measured based on how long it takes from the onset of an event to its servicing.

To model the task of a service robot surveying its environment for changes, this paper extends continual area sweeping. In the original continual area sweeping formulation, the objective is to minimize the time to detect events. This paper additionally considers the objective of maximizing the number of events detected per second (DPS). Assumptions about the distribution and appearance of events are also relaxed in order to better represent this scenario. We introduce the DPS-MAX approach to maximize detections per second. DPS-MAX combines a novel formulation based on a Semi-Markov Decision Process in the average reward setting, and then a deep reinforcement learning algorithm to solve it.

Evaluations of our DPS-MAX approach are presented in two simulation domains. An abstract gridworld is used to compare the performance of the Reinforcement Learning (RL) approach with the approach presented by Ahmadi and Stone [1], which serves as a baseline. Results show DPS-MAX significantly improves performance in the most general scenario, and more flexibly handles complex event patterns by leveraging extra environmental information. DPS-MAX, unlike the baseline, is also shown to recognize previously seen geometric features between different environments on a simulated service robot in Gazebo.

The primary contribution of this paper is the DPS-MAX approach which combines a novel Semi-Markov Decision Process problem formulation with a deep reinforcement learning algorithm to solve it. DPS-MAX addresses a general class of continual area sweeping problems, specifically those motivated by the growing area of service robots. Under the assumptions reflective of such scenarios, DPS-MAX significantly outperforms the prior state-of-the-art algorithm for continual area sweeping.

II. Continual Area Sweeping

In the continual area sweeping task, a robot continually travels in an environment with the goal of detecting or reacting to events of interest. The environment is represented as a 2D map which is divided into a set of discrete grid cells $g \in G$. A set of events $e \in E$ can occur anywhere in the environment at any time t, and the distribution of events is

* Equal contribution
[1] Department of Computer Science, The University of Texas at Austin
[2] Amazon (work done prior to joining Amazon)
[3] Sony AI

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, NRI-1925082), ONR (N00014-18-2243), FLI (RFP2-000), ARO (W911NF-19-2-0333), DARPA, Lockheed Martin, GM, and Bosch. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

unknown. The robot makes sequential decisions which are broken down into discrete decision steps $n \in \mathbb{N}$. At each decision step n, the robot can take an action a_n to move to any reachable grid cell g (including staying at the current cell). This action space focuses on the decision of where to visit, and assumes the shortest path will be taken. When an action is executed, the robot is able to detect any events in every grid cell along its path. The number of such detections is d_n. Note that the robot must physically travel from grid cell to grid cell, and as such may take a variable length of time to do so. Following this fact, the wall-clock time of decision step n is denoted as t_n, and the problem is modeled as a Semi-MDP (see Section IV-A).

A. Metrics

We define two metrics, *average detection time (ADT)* and *detections per second (DPS)*, each appropriate for a different class of applications.

Average detection time (ADT) is the average time elapsed from occurrence to detection of events. More formally, let $o(e)$ denote the time when event e occurs in the grid, and let $s(e)$ denote the time at which event e was seen. If e has never been detected, then let $s(e)$ be the current time. Then ADT is $\frac{1}{m}\sum_{i=1}^{m}(s(e_i) - o(e_i))$, where m is the total number of events that have occurred. This metric is used in the original continual area sweeping formulation [1]. If the goal of the robot is to be highly responsive to emergencies, such as spilled drinks for a maintenance robot, then it is appropriate to minimize average detection time.

Detections per second (DPS) is the average of the number of events detected per unit time, computed as $\frac{1}{t_n}\sum_{i=1}^{n} d_i$. If the goal of the robot is to maintain up-to-date information in its environment, then it should detect as many changes as possible over time. Thus, maximizing detections per second is more meaningful.

Both metrics are defined in the continual setting, so the most relevant observation is the long term average as m and n become arbitrarily large.

