6-2023

Dynamic police patrol scheduling with multi-agent reinforcement learning

Songhan WONG

Waldy JOE

Hoong Chuin LAU
Singapore Management University, hclau@smu.edu.sg

Citation

WONG, Songhan; JOE, Waldy; and LAU, Hoong Chuin. Dynamic police patrol scheduling with multi-agent reinforcement learning. (2023). *Proceedings of the 17th International Conference on Learning and Intelligent Optimization, Nice, France, 2023 June 4-8*. 567-582.
Available at: https://ink.library.smu.edu.sg/sis_research/8347

Dynamic Police Patrol Scheduling with Multi-Agent Reinforcement Learning

Songhan Wong, Waldy Joe, and Hoong Chuin Lau(✉)

School of Computing and Information Systems, Singapore Management University, Singapore, Singapore
songhanwong.2020@mitb.smu.edu.sg, waldy.joe.2018@phdcs.smu.edu.sg, hclau@smu.edu.sg

Abstract. Effective police patrol scheduling is essential in projecting police presence and ensuring readiness in responding to unexpected events in urban environments. However, scheduling patrols can be a challenging task as it requires balancing between two conflicting objectives namely projecting presence (*proactive* patrol) and incident response (*reactive* patrol). This task is made even more challenging with the fact that patrol schedules do not remain static as occurrences of dynamic incidents can disrupt the existing schedules. In this paper, we propose a solution to this problem using Multi-Agent Reinforcement Learning (MARL) to address the Dynamic Bi-objective Police Patrol Dispatching and Rescheduling Problem (DPRP). Our solution utilizes an Asynchronous Proximal Policy Optimization-based (APPO) actor-critic method that learns a policy to determine a set of prescribed dispatch rules to dynamically reschedule existing patrol plans. The proposed solution not only reduces computational time required for training, but also improves the solution quality in comparison to an existing RL-based approach that relies on heuristic solver.

Keywords: Reinforcement Learning · Multi-Agent · Dynamic Dispatch and Rescheduling · Proximal Policy Optimization · Police Patrolling

1 Introduction

Effective scheduling of police patrols is essential to project police presence and ensure readiness to respond to unexpected events in urban environments. Law enforcement agencies have the challenging task of balancing two conflicting objectives of projecting presence (*proactive* patrol) and incident response (*reactive* patrol). When an unexpected incident occurs, complex and effective response decisions must be made quickly while minimizing the disruption to existing patrol schedule. Such a decision is complex because each decision contains multiple components, namely which agent needs to be dispatched to respond to the incident and secondly which existing schedules are disrupted and/or require some re-planning. In a real-world environment, the scale of the problem needs to consider multiple patrol areas and teams. Multiple patrol teams need to operate in cooperative manner to maximize the effectiveness of police patrolling. Hence, it is challenging to develop an efficient real-time strategy for reallocating resources when an incident occurs.

In this paper, we present a solution based on Multi-Agent Reinforcement Learning (MARL) that enables rescheduling of patrol timetable whenever dynamic events occur. The problem addressed in this paper is based on the Dynamic Bi-Objective Police Patrol Dispatching and Rescheduling Problem (DPRP) introduced in [9]. This problem is a variant of Dynamic Vehicle Routing Problem (DVRP) with the element of university time-tabling scheduling incorporated and in the context of cooperative multi-agent environment.

The key contribution of the paper is the successful application of Asynchronous Proximal Policy Optimization (APPO) policy-gradient method with dispatch-rules based actions for solving a dynamic patrol scheduling problem based on a real-world police patrolling environment. Our solution method emphasizes on the use of patrol dispatch rules to significantly reduce the computational time in making such a complex decision. In addition, RL is used to learn the policy in choosing the dispatch rule rather than relying on some fixed heuristic rules. We experimentally demonstrate that our proposed solution method is able to reduce training time by a factor of 40 while improving the quality of the solution by around 10% against the benchmark approach [9].

2 Background

The police patrol routing problem can generally be seen as an extension of the stochastic DVRP. In addition to route optimization, this routing problem also needs to consider scheduling aspect i.e. when and how long an agent remains in a particular node within a given route. A patrol unit patrolling in existing allocated area can be dispatched to an emergency call, and a redistribution of patrol resources is necessary to ensure optimal deployment. Existing works in the literature such as [2,5] mostly addresses the offline planning aspect of patrolling problem where the planned schedule is assumed to be static. These solutions include genetic algorithm, routing policies, and local search based on the network Voronoi diagram [7,16]. However, significant operational challenges lie mainly in the dynamic planning aspect, since there may be disruption from an unforeseen event that requires dispatch and re planing of the existing patrol schedules.

2.1 Scheduling Problem with Reinforcement Learning

The use of RL to solve dynamic scheduling problem has gained traction in the community [10,11,17] in recent years. Single-agent reinforcement learning (RL) algorithm suffers from a curse of dimensionality since the state-action space grows exponentially as the number of agents increases, particularly in the context of large-scale patrolling environment in modern metropolis. Cooperative multi-agent reinforcement learning (MARL) algorithms are necessary for solving such problem. Based on current works on MARL [1,8,12], MARL approach training schemes can generally be classified into centralized setting, decentralized setting with networked agents, and fully decentralized with independent learners setting. Common challenges of MARL include non-unique learning goals among agents, non-stationary environment, scalability, and partial observability [18].

