Adaptive and AI-Driven Vehicle Patrol Scheduling with Integrated Emergency Response

Majid Ghasemi
Dept. of Computer Science
Wilfrid Laurier University
Waterloo, Ontario, Canada
mghasemi@wlu.ca

Dariush Ebrahimi
Dept. of Computer Science
Wilfrid Laurier University
Waterloo, Ontario, Canada
debrahimi@wlu.ca

IECON 2025 – 51st Annual Conference of the IEEE Industrial Electronics Society | 979-8-3315-9681-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/IECON58223.2025.11221153

***Abstract*—Vehicle Patrol Scheduling (VPS) is vital for urban security, requiring broad coverage and rapid emergency responses. This paper addresses VPS through computational optimization to enhance patrol efficiency. We propose two methods: Adaptive Hill-Climbing-Based Patrol Scheduling (AHBPS2), which uses an iterative hill-climbing strategy with integrated emergency response, and Patrol Planning with Proximal Policy Optimization ($\mathcal{P}^4\mathcal{O}$), which models VPS as a Markov Decision Process and employs Deep Reinforcement Learning to learn optimal strategies. Extensive simulations on real-world urban maps show that $\mathcal{P}^4\mathcal{O}$ achieves higher visit frequencies and superior emergency handling compared to AHBPS2, underscoring the potential of these techniques for developing adaptive, efficient urban patrol systems.**

***Index Terms*—Patrol Scheduling, Urban Safety, Proximal Policy Optimization, Deep Reinforcement Learning**

I. Introduction

The practice of security patrolling is essential for ensuring business continuity and public safety. It deters illicit activities while safeguarding individuals and property. Professional companies hire trained staff and utilize simple technologies, such as GPS tracking, to design a predetermined list of routes for their personnel. Vehicle Patrol Scheduling (VPS), a variant of the Vehicle Routing Problem (VRP) introduced in [1], deals with scheduling patrols for public safety with a focus on efficient area coverage. While VRP seeks to optimize delivery routes for vehicles subject to constraints like capacity and time windows, it has diversified into many variants [2], [3]. Solving VPS presents challenges due to its NP-hardness and constraints such as shift durations, rest periods, and variable travel times. Patrol vehicles must maximize visits and revisit locations at specific intervals, necessitating scalable optimization techniques for real-time adjustments in large urban networks [4].

The literature reveals several existing approaches to VPS, a specialized instance of VRP, extensively studied within logistics, operations research, and Intelligent Transportation Systems. Traditional methods predominantly utilize optimization and heuristic techniques, but recent studies increasingly explore Reinforcement Learning (RL) solutions. For instance, Deep RL techniques like Deep Q-Learning and Double Deep Q-Learning have effectively optimized dynamic UAV patrol routes for disaster scenarios [5], while Multi-Agent RL (MARL) integrated with Geo-simulation has enhanced UAV patrol coordination in dynamic settings [6]. RL has also addressed dynamic vehicle routing challenges under traffic uncertainty through approximate dynamic programming, significantly outperforming static routing methods [7].

In this paper, we consider the VPS problem, where a fleet of vehicles is tasked with patrolling a sequence of locations in a given geographic area over a fixed operational time horizon (e.g., 12 hours from 8 PM to 8 AM). The objective is to maximize the overall number of locations that are being monitored, adhering to some constraints: Firstly, every location needs a minimum time of observation, i.e., vehicles must spend a certain amount of time at the location (e.g., 5 minutes). Secondly, there is a maximum time between two successive visits to any location (e.g., 30 minutes). Thirdly, all vehicles must return to the depot to rest after working for a certain number of hours (e.g., 2 hours). Finally, all vehicles must have a minimum amount of rest time before returning to work (e.g., 10 minutes). Furthermore, the patrolling system must incorporate possible emergency demands at certain locations, which cause the vehicles to alter their routing temporarily in order to fulfill these priorities with urgency.