B. Assumptions

We assume that at each time step, the number of events in a grid cell $g \in G$ has an upper bound. Events can also stop after they occur. For example, in the object tracking task, if a water bottle is placed on a desk, and its owner later picks it up, the event of the disappearance of the object overwrites the event of its appearance. In realistic domains, the bound on number of events in each grid cell is usually close to 1 for a fine enough grid representation.

III. Related Work

Coverage path planning is a family of problems in which an agent is given a map of its environment and must generate a navigational path that covers its environment. The family spans three main categories, which we survey below.

A. Uniform Coverage

Uniform coverage, also known as sweeping, has an agent generate a navigational path such that the agent passes through the entire volume of the map [2], [3], [4]. This approach is useful for a variety of applications where the robot must travel over the entire area, such as lawn mowing or vacuum cleaning.

B. Adversarial Coverage

Patrolling is a related problem that operates in a similar setting. Gatti [5] describes a game theoretic approach based on adversarial guard and robber agents that act strategically. Much of the other work in this area is concerned with such an adversarial two player game scenario [6], [7]. This work focuses on non-adversarial settings motivated by service robot environments where events from tasks such as cleaning or semantic mapping do not involve opponents.

C. Non-uniform Coverage

Uniform coverage can be wasteful when areas of a map do not have equal importance. In such a setting, non-uniform coverage approaches optimize some metric by giving more focus to certain areas.

1) Coverage with Metrics: Ergodic coverage aims to optimize the ergodicity metric, in which time spent in a given area is related to the spatial distribution of regions of interest over that area [8]. Information surfing is also a non-uniform coverage approach but instead seeks to maximize discriminatory information by planning a path that exploits local information gradients [9]. Most non-uniform coverage approaches focus on planning with a given distribution of events.

In some applications such as surveillance, a robot may need to cover an area repeatedly with an unknown or changing distribution of events. More recent work relaxes the assumption of having a known events distribution in ergodic coverage. Mavrommati et. al. present an adaptive planning approach that works with a changing events distribution while still optimizing the ergodicity metric [10]. Continual area sweeping is a different non-uniform coverage problem where an area has to be covered repeatedly with an unknown and changing event distribution while optimizing the ADT or DPS metric. Unlike ergodic coverage which only cares about how much time is spent proportionally in each region, in continual area sweeping *when* each region is visited is also important.

2) ADT-GREEDY *Algorithm:* The closest work to this paper is by Ahmadi and Stone [1] who introduced the non-uniform continual area sweeping problem and proposed a greedy algorithm that minimizes the average detection time (ADT) while learning a changing distribution of events. For the remainder of this paper, this approach will be referred to as ADT-GREEDY.

ADT-GREEDY makes two assumptions that are revisited in the current work. The first is an assumption that at each time step there is a fixed probability p_g for an event occurring at grid cell g. It follows that the number of events

in each grid cell follows a binomial distribution $B(t, p_g)$, where t is the number of time steps since the cell was last visited. Henceforth, this will be referred to as the *binomial assumption*. The second assumption is that there is no upper bound on the number of events per cell, henceforth called the *unbounded events assumption*. A convenient consequence of these assumptions is that the expected number of events in a cell is linear in the time since the cell was last visited. This linearity exists because the expectation with respect to the Binomial distribution $B(t, p_g)$ is $t \cdot p_g$. Ahmadi and Stone [1] show that under these conditions maximizing the total expected number of detected events is the same as minimizing ADT. The ADT-GREEDY algorithm consists of a learning module that learns p_g for every grid cell g. A planning module then greedily chooses the target cell that leads to the path with the highest expected number of events.

This paper shows that ADT-GREEDY leads to suboptimal behavior when the assumptions are violated, which motivates using reinforcement learning to maximize continual area sweeping metrics without directly learning the event distribution.

IV. Approach

This section proposes a novel formulation of continual area sweeping as a Semi-Markov Decision Process which is then solved using a deep RL approach. The combination of this novel formulation and algorithm to solve it is called DPS-MAX, which provably maximizes average detections per second (DPS).