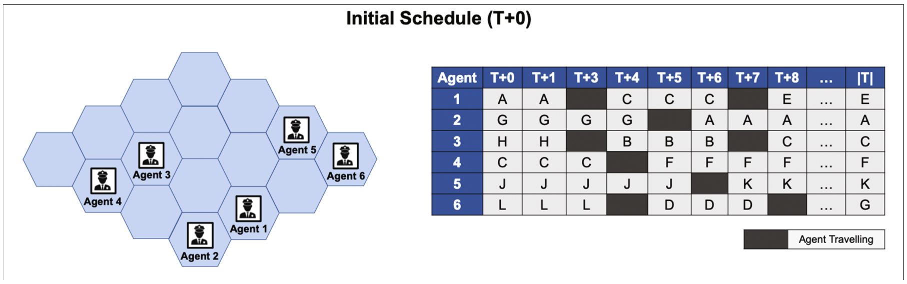
In a closely related problem of the Job-Shop Scheduling Problem (JSSP), the use of Proximal Policy Optimization (PPO) [14] with Graph Neural Network algorithm was able to learn the priority dispatch rule for JSSP from first principles without intermediate handcrafted rules [17]. While such generalized learning approach is novel, it is important to note that the action-space of JSSP is reduced as the solution space narrows over time due to precedence constraint. This, however is not true for DPRP, where the same graph node can be visited multiple times and the action-space does not reduce over time. Another proposed method [11] based on actor-critic Deep Deterministic Policy Gradient (DDPG) algorithm in multi-agent environment only considers a set of simple dispatching heuristics rules in the agent action space. The constrained action-state space proved to be efficient for learning and sufficiently effective for solving complex scheduling problem. However, such a framework requires in-depth prior domain knowledge, and retraining of the model is needed when input parameters vary.

For our problem DPRP, [9] demonstrated successful application of a deep RL-based method with a rescheduling heuristic based on input and constraints from the real-world environment. The proposed method combines the value function approximation through Temporal-Difference (TD) learning with experience replay and an ejection chain heuristic solver. The solution is able to compute dispatch and rescheduling decisions instantaneously as required in real-world environment. There were also other works that addressed a similar variant of such problem. Most of these approaches adopted a two-stage decomposition that learns the dispatching and scheduling policies in separate stages (see [3,4]).

3 Problem Description

The problem being addressed in this paper closely represents the scenario of police patrolling in a modern and large city in which police needs to respond to incidents of various types within very short time frames (in less than 10 min within receiving the call for response). At the start of the day, police officers are assigned to different patrol areas under their jurisdiction. A centralized authority is tasked to plan the resource to ensure sufficient patrol presence for the entire city. In addition to patrol duties, the plan must also adapt to incidents arising in real time, to ensure that police officers are able to respond to incidents as soon as possible while not compromising the level of patrols within each jurisdiction.

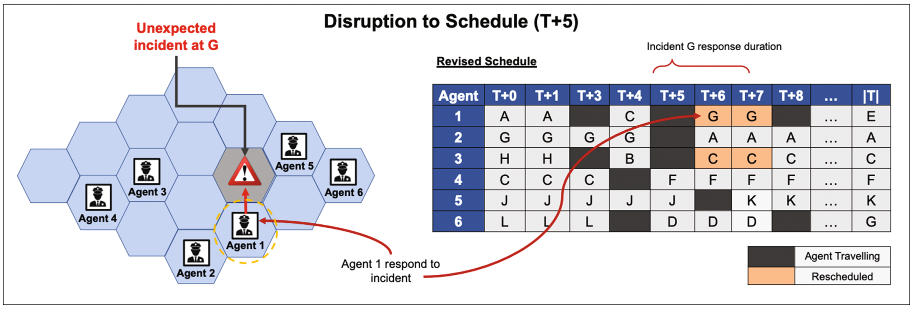
Figure 1 shows an example of multiple patrol officers dispatched to different patrol areas based on the initial schedule given. We assume that all patrol agents have homogeneous capability.



Agent	T+0	T+1	T+3	T+4	T+5	T+6	T+7	T+8	...	|T|
1	A	A	■	C	C	C	■	E	...	E
2	G	G	G	G	■	A	A	A	...	A
3	H	H	■	B	B	B	■	C	...	C
4	C	C	C	■	F	F	F	F	...	F
5	J	J	J	J	J	■	K	K	...	K
6	L	L	L	■	D	D	D	■	...	G

■	Agent Travelling

Fig. 1. Schematic diagram that shows multi-agent patrol environment and the initial timetable schedule at $T = 0$.