To address this problem, we first propose a scalable heuristic method, Adaptive Hill-Climbing-Based Patrol Scheduling (AHBPS2), to efficiently solve large-scale dynamic environments. AHBPS2 is an enhancement of the AHBPS method introduced in our previous work that does not deal with emergencies. In contrast to AHBPS, which is aimed at building optimal patrol schedules adhering to some constraints, AHBPS2 improves flexibility by adding a dynamic re-optimization procedure that provides real-time adaptation for emergency situations. This aspect guarantees that the scheduling process performs well under extremely dynamic conditions where patrol demands regularly fluctuate.

Since heuristic methods do not necessarily produce optimal solutions, we also adopt a Reinforcement Learning (RL)-based approach to improve performance. By formulating the problem as a Markov Decision Process (MDP), we enable the use of a learning-based methodology that can iteratively improve patrol strategies through interaction with the environment. Our approach, Patrol Planning with Proximal Policy Optimization, $\mathcal{P}^4\mathcal{O}$, incorporates an emergency handling system and enhances the optimization process, producing near-optimal solutions for complex dynamic urban networks.

We evaluate the performance of AHBPS2 and $\mathcal{P}^4\mathcal{O}$ with extensive simulations over real urban maps with a variety of parameters, including network size, patrol time, rest time, and visitation frequency to some places. A close look at the results obtained by AHBPS2 and $\mathcal{P}^4\mathcal{O}$ demonstrates the scalability and effectiveness of the proposed solutions. Our overall result indicates that $\mathcal{P}^4\mathcal{O}$ surpasses AHBPS2 in all cases, particularly in complex scenarios with extensive metropolitan networks and a high rate of emergencies. Both methods are effective in emergency scenarios during simulation.

The remainder of this paper is structured as follows. Section II presents the system model and problem description. Section III introduces the AHBPS2 method. Section IV details our proposed RL approach, $\mathcal{P}^4\mathcal{O}$. Section V provides the performance evaluation of the implemented methods, and Section VI concludes the paper, summarizing the findings.

II. System Model and Problem Definition

In this section, we explain the system model used in this study, detailing its structure and main components. The proposed model is designed to handle VPS in dynamic environments by integrating constraints such as operational time limits, required rest periods, location revisit intervals, and emergency response requirements.

The system represents the patrol area as a graph, where nodes correspond to locations requiring patrols, and edges denote the travel paths between them. Vehicles are assigned to patrol routes while optimizing coverage efficiency, ensuring timely responses to emergencies, and maintaining operational feasibility under predefined constraints. To achieve this, our model incorporates real-time adaptability, allowing schedules to be dynamically updated in response to changing conditions, such as emergency demands or traffic variations. This structure enables scalable and efficient patrol planning, ensuring both broad coverage and rapid incident response in complex urban environments.

We conceptualize the operational environment for a fleet of patrol vehicles as a weighted directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ represents the set of nodes, which includes a depot $l \in \mathcal{N} = 0$, and a set of locations $l = \{1, 2, ..., |\mathcal{N}|\}$ that must be patrolled regularly. Each location has a minimum patrolling time $\mathcal{T}^{patrol}$ and a maximum allowable gap between visits $\mathcal{T}^{gap}$. The set $\mathcal{E}$ signifies the links representing possible routes between these nodes, with weights corresponding to travel time between locations, denoted as $\mathcal{W}_{l,l'}$, which are measured based on current traffic conditions. These weights are dynamically updated to reflect the real-world nature of the problem. During the operational period, vehicles in the fleet ($c \in \mathcal{C}$) are systematically assigned routes within the graph. Each route begins at the depot, visits designated locations, and returns to the depot upon completing its shift. The complexity of this routing lies in the possibility of revisiting certain locations multiple times, as the vehicle may have sufficient time during its shift to conduct additional patrols, ensuring continuous safety and optimal coverage. Each vehicle must adhere to a mandatory rest time $\mathcal{T}^{rest}$ after a maximum operation time of $\mathcal{T}^{shift}$. This structure ensures comprehensive coverage of all areas of interest during operational time.