A. Semi-MDP Model

The proposed DPS-MAX approach is the combination of a novel formulation of the continual area sweeping problem as a Semi-Markov Decision Process (SMDP) along with a deep reinforcement learning algorithm to solve it. This section describes the SMDP formulation, which consists of $(\mathcal{S}, \mathcal{A}, R, P)$:

$\mathcal{S}$ is the state space. Each state consists of three components:

2D costmap: Notated as G in Section II, the discretized grid of the environment is included. This grid is represented by a 2D array where a cell that has an obstacle is given a value of 1, and others are given a 0.

Robot position: The robot's position in the environment is represented with a grid where the robot's current cell is 1 and the remaining cells are 0.

Event uncertainty: Suppose t seconds have passed since the robot has visited a particular cell. When $t = 0$, it is known that the robot has seen all of the events in the cell, but as t increases so does uncertainty. Encoding this uncertainty allows the robot to take events it has seen into consideration when making decisions. Under a Poisson distribution, the probability that 0 new events have appeared in a cell after time t_d is $\exp(-\alpha t_d)$, with α the rate at which events appear. Each grid cell is filled in with this probability. $t_d = \infty$ if the cell has never been seen. This formulation does not assume that event appearance is exactly Poisson; rather, these probabilities provide initial information that the function approximator can later use to learn the true dynamics of event appearance.

Combined with a CNN function approximator, these grid representations encode the assumption that local spatial regions of the state should be similar (see Section IV-D).

$\mathcal{A}$ is the action space, which includes all empty cells in G, or in other words, the free spaces that the robot can navigate to. The robot takes one action a_n at each decision step n. The key difference between an SMDP and an MDP is that different actions can have different durations. It takes time for the robot to physically move, so actions that move the robot to a far away cell take more time than actions that cause the robot to navigate a shorter distance.

$P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition kernel. A key component of P is the unknown probability of event appearance.

$R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a measurable function denoting the reward given for a transition, defined in section IV-B.

A stationary policy π describes the action to take in a given state, and is thus a map from $\mathcal{S}$ to probability measures on $\mathcal{A}$.

The most common way to deal with non-episodic tasks is discounting future rewards. In the continual setting, the goal is not to maximize total rewards, but rather to optimize long-term averages. Thus we instead use the following average reward formulation:

ρ^π is the average reward function:

$$\rho^\pi(\mu) \coloneqq \liminf_{n\to\infty} \frac{1}{n} \mathbb{E}\Big[\sum_{k=0}^{n-1} R(s_k, a_k, s_{k+1})\Big] \qquad (1)$$

where μ is an initial state distribution, and the expectation is taken with respect to the appropriate measure derived from π and μ [11]. For convenience, when $\mu(s) = 1$ for some state s, we use the notation $\rho^\pi(s)$.

The optimal differential value function, then, is:

$$Q^*(s,a) = \mathbb{E}_P R(s,a,s') - \sup_\pi \rho^\pi(s) + \mathbb{E}_P\big[\max_{a' \in \mathcal{A}} Q^*(s', a')\big]$$

The goal under this formulation is to approximate Q^*, from which we can derive an optimal policy.

B. Reward Construction

The reward construction of DPS-MAX should maximize average detections per second (DPS). Note that it is not sufficient to construct a reward function where the value is 1 if an event was detected, 0 otherwise. Average reward is maximized as in Equation 1, which maximizes the average number of detections per decision step. Since the problem formulation is as an SMDP, however, maximizing *detections per decision step* is not the same as maximizing *detections per second*. Special care is needed, because actions take different lengths of time, making these two metrics not even approximately similar.

Reward construction is an important part of SMDP design, and many schemes deal with handling the time and decision step mismatch [12]. The new reward function is designed

specifically for the case of optimizing a rate, such as detections per second.