Disruption to Schedule (T+5)
Unexpected incident at G
Agent 1
Agent 2
Agent 3
Agent 4
Agent 5
Agent 6
Agent 1 respond to incident
Revised Schedule
Incident G response duration

Agent	T+0	T+1	T+3	T+4	T+5	T+6	T+7	T+8	...	|T|
1	A	A	■	C	■	🟧 G	🟧 G	■	...	E
2	G	G	G	G	■	A	A	A	...	A
3	H	H	■	B	■	🟧 C	🟧 C	C	...	C
4	C	C	C	■	F	F	F	F	...	F
5	J	J	J	J	J	■	K	K	...	K
6	L	L	L	■	D	D	D	■	...	G

■	Agent Travelling
🟧	Rescheduled

Fig. 2. An incident happened at $T + 5$ and patrol agent 1 is deployed to respond to the incident. Following the disruption, a rescheduling is made to the time table of the relevant patrol agents (patrol agent 1 and 3) at $T \geq 5$.

Figure 2 shows an example of a situation where an incident occurs at location G during in a given day or shift. In this case, Agent 1 is deployed to respond to the incident. The original patrol location of C is changed to G at $T = 5$ and this change may result in the need to reschedule the plans of other agents. For simplicity, we assume that only one incident can occur one at a time. In addition, we assume that the condition of partial observability does not exist with the presence of a central authority akin to a police headquarter.

4 Model Formulation

The objective of the problem is for every agent to make rescheduling and dispatching decision at every decision epoch in order to maximize both global patrol presence and response rate to dynamic incidents.

We model our problem as a fully cooperative multi-agent Markov Decision Process - $(\mathbf{S}, \mathbf{A}, \mathbf{T}, \mathbf{R})$ where $\mathbf{S}$ is a state for the timetable, $\mathbf{A}$ is a set of actions taken by the agents, $\mathbf{T}$ is the transition probability vector between state for different state-action pairs, and $\mathbf{R}$ is the immediate reward transitioning from state s to s' given action a.

The objective of the problem is for every agent to make rescheduling and dispatching decision at every decision epoch in order to maximize both global patrol presence and response rate to dynamic incidents (Table 1).

Table 1. Set of notations used in this paper.

Notation	Description
I	Set of patrol agents, $I \in \{1, 2, 3, \cdots, \|I\|\}$
J	Set of patrol areas, $J \in \{1, 2, 3, \cdots, \|J\|\}$
T	Set of time periods in a shift, $T \in \{1, 2, 3, \cdots, \|T\|\}$
k	Decision epoch
t_k	Time period in a shift where decision epoch k occurs, $t_k \in T$
$a(k)$	Set of dispatch actions taken by all agents at decision epoch k
$x_i(k)$	Dispatch action taken by agent i at decision epoch k
$\delta_i(k)$	A schedule of patrol agent i at decision epoch k
$\delta(k)$	A joint schedule of all patrol agents at decision epoch k
$\delta_{-i}(k)$	A joint schedule of all patrol agents except for agent i at decision epoch k
$\delta_i^a(k)$	A schedule of patrol agent i after executing action a at decision epoch k
$\delta^a(k)$	A joint schedule of all patrol agent after executing action a at decision epoch k
τ_{target}	A response time target
τ_{max}	A maximum buffer time for response time for incident
τ_k	Actual response time to incident at epoch k
$D_h(\delta^{'}, \delta)$	Hamming distance between schedules $\delta^{'}$ and δ
$d(j, j')$	Travel time from patrol area j to another patrol area j'
Q_j	Minimum patrol time for patrol area j
σ_j	Patrol presence for area j in terms of ratio of the effective patrol time over Q_j
ω_k	State representation of dynamic incident that occurs at decision epoch k
$N(k)$	State representation of patrol agents availability at decision epoch k
$\Omega_{i,k}$	Patrol status of agent (patrolling or travelling) i at decision epoch k
$D_{i,k}$	Patrol or travel destination of agent i at decision epoch k
$M_{i,k}$	Travel arrival time of agent i at decision epoch k

4.1 State

The state of the MDP, S_k is represented as the following tuple: $\langle t_k, \delta(k), \omega(k), N(k) \rangle$. t_k is the time period in a shift where decision epoch k occurs. $\delta(k)$ is the joint schedule of all patrol agents, $\omega(k)$ is the dynamic incident, and $N(k)$ is the patrol agents' availability at decision epoch k.

Joint Schedule. The joint schedule, $\delta(k)$ has a dimension of $|T| \times |I| \times |J|$, which represents the time tables for all patrol agents.

Incident. A dynamic incident, $w(k)$ occurs at decision epoch k and is described as the following tuple: $\left\langle \omega_k^j, \omega_k^t, \omega_k^s \right\rangle$ where $\omega_k^j \in J$ refers to the location of the incident, $w_k^t \in T$ refers to the time period when the incident occurs, and w_k^s refers to the number of time periods required to resolve the incident.

Agents' Availability. The agents' availability, $N(k)$ represents the availability of patrol agents in the decision epoch k. It comprises the individual agent's availability, $N_i(k)$ for every agent i.

4.2 Action

$a_i(k)$ is a dispatch action taken by an agent i at decision epoch k. The set of action space is a set of dispatch rules described in Table 2. The action taken by the agent determines the state of the agent's individual time table at the next time step. The selection of action is made with an ϵ-greedy strategy.

Table 2. List of dispatch-rule-based actions for patrol agents.