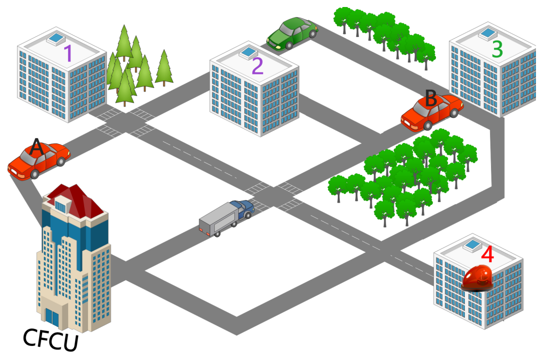


Fig. 1: System model. The environment is a graph with numbered, color-coded nodes (patrol locations) and a depot (start/end). Edges represent travel paths with time weights, while the CFCU monitors routes, dispatches vehicles, and updates schedules in real time.

To manage the entire patrol system, we have established the Central Fleet Control Unit (CFCU) at the depot. The CFCU oversees all vehicles, updates their schedules in real-time, and dispatches them to handle emergencies as they arise. By analyzing live data such as vehicle locations, travel times, and emergency requests, the CFCU ensures that the patrol system can adapt to the ever-changing urban landscape while providing effective coverage and response times. Our goal is to improve the total number of locations visited during the workday to improve coverage. To accomplish this, we ensure that no more than one vehicle is assigned to the same location at the same time and that each location is revisited within a designated timeframe to ensure ongoing safety. There is a likelihood of emergency situations (e.g., robbery) occurring at specific locations during shifts. In such cases, the nearest vehicle must be dispatched to the emergency site, necessitating a complete reorganization of that vehicle's schedule and the entire fleet.

As illustrated in Fig. 1, we shall consider a scenario in which Vehicle B is currently patrolling Location 3, while Vehicle A is en route to patrol Locations 1 or 2. Location 4, which was previously patrolled by Vehicle B, experiences an emergency situation requiring immediate attention. In this case, the CFCU must dynamically reassign a vehicle to address the emergency at Location 4. Based on real-time data, such as the current positions of the vehicles and their travel times to Location 4, the CFCU evaluates the optimal assignment. It may direct Vehicle B to interrupt its ongoing patrol at Location 3 and proceed to Location 4, or alternatively, it could dispatch Vehicle A to Location 4. To ensure efficient emergency handling, the CFCU simultaneously reorganizes the routes and schedules of the entire fleet, balancing the need to respond promptly to the incident with maintaining overall patrol coverage.

To illustrate the computational complexity of the VPS problem, we compare it to the well-known Traveling Salesman Problem (TSP), which seeks a minimal Hamiltonian cycle visiting each city exactly once. While VPS shares structural similarities with TSP, it introduces additional complexities such as flexible visitation numbers influenced by operational constraints. The possibility of a polynomial-time reduction from TSP to VPS underscores the inherent computational demand of VPS. This complexity necessitates advanced algorithmic solutions, particularly for dynamic patrol scheduling and route optimization. In subsequent sections, we develop robust strategies to address these challenges and achieve near-optimal patrol plans.

III. The Heuristic Method: AHBPS2

In this section, we present the Adaptive Hill-Climbing-Based Patrol Scheduling (AHBPS2) algorithm (see Alg. 1), which aims to effectively manage vehicle routes while ensuring a quick response to emergencies. The algorithm works by iterating through scheduled shifts and utilizing a hill-climbing approach to optimize patrol routes, thus guaranteeing thorough area coverage and swift emergency response in dynamic settings. The inputs for AHBPS2 consist of the graph $\mathcal{G}(\mathcal{N},\mathcal{E})$, the vehicle fleet $\mathcal{C}$, the depot location $\mathcal{D}$, the set of shifts $\mathcal{S}$, the maximum operation time per shift $\mathcal{T}^{\text{shift}}$, the minimum patrol time $\mathcal{T}^{\text{patrol}}$, the travel times $\mathcal{W}_{l,l'}$, and the probability of an emergency $\mathcal{P}^{\text{emergency}}$. The algorithm outputs the set of visited locations, the routes for each vehicle during each shift, and a record of handled emergencies.