Proposition 1. *Take* $\{(s_n, a_n)\}_{n\geq 0} \subset \mathcal{S} \times \mathcal{A}$ *to be a trajectory generated from a policy* π*. Let* $\{\phi_n\}_{n\geq 0} \subset \mathbb{R}$ *be a sequence, and* $\{t_n\}_{n\geq 0} \subset \mathbb{R}$ *be an increasing sequence denoting the associated environmental time. Construct* R *in the following way:*

$$R(s_0, a_0, s_1) \coloneqq 0$$
$$R(s_n, a_n, s_{n+1}) \coloneqq (n+1)\frac{\phi_{n+1}}{t_{n+1}} - n\frac{\phi_n}{t_n}$$

Then $\rho^\pi(s_0) = \liminf_{n\to\infty} \frac{\mathbb{E}\,\phi_n}{t_n}$

Proof.
Substituting in (1):

$$\rho^\pi(s_0) = \liminf_{n\to\infty} \frac{1}{n}\mathbb{E}\left[\sum_{k=0}^{n-1}(k+1)\frac{\phi_{k+1}}{t_{k+1}} - k\frac{\phi_k}{t_k}\right]$$

The sum telescopes, leading to:

$$\rho^\pi(s_0) = \liminf_{n\to\infty} \frac{1}{n} n \frac{\mathbb{E}\,\phi_n}{t_n}$$

□

The corollary of this proposition is that setting ϕ_n to be the number of detections seen at decision step n provably optimizes average detections per second.

C. Deep R-Learning

R-Learning is a classical approach for learning an optimal differential value function [13], [14]. Its purpose is to handle infinite-horizon tasks where finding a policy to maximize average reward is more meaningful than temporal discounting. For this problem, discounting is not a good fit as in order to optimize average detections per second (DPS), detections in the future cannot be considered less valuable. To represent the value function, a suitable function approximator is needed. Even in simple tasks, a tabular representation cannot be directly used as the event uncertainty component of the state described in section IV-A leads to uncountably many states. We introduce a deep neural network variant of R-Learning based on double DQN [15], which allows for the integration of neural networks with double Q-Learning.

Algorithm 1 describes the new algorithm. The key changes to double DQN are highlighted here. First, the target in line 9 reflects the R-Learning update by subtracting out the running average reward estimate. Lines 11 and 12 compute the change to ρ based on the temporal difference error. In line 12, the TD errors of the batch are averaged so long as the actions taken were close to optimal, and is thus slightly different from the original R-learning algorithm which only updates ρ when non-exploratory actions are taken. As a result δ essentially controls a bias-variance trade off of average reward updates. A low δ will lead to lower bias as it is closer to approximating ρ^{π^*}, but there will be higher variance as it takes smaller batch averages. If line 12 attempts to take the average of an empty set, then the subsequent if-statement will not execute.

Algorithm 1 Deep R-Learning

1: Initialize empty experience replay buffer $\mathcal{D}$.
2: Initialize network Q with random weights $\boldsymbol{\theta} = \boldsymbol{\theta}^-$.
3: Initialize $\rho = 0$.
4: **for** t = 1, …, M **do**
5: Select an action a_t according to an action selection mechanism like ϵ-greedy.
6: Execute a_t and store the resulting transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$.
7: Randomly sample a batch of transitions $\{(s_j, a_j, r_j, s_{j+1})\}$ from $\mathcal{D}$.
8: Let $q_{max} = Q\left(s_{j+1}, \operatorname{argmax}_a Q\left(s_{j+1}, a; \boldsymbol{\theta}\right); \boldsymbol{\theta}^-\right)$
9: Let $y_j = r_j - \rho + q_{max}$
10: Take a gradient descent step on $\left(y_j - Q\left(s_j, a_j; \boldsymbol{\theta}\right)\right)^2$.
11: Let $\Delta_j = y_j - Q\left(s_j, a_j; \boldsymbol{\theta}\right)$
12: Let $\Delta = \text{avg}\{\Delta_j \text{ s.t. } |Q(s_j, a_j) - \max_a Q(s_j, a)| < \delta\}$
13: **if** Δ is well-defined **then**
14: $\rho = \rho + \alpha\Delta$ for learning rate α
15: **end if**
16: Every τ steps, set $\boldsymbol{\theta}^- = \boldsymbol{\theta}$.
17: **end for**