Dispatch Rule	Description
a1. Respond to incident	Travel to the location of incident if there is an occurrence of incident. Otherwise, this action is not allowed
a2. Continue	Continue to patrol the same area or continue traveling to the destination patrol location if the agent was in the midst of traveling to another patrol area. This heuristics tries to minimizes deviation from initial schedule as the initial schedule was optimal for patrol presence in a situation where no unforeseen incident occurs
a3. Patrol an unmanned area	Travel to an unmanned patrol area that yields the best patrol presence utility. If multiple patrol areas have same the utility, randomly select one. This heuristics is a greedy approach that aims to maximize patrol presence utility of the schedule, with the assumption that unforeseen incidents may occur in unmanned patrol areas
a4. Patrol an existing manned area by other agents	Patrol an existing area currently being patrolled by other agents. Choose the area that has the best patrol presence utility. If multiple patrol areas have the same utility, randomly select one. This heuristics also aims to maximize patrol presence utility, but allowing the agent to take over an existing patrol area of another agent if the original agent needed to be dispatched elsewhere
a5. Nearest and least disruptive	Patrol the next nearest location such that it results in least deviation from the initial schedule. If multiple patrol areas have same the travel distance, randomly select one. This heuristics tries select a patrol area in such a way that it minimizes travel time and deviation from the initial schedule

4.3 Transition

A decision epoch k occurs at every time step. We move to the next decision epoch $k+1$ after all agents have completed their actions, transiting from the pre-decision state S_k to the post-decision state S_{k+1}. In our formulation, the transition between state is deterministic, and we let transition probability $\mathbf{T} = 1$ for every state-action pair. It is deterministic as in if an agent has chosen an action, there is no possibility that the agent deviates from the chosen action.

4.4 Constraints

We subject our final schedule $\delta(T)$ at the end of the last decision epoch to the following soft constraint:

$$D_h(\delta(T), \delta(0)) \leq D_{h,\max} \tag{1}$$

where $D_h(\delta(T), \delta(0))$ is the Hamming distance of the final schedule with respect to the initial schedule $\delta(0).D_{h,\max}$ is the maximum Hamming distance allowed. The constraint helps minimize disruption to our existing schedule caused by rescheduling.

4.5 Patrol Presence

Before discussing the reward function, we define patrol presence as the number of time periods each patrol area is being patrolled. Every patrol area j must be patrolled for a minimum of Q_j time periods in a given shift. A schedule with good patrol presence seeks to maximize the time spent patrolling while minimizing the travel time of patrol agents when moving to different patrol areas. The patrol presence utility function $f_p(\delta)$ is defined as the following

$$f_p(\delta) = \frac{\sum_{j \in J} U_p(j)}{|T| \times |I|} \tag{2}$$

where $U_p(j)$

$$\begin{aligned} U_p(j) &= \min(\sigma_j, 1) + 1_j \times e^{-\beta(\sigma_j - 1)} \\ 1_j &= \begin{cases} 1, \sigma_j > 1 \\ 0, \sigma_j \leq 1 \end{cases} \end{aligned} \tag{3}$$

where β is coefficient for patrol presence utility and patrol presence σ_j is defined as

$$\begin{aligned} \sigma_j &= \frac{\sum_{t=1}^{|T|} p_{j,t}}{Q_j} \\ p_{j,t} &= \begin{cases} 1, \text{patrol is present at area } j \text{ at time step } t \\ 0, \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

This utility function measures the utility of each patrol in each patrol area with respect to the minimum patrol requirement of that area, and additional patrol time comes with a diminishing return of utility beyond the minimum requirement.

4.6 Reward Function

The reward function $R\big(S_k, a_i(k)\big)$ takes into consideration of three factors; patrol presence, incident response, and deviation from existing schedule. Note that the reward at $t < T$ only considers incident response, while the final reward when at the end of episodes at $t = T$ includes additional factors of patrol presence reward and schedule deviation penalty. $f_r\big(a_i(k)\big)$ quantifies the success of an incident response when action $a_i(k)$ is taken by agent i. Similar to the patrol presence utility, any incident that is responded later than the target time will incur a reduced utility. The reward function is defined as

$$R\big(S_k, a_i(k)\big) = \begin{cases} f_r\big(a_i(k)\big) + f_p\big(\delta(t_k)\big) - p_r\big(\delta(t_k)\big), & t_k = T \\ f_r\big(a_i(k)\big), & t_k < T \end{cases}$$
$$f_r(a_i(k)) = \begin{cases} U_r = \exp^{-\alpha \times \mathbf{max}(0, \tau_k - \tau_{target})}, & \tau_k > 0 \\ 0, & \tau_k = 0 \text{ (Incident not responded)} \end{cases} \tag{5}$$

where α is the coefficient for response utility of a late response, τ_k is the response time taken at decision epoch k and τ_{target} is the target response time.