The algorithm begins by initializing key parameters, including the shift start time ($\mathcal{SST}$), shift end time ($\mathcal{SET}$), current time ($\mathcal{CT}$), a list of visited locations ($Visited_{locations}$), and a data structure to track each vehicle's route ($routes$) in Line 1. For each shift $s \in \mathcal{S}$, the algorithm sets the shift's start and end times (Line 3). A while-loop (Line 4) continues running until the current time $\mathcal{CT}$ reaches the shift end time $\mathcal{SET}$.

Within each iteration of the while-loop, the algorithm evaluates every vehicle $c \in \mathcal{C}$ (Line 5). For each vehicle, it first checks for emergencies based on a given probability (Line 6), simulating potential urgent situations that might occur during patrol operations. If emergencies are detected (Line 7), vehicle states and patrol routes are immediately adjusted, ensuring rapid and effective emergency responses (Line 8). When a vehicle is available and not actively handling an emergency (Line 9), the algorithm selects the next patrol location based on the current simulation time, the vehicle's current state, operational constraints, and the predetermined patrol sequence (Line 10). If a vehicle is handling an emergency, its next location is immediately set to the emergency location, overriding regular patrol planning. The vehicle's position is updated to this new location (Line 11), recorded in the visited locations list (Line 12), and appended to the vehicle's route for the current shift (Line 13).

Algorithm 1: Adaptive Hill-Climbing-Based Patrol Scheduling (AHBPS2)

```
   Input: $\mathcal{G}(\mathcal{N},\mathcal{E})$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{S}$, $\mathcal{T}^{shift}$, $\mathcal{T}^{patrol}$, $\mathcal{W}_{l,l'}$, $\mathcal{P}^{\text{emergency}}$
   Output: $Visited_{locations}$, $routes$, $emergencies$
1  $\mathcal{SET}$, $\mathcal{SST}$, $\mathcal{CT}$, $Visited_{locations}$, $routes \leftarrow \emptyset$;
2  foreach $s \in \mathcal{S}$ do
3      Set start time $\mathcal{SST}$ and end time $\mathcal{SET}$ for $s$;
4      while $\mathcal{CT} < \mathcal{SET}$ do
5          foreach $c \in \mathcal{C}$ do
6              $emergencies \leftarrow$
                 CheckForEmergencies($\mathcal{C}, \mathcal{P}^{\text{emergency}}$);
7              if emergencies are not empty then
8                  Update vehicle states by calling
                     HandleEmergencies($emergencies, \mathcal{C}$);
9              end
10             if $c$ is available and no active emergency then
11                 $next\_location \leftarrow$
                     PlanNextLocation($c, \mathcal{CT}$);
12                 Update $c$'s location to $next\_location$;
13                 Add $next\_location$ to $Visited_{locations}$;
14                 Append $next\_location$ to $routes[s][c]$;
15             end
16         end
17         Apply hill-climbing optimization;
18         Update $\mathcal{CT}$ for the next iteration;
19     end
20 end
21 return $Visited_{locations}$, $routes$, $emergencies$;
```

After processing all vehicles in each iteration, hill-climbing optimization is applied to refine the current patrol routes for improved coverage efficiency (Line 14). The current time $\mathcal{CT}$ is then incremented to proceed to the next iteration (Line 15). Once the while-loop has finished for all shifts, the algorithm outputs the full list of visited locations, the final routes for each vehicle per shift, and the emergency logs (Line 14). These integrated mechanisms ensure the algorithm remains adaptive, balancing efficient routine patrol coverage with real-time responsiveness to dynamic emergency scenarios.