D. Q Function Representation

To represent Q, Algorithm 1 uses an encoder-decoder network as a way of exploiting the topology of $\mathcal{S}$ and $\mathcal{A}$. For a practical map, there can be millions of actions, since the agent can choose to move anywhere (resulting in close to height×width of G number of actions). Value based methods are normally poorly suited for such a large action space, but this choice of architecture overcomes that limitation. Due to convolutional layers, updates made to the Q-value of a state-action pair immediately generalize to a local neighborhood. Figure 1 illustrates the architecture, which shows an encoder-decoder network where the environment map, robot position, and event uncertainty are represented as grids and fed in as the input. The output is the action-value for each cell (action) in the map. The max-pool and upsampling layers help coalesce the action values of neighbors.

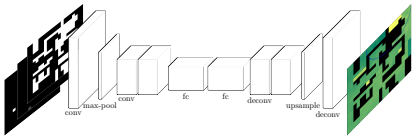
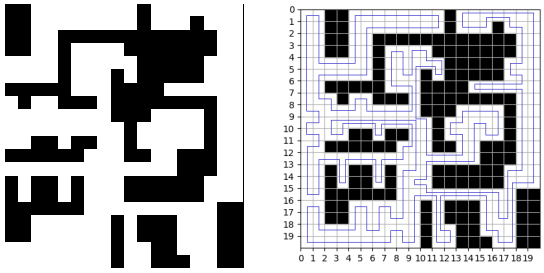


Fig. 1: In these experiments, the architecture comprises a 32×5×5@3 conv, 2×2@2 max-pool, two 16×4×4@2 conv, and a 500 unit fully connected layer, while the decoder part is the reverse with the conv layers swapped for deconv layers and upsampling for the max-pool.



(a) (b)

Fig. 2: (a) Gridworld where black represents walls. In 5 random cells, events have a periodicity ranging from 10 to 50 seconds in the periodic case, or show up with a fixed probability ranging from $1/10$ to $1/50$ per second in the binomial case. (b) One coverage patrolling path for (a).

V. Experiments

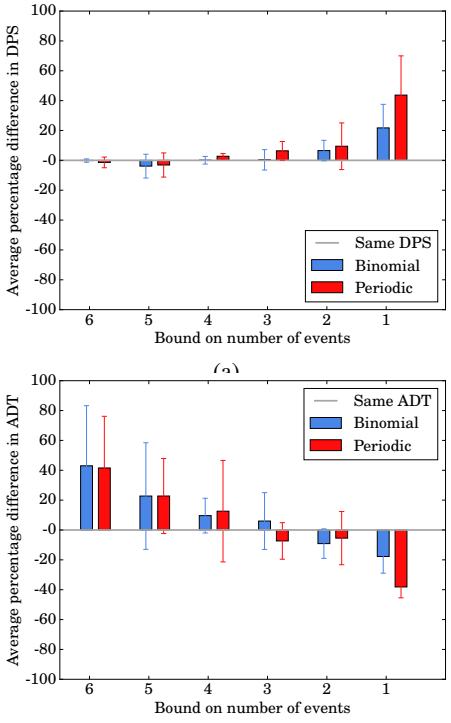
We evaluate DPS-MAX on an abstract gridworld. The gridworld is used to compare the performances of the DPS-MAX approach and the ADT-GREEDY algorithm under different environmental assumptions. We hypothesize that (1) DPS-MAX will outperform ADT-GREEDY when the environment violates the binomial assumption and the unbounded events assumption as defined in section III-C.2, (2) ADT-GREEDY will do better under the ADT metric, while DPS-MAX will do better under the DPS metric, and (3) DPS-MAX is able to incorporate knowledge that influences event appearance into its state space, thereby leading to improved performance.

A. Gridworld Experiments

This evaluation tests on a gridworld in order to evaluate the performance of the proposed algorithm under the different environmental assumptions and the two metrics. We test our hypothesis that DPS-MAX outperforms ADT-GREEDY when assumptions of ADT-GREEDY are violated, and study how much the ADT and DPS metrics align with each other.