The penalty function for the deviation of the schedule $p_r(\delta(t_k))$ based on the Hamming distance D_h is defined as the following step functions:

$$p_r\big(\delta(t_k)\big) = C_1 \cdot D_h\big(\delta(t_k), \delta(0)\big) + C_2 \cdot 1_H\big(\delta(t_k), \delta(0)\big)$$
$$1_H\big(\delta(t_k), \delta(0)\big) = \begin{cases} 0, D_h\big(\delta(t_k), \delta(0)\big) \le D_{h,\max} \\ 1, D_h\big(\delta(t_k), \delta(0)\big) > D_{h,\max} \end{cases} \tag{6}$$

where C_1, C_2 are the weights of the Hamming distance penalty coefficients.

5 Solution Approach

The RL algorithm selected for our solution approach is an asynchronous variant of Proximal Policy Optimization (APPO) [15] algorithm based on IMPALA [6] actor-critic architecture. IMPALA is an off-policy actor-critic algorithm that decouples acting and learning, which allows multiple actors to generate experience in parallel, creating more trajectories over time. The off-policy RL algorithm uses the trajectories created by policy μ (behavior policy) to learn the value function of target policy π.

At the start of each trajectory, the actor updates its own policy μ in response to the latest policy from the learner, π, and uses it for n steps in the environment. After n steps, the actor sends the sequence of states, actions, and rewards along with the policy distributions to the learner through a queue. Batches of experiences collected from multiple actors are used to update the learner's policy. Such design allows the actors to be distributed across different machines. However, there is a delay in updating the policies between actors and the learner, as the learner policy π may have undergone several updates compared to the actor's policy μ at the time of the update. Therefore, an off-policy correction method called V-trace is used here.

V-Trace Target. For a trajectory state (x_t, a_t, x_{t+1}, r_t), the $n-$steps V-trace target for $V(x_s)$ at time s is given as,

$$v_s \overset{\text{def}}{=} V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \left(\prod_{i=s}^{t-1} c_i \right) \rho_t (r_t + \gamma V(x_{t+1}) - V(x_t)) \tag{7}$$

where $\rho_t = \min\left(\bar{\rho}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}\right)$ and $c_i = \min\left(\bar{c}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}\right)$ are truncated importance sampling weights. $\bar{\rho}$ and $\bar{c}$ are truncation constants with $\bar{\rho} \geq \bar{c}$. In the case of on-policy learning, then $c_i = 1$ and $\rho_t = 1$, and (7) becomes

$$v_s = \sum_{t=1}^{s+n-1} \gamma^{t-s} r_t + \gamma^n V(x_{s+n}) \tag{8}$$

which is the on-policy $n-$step Bellman target. By varying $\bar{\rho}$, we change the target of the value function to which we converge. When $\bar{\rho} = \infty$ (untruncated), the value function of the v trace will converge to the target policy V_π; When $\hat{\rho} \to 0$ (untruncated), the value function converges to behavior policy V_μ. Any value of $\hat{\rho} < \infty$ indicates the value function of the policy somewhere between μ and π. The truncation constant $\bar{c}$ changes the speed of convergence.

V-Trace Actor-Critic Algorithm. For every iteration update, $V_\theta(s)$ is the value function of state s parametrized by θ, which is updated by

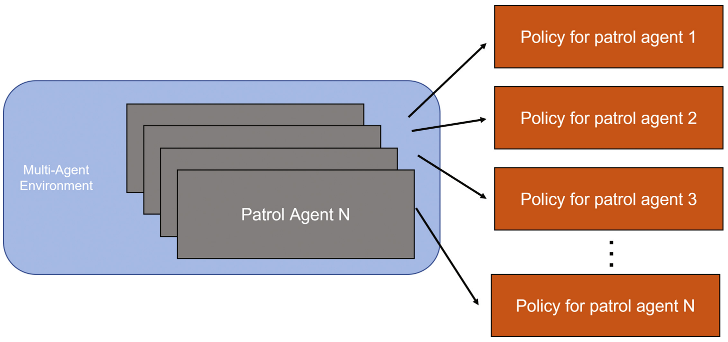
$$\Delta\theta = (v_s - V_\theta(s)) \nabla_\theta V_\theta(s) \tag{9}$$

and the policy parameters w is updated through policy gradient

$$\Delta w = \rho_s \nabla_w \log \pi_w(a_s|s)(r_s + \gamma v_{s+1} - V_\theta(s)) \tag{10}$$

Proximal Policy Optimization (PPO). The use of Proximal Policy Optimization (PPO) improve the training stability of our policy by avoiding excessive large policy update. PPO uses the following clipped surrogate objective function for policy update:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{11}$$



Multi-Agent Environment
Patrol Agent N
Policy for patrol agent 1
Policy for patrol agent 2
Policy for patrol agent 3
Policy for patrol agent N

Fig. 3. Multi-agent setup with individual policy for the patrol agents.

Algorithm 1. Dispatch-rule based scheduling RL algorithm

```
for each episode do
    Set initial schedule $\delta \rightarrow \delta_0$ from a set of initial schedules, generate a scenario with a set of incident scenarios $\{\omega_k, \omega_{k+1}, \cdots\}$
    Set entries of $\delta_i(t \geq k) \rightarrow \varnothing \quad \forall i \in I$
    for $t = t_k$ to $T$ do
        for each patrol agent $i$ (in random order) do
            takes a feasible patrol action with $\epsilon$-greedy strategy that decides the state of $\delta_i(t)$
        end for
    end for
    Revised schedule is completed $\delta \rightarrow \delta'$
end for
```

where θ is the policy parameter, $\mathbb{E}_t$ is the empirical expectation, r_t is the ratio of the probabilities under the new and old policies, $\hat{A}_t$ is the estimated advantage at time t, ϵ is a hyper parameter, usually 0.1 or 0.2.

As our environment is a multi-agent environment, we assign one policy network to each agent as shown in Fig. 3. Algorithm 1 describes the procedure for rescheduling according to the solution approach. We implemented our solution method using Ray RLlib library [13].

6 Experimental Setup

6.1 Environment

The patrol environment comprises hexagonal grids of size 2.2 km × 2.2 km derived from local police patrol sectors, each grid representing a patrol area. We have chosen a large patrol setup with $|I| = 11$, $|J| = 51$ for our environment. This patrol setup represents an environment with a relatively low agent-to-area ratio. The duration of a day shift is 12 h and is divided into 72 discrete 10-min time units. The maximum Hamming distance for the revised schedule $D_{h,\max}$ is set at 0.4.

6.2 Model Parameters

The input state vectors are flattened into a one-dimensional array as a concatenated input to a fully-connected neural network encoder. The encoder neural network has a size of 256 × 256, with *tanh* activation functions. The training batch size is 500. Learning rate $\alpha = 0.0001$, $\epsilon_{initial} = 0.6, \epsilon_{final} = 0.01$. We set the discount factor $\gamma = 1.0$ since the reward function for patrol presence and deviation from initial schedule is only evaluated at the end of the episode when $\tau_k = T$ when the revised schedule is complete. Agents must therefore take into consideration this final reward without discount when evaluating action choices in an earlier decision epoch. We set the maximum number of training episodes to be 5000 episodes.

6.3 Training and Test

During training, there are 100 samples of initial joint schedules for initialization. Each sample consists of an initial joint schedule for all patrol agents for the entire day. The initial schedules are obtained via a mixed linear integer program prior to training. We run a total of 5000 training episodes. During each training episode, an initial schedule is randomly sampled from the pool, and a set of incidents is generated based on Poisson distribution with λ set as 2 i.e. the rate of occurrences of incident is 2 per hour. A training episode ends when the time step reaches the end of the shift. The training results generally begin to converge after 3000 episodes. After training is completed, we evaluated the performance of our solution approach based on 30 samples of initial joint schedules separate from the training set.

6.4 Evaluation Metrics

Our evaluation considers the following metrics: **patrol presence score (%)**, **incidence response score (%)** and **deviation from original schedule (Hamming distance)**. A good solution should have both high patrol presence and incidence response scores (where 100% means all incidents are responded within the stipulated response time), and a low deviation from original schedule (where 0 means that the existing schedule remains unchanged). We benchmark the quality of our solution approach against two approaches:

- **Myopic rescheduling heuristic** - Baseline algorithm with ejection chain rescheduling heuristic for comparison with VFA-H and our approach APPO-RL;
- **Value Function Approximation heuristic (VFA-H)** [9] - An RL-based rescheduling heuristic with ejection chain based on a learnt value function.

7 Experimental Results

7.1 Solution Quality

The evaluation scores summarized in Table 3 show that our solution approach APPO-RL outperformed the VFA-H method. The patrol presence and incidence response scores of our method are higher than that of VFA-H by +10.6% (+12.6% vs +2%) and +6.6% (−9.9% vs −16.5%) respectively. However, since schedule deviation is set as a soft constraint in our proposed solution, we see that the maximum Hamming distance $D_{h,\max}$ of 0.403 obtained by our approach exceeded the threshold of 0.4, implying that in some cases the maximum Hamming distance imposed has been violated, while this constraint is not violated in the VFA-H's method. On average, however, it is encouraging to see that the Hamming distance ($D_{h,\text{mean}}$) is less than in VFA-H.

The biggest advantage of our solution approach is the improved computational efficiency in solving the DPRP problem. Compared to VFA-H, the training time required is about 40x less. This is due to the reduced search space as we limit the number of action states to a selected few dispatch rules.

Table 3. Evaluation metric scores and mean training time.

Metric	VFA-H	APPO RL
Δ in mean incidence response score over Myopic	+2%	+12.6%
Δ in mean patrol presence score over Myopic	−16.5%	−9.9%
$D_{h,\max}$	0.399	0.403
$D_{h,\text{mean}}$	0.387	0.342
Mean training time per episode (s)	436	10.9

Figure 4 presents the performance of the three approaches with respect to the number of training episodes.

7.2 Constraint Sensitivity Analysis

We conducted a constraint sensitivity analysis to evaluate the trade-off between solution quality and constraint satisfaction by varying the value of C_2, the coefficient of a step function penalty term linked to the Hamming distance constraint threshold of $D_{h,\max} = 0.4$. As shown in Fig. 5, as we decreased the soft-constraint penalty coefficient C_2 on Hamming distance, the response score generally improved while the patrol presence score remained largely at similar levels. This is expected as agents have fewer constraints to consider when responding to incidents.