1) Setup: Figure 2a illustrates the setup for the following gridworld experiments. A 20×20 grid is populated with random locations at which events may occur. In each cell, either events occur periodically, or the number of events follow the binomial distribution. In the binomial case, events appear with a fixed probability between $1/10 - 1/50$ each time step, and in the periodic case, according to a fixed period between $10 - 50$ time steps. These events occur in 1 of 5 fixed locations which are randomly generated at the start of each experiment, with a probability or time period associated with each of the 5 locations at the start of the experiment. We also evaluate the effects of the bound on the number of events per grid cell by varying the bound from 6 to 1. This set of experiments tests the effect of the unbounded assumption made by the ADT-GREEDY algorithm, with 6 being closer to the original assumption and 1 completely violating it.

For each configuration, 8 grids of random event positions and occurrence probabilities/periods are generated. Since the instances have randomly generated event positions, the best achievable DPS and ADT are different for each instance. We compare DPS-MAX to ADT-GREEDY by taking the percentage difference in our DPS or ADT over the DPS or ADT of ADT-GREEDY, averaged across each configuration.



(b)

Fig. 3: (a) Average percentage difference in detections per second (DPS) of DPS-MAX over ADT-GREEDY, where higher than 0 means DPS-MAX outperforms ADT-GREEDY. (b) Average percentage difference in average detection time (ADT) of DPS-MAX over ADT-GREEDY, where lower than 0 means DPS-MAX outperforms ADT-GREEDY. Error bars report standard deviation.

In this set of experiments, the learning rate α in Algorithm 1 is set to 0.0001. The exploration strategy is to initialize the agent in a random position, run ϵ-greedy exploration for 50 steps, and then reset the agent to a random position. The low learning rate and the frequent resets are used to ensure sufficient exploration.[1] The stopping criterion for training is the following: after every 20,000 training steps, the model is evaluated by executing the policy at a random initial position, and training terminates if the DPS has not improved in the last 10 evaluations.[2]

2) Results: Figure 3a shows the average percentage difference in DPS, where higher than 0 means DPS-MAX detects more events per unit time than ADT-GREEDY. Figure 3b shows the average percentage difference in ADT, where lower than 0 means on average DPS-MAX takes less time between event appearance and detection than ADT-GREEDY.

As shown by both figures, DPS-MAX has the most advantage over the ADT-GREEDY approach when the binomial

[1]Otherwise, exploration tends to stick around grid cells with frequent events, and not cover enough of the state space; causing high variances in evaluations since the robot's initial position is random.

[2]On average, training terminated in around 400,000 steps.

and unbounded events assumptions are most violated. When event appearance is periodic and the number of events in each cell is bounded by 1, DPS-MAX achieves the best improvement in DPS (43.7%) and the most reduction in ADT (38.4%). The reduction in ADT is surprising because unlike ADT-GREEDY, DPS-MAX does not directly optimize for ADT. In fact, the two metrics align except for a few cases. For instance, when the bound is 4 in the periodic setting, DPS-MAX has better DPS but worse ADT compared to ADT-GREEDY.

When the bound on events in grid cells is high, DPS-MAX does not outperform ADT-GREEDY on either metric. One possible explanation is that detecting many events in one step gives a large reward, which causes instability in learning. Such a scenario is not the focus of this work, so we leave further investigation of this case to future work.

For comparison, we tested coverage patrolling [2], which uniformly covers all grid cells. In the case of binomial event appearance and bound = 1, following the path in Figure 2b leads to 266.7% in average percentage difference in ADT and -65.8% in average percentage difference in DPS compared to ADT-GREEDY. As expected, this performance is worse on both metrics than ADT-GREEDY and DPS-MAX, showcasing the advantage of biasing travel time in favor of cells with frequent events.