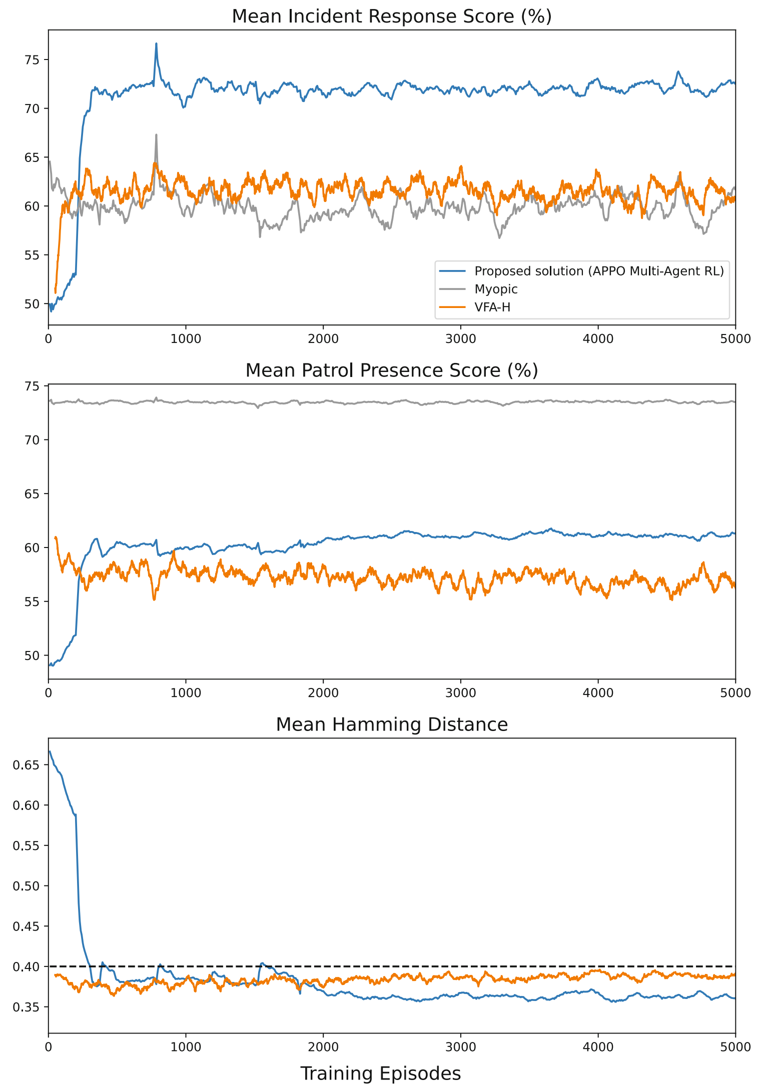


Fig. 4. Comparison of our solution approach (blue) with VFA-H (orange) and myopic (grey) baseline methods on various evaluation metrics. (Color figure online)

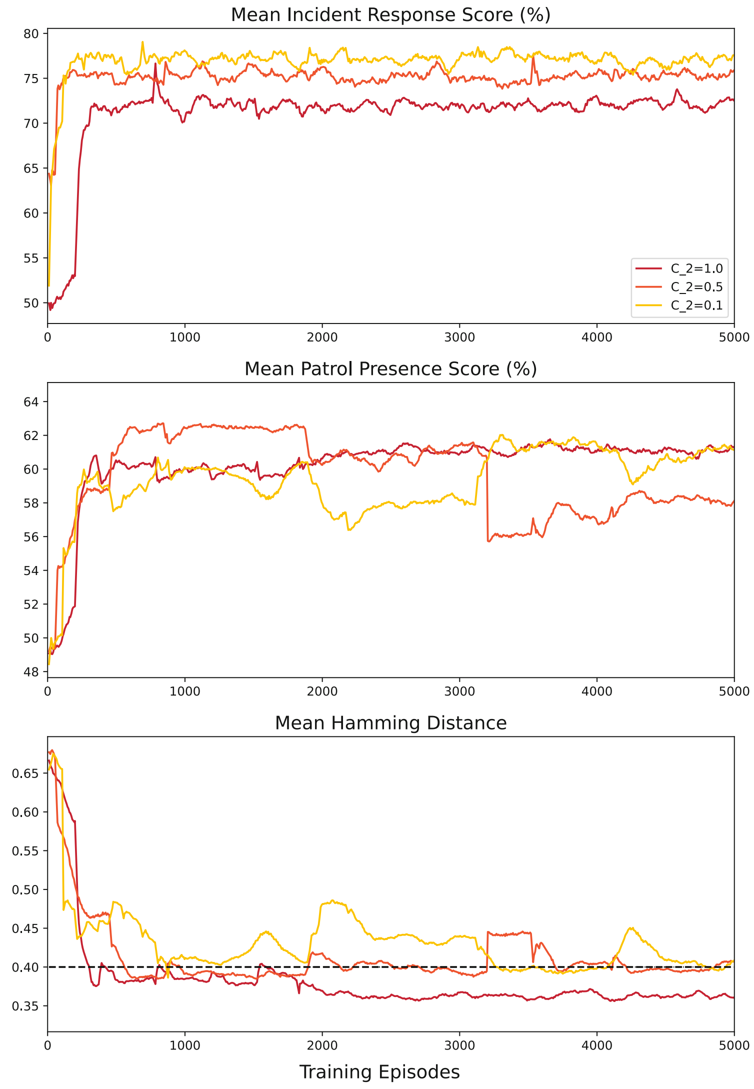


Fig. 5. Results of constraint sensitivity analysis based on our solution approach (darker colour indicates higher weightage of constraint penalty coefficient C_2). (Color figure online)

8 Discussion and Future Work

We discuss two major challenges of our work.