B. Incorporating Extra Knowledge

Leveraging information about external factors in the environment can improve performance. Consider a person moving around in a building doing an activity like throwing away trash. In this setting, the robot would benefit from incorporating knowledge of where the person is, as the appearance of events is highly correlated. While ADT-GREEDY does not provide a way to add such knowledge, DPS-MAX does by adding information to the state. To illustrate, we conduct an experiment in which a person walks randomly on the grid in Figure 2a and causes an event with a 30% chance in each step, with the number of events in each cell bounded by 1. The algorithm is the same as above with the only difference being that a grid with the person's location is added to the state for DPS-MAX. With the person's location incorporated, DPS-MAX achieves a 3922.6% increase in DPS over ADT-GREEDY and a 39.2% reduction in ADT. Without the extra information, the average increase in DPS is 631.7% and DPS-MAX has worse ADT than ADT-GREEDY with an average increase of 1029.2%. Thus, DPS-MAX allows the function approximator to learn the association between the person and events leading to significantly better performance.

Unlike ADT-GREEDY, DPS-MAX can also recognize previously seen geometric features, which mitigates re-learning on familiar environments. Please refer to our full paper for details.[3]

VI. Conclusion

This paper extends the formulation of the continual area sweeping problem using an SMDP, and proposes a deep R-learning approach to maximize average detections per second. These two components comprise the main contribution of this paper, which is the introduction of the novel DPS-MAX approach for the general class of continual area sweeping problems that we expect to arise frequently in the growing area of service robotics, and a demonstration that, under the assumptions most reflective of such scenarios, DPS-MAX significantly outperforms the prior state-of-the-art algorithm. Furthermore, DPS-MAX can discover structure in event occurrence (such as geometric features) and leverage extra state information (such as the location of a person). An initial demonstration of DPS-MAX on real service robots is presented.[4] An interesting direction for future work is to utilize DPS-MAX as a background default behavior that improves general knowledge of the environment when no other task is active.

[3]https://arxiv.org/abs/2006.00589

[4]https://youtu.be/KqEwfs1xqdw

References

[1] M. Ahmadi and P. Stone, "Continuous area sweeping: A task definition and initial approach," in *Proceedings of International Conference on Advanced Robotics (ICAR)*, 2005.

[2] H. Choset, "Coverage for robotics - A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, 2001.

[3] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence*, 2001.

[4] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2013.09.004

[5] N. Gatti, "Game theoretical insights in strategic patrolling: Model and algorithm in normal-form." in *ECAI*, 2008, pp. 403–407.

[6] N. Basilico, N. Gatti, and F. Amigoni, "Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder," *Artificial Intelligence*, vol. 184, pp. 78–123, 2012.

[7] B. Bošanskỳ, V. Lisỳ, M. Jakob, and M. Pěchouček, "Computing time-dependent policies for patrolling games with mobile targets," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 989–996.

[8] E. Ayvali, H. Salman, and H. Choset, "Ergodic coverage in constrained environments using stochastic trajectory optimization," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5204–5210.

[9] C. R. Ratto, K. R. Shipley, N. Beagley, and K. C. Wolfe, "Information surfing with the JHU/APL coherent imager," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XX*, S. S. Bishop and J. C. Isaacs, Eds., vol. 9454, International Society for Optics and Photonics. SPIE, 2015, pp. 510 – 524. [Online]. Available: https://doi.org/10.1117/12.2176338

[10] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, "Real-time area coverage and target localization using receding-horizon ergodic exploration," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 62–80, 2017.

[11] E. A. Feinberg, "On measurability and representation of strategic measures in markov decision processes," *Lecture Notes-Monograph Series*, pp. 29–43, 1996.

[12] M. Baykal-Gürsoy, "Semi-markov decision processes," *Wiley Encyclopedia of Operations Research and Management Sciences*, 2010.

[13] A. Schwartz, "A reinforcement learning method for maximizing undiscounted rewards," in *Proceedings of the tenth international conference on machine learning*, vol. 298, 1993, pp. 298–305.

[14] S. Mahadevan, "Average reward reinforcement learning: Foundations, algorithms, and empirical results," *Machine learning*, vol. 22, no. 1-3, pp. 159–195, 1996.

[15] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in *AAAI*, vol. 2. Phoenix, AZ, 2016, p. 5.