Generality. With a limited and handcrafted set of dispatch rules as actions, we are able to reduce the search space considerably even as the number of agents increases. Nonetheless, our approach requires some experimentation or prior domain knowledge in order to select a set of optimal dispatch rules suitable for the problem. This makes it challenging when we need to apply to a slightly different problem set, as the set of prescribed dispatch rules may have to be modified. It is also unclear if the learned policy is applicable as we vary the size of the environment. We may therefore have to train different policies as we change the size of patrol area and number of agents.

Constraint Satisfaction. The biggest drawback of our solution approach is that constraint satisfaction is not always guaranteed. In the DPRP problem, this constraint was set mainly to minimize disruption to patrol agents. Although such a violation of constraint is acceptable to some extent in our problem set, one can argue that this may not be applicable to other situations.

To conclude, we have demonstrated a successful application of multi-agent reinforcement learning technique in solving dynamic scheduling problem in the context of police patrolling. Our proposed method is able to improve solution quality while reducing training time considerably. We are in discussion with a local law enforcement agency to develop a prototype tool for real-world experimentation.

From the research standpoint, it would be worthwhile to apply this in a multi-agent environment where we have non-homogeneous patrol agents that need to collaborate with one another in order to respond to incidents. It would also be interesting to research further into the aspect of constrained RL to ensure that hard constraint satisfaction is met.

References

1. Canese, L., et al.: Multi-agent reinforcement learning: a review of challenges and applications. Appl. Sci. **11**(11), 4948 (2021)
2. Chase, J., Phong, T., Long, K., Le, T., Lau, H.C.: Grand-vision: an intelligent system for optimized deployment scheduling of law enforcement agents. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 31, pp. 459–467 (2021)
3. Chen, X., Tian, Y.: Learning to perform local rewriting for combinatorial optimization. In: 33rd Conference on Neural Information Processing Systems (2019)
4. Chen, Y., et al.: Can sophisticated dispatching strategy acquired by reinforcement learning? - a case study in dynamic courier dispatching system. In: AAMAS 2019: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (2019)
5. Dewinter, M., Vandeviver, C., Vander Beken, T., Witlox, F.: Analysing the police patrol routing problem: a review. ISPRS Int. J. Geo Inf. **9**(3), 157 (2020)
6. Espeholt, L., et al.: IMPALA: scalable distributed deep-RL with importance weighted actor-learner architectures. In: Proceedings of the 35th International Conference on Machine Learning, pp. 1407–1416 (2018)
7. Ghiani, G., Guerriero, F., Laporte, G., Musmanno, R.: Real-time vehicle routing: solution concepts, algorithms and parallel computing strategies. Eur. J. Oper. Res. **151**(1), 1–11 (2003)

8. Gronauer, S., Diepold, K.: Multi-agent deep reinforcement learning: a survey. Artif. Intell. Rev. **55**(2), 895–943 (2022)
9. Joe, W., Lau, H.C., Pan, J.: Reinforcement learning approach to solve dynamic bi-objective police patrol dispatching and rescheduling problem. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 32, pp. 453–461 (2022)
10. Li, W., Ni, S.: Train timetabling with the general learning environment and multi-agent deep reinforcement learning. Transp. Res. Part B: Methodol. **157**, 230–251 (2022)
11. Liu, C.L., Chang, C.C., Tseng, C.J.: Actor-critic deep reinforcement learning for solving job shop scheduling problems. IEEE Access **8**, 71752–71762 (2020)
12. OroojlooyJadid, A., Hajinezhad, D.: A review of cooperative multi-agent deep reinforcement learning. arXiv preprint arXiv:1908.03963 (2019)
13. Ray: Ray RLlib. https://www.ray.io/rllib
14. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
15. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: Proceedings of the International Conference on Machine Learning, pp. 387–395. PMLR (2014)
16. Watanabe, T., Takamiya, M.: Police patrol routing on network Voronoi diagram. In: Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication, pp. 1–8 (2014)
17. Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P.S., Chi, X.: Learning to dispatch for job shop scheduling via deep reinforcement learning. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1621–1632 (2020)
18. Zhang, K., Yang, Z., Başar, T.: Multi-agent reinforcement learning: a selective overview of theories and algorithms. In: Vamvoudakis, K.G., Wan, Y., Lewis, F.L., Cansever, D. (eds.) Handbook of Reinforcement Learning and Control. SSDC, vol. 325, pp. 321–384. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-60990-0_12