IV. The Reinforcement Learning Approach: $\mathcal{P}^4\mathcal{O}$

To enhance the performance of the AHBPS2 algorithm and to adapt to the dynamic nature of the environment, we propose an RL approach called Patrol Planning with Proximal Policy Optimization algorithm ($\mathcal{P}^4\mathcal{O}$). We frame the VPS problem as an MDP, which enables us to clearly define the states, actions, transitions, rewards, and objectives within the patrol scheduling framework. The MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \gamma, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ the set of actions, γ the discount factor, $\mathcal{P}$ the state transition probability function, and $\mathcal{R}$ the reward function. The functions for state, action, and reward within the MDP components of our problem are outlined as follows:

Agent: The agent is CFCU, situated at the depot. Its main responsibility is to determine the best routes for vehicles and the locations each vehicle needs to visit. Additionally, it must reorganize the fleet in emergencies and dispatch the nearest vehicle(s) to secure high-risk locations, regardless of the circumstances.

State: The state space encompasses the various situations or conditions the agent must evaluate for each vehicle. A state is characterized by five key attributes: (1) the set of locations each vehicle has visited up to timestep t, (2) the current shift number, which indicates how many shifts each vehicle has completed, (3) the emergency status, highlighting location(s) that need immediate attention, (4) the last visit time for each location, and (5) the remaining shift time for each vehicle, which can vary since vehicles may be dispatched at different times by the agent. This state space combines both discrete and continuous elements. The discrete components include the set of visited locations, the shift number, and the emergency status, while the continuous components consist of the remaining shift time and the last visit time for locations. This mixed representation effectively captures the dynamic and varied nature of the patrol scheduling problem, allowing the agent to make well-informed decisions based on both categorical and numerical data.

Actions: The actions available to the agent involve selecting the next location for a vehicle to visit based on its current state. The options are restricted to locations that are accessible and not blocked. This action space is discrete, as it requires choosing a location from a defined set of available options.

Reward: The reward function is designed to prioritize patrolling a greater number of locations. Visiting a location yields a small positive reward (+0.5) to encourage increased coverage. Conversely, if an emergency occurs and the agent fails to dispatch a vehicle within the allowable response time, which is determined by the travel time from the nearest vehicle's position to the emergency location, a substantial penalty (-50) is imposed.

By framing the problem within the MDP framework, the agent learns how to effectively assign vehicles to various locations in the environment to maximize total visits. This is achieved by taking actions that lead to more advantageous states, thereby accumulating higher rewards. The resulting policy will outline optimal routes for each vehicle, ensuring they can respond to emergencies efficiently.

The details of the $\mathcal{P}^4\mathcal{O}$ algorithm are outlined in Alg. 2. The process starts with the initialization phase, where the environment is set up with the network graph $\mathcal{G}$, the vehicle set $\mathcal{V}$, and temporal constraints such as the maximum shift duration $\mathcal{T}^{shift}$, the required rest period $\mathcal{T}^{rest}$, and the minimum patrol time $\mathcal{T}^{patrol}$. Additionally, the travel time matrix $\mathcal{W}_{l,l'}$ and the emergency probability $\mathcal{P}^{emergency}$ are established (Line 1). Next, in Line 2, the $\mathcal{P}^4\mathcal{O}$ agent is initialized using an actor-critic network. Training occurs over $\mathcal{K}$ episodes, which are chosen based on empirical methods to ensure convergence while maintaining computational efficiency (Line 3). At the beginning of each episode, the environment parameters are reset in Line 4, and the cumulative reward $\mathcal{R}_{episode}$ is set to zero (Line 5).

In each episode, the algorithm goes through decision steps until all shifts are completed as indicated in Line 6. For every vehicle $v \in \mathcal{V}$ (Line 7), the current state s_v is observed in Line 8, and the next location a_v is sampled from the policy $\pi_\theta(s_v)$ in Line 9. If an emergency event occurs with

Algorithm 2: Patrol Planning with Proximal Policy Optimization ($\mathcal{P}^4\mathcal{O}$)

```
Input: $\mathcal{G} = (\mathcal{N}, \mathcal{E})$: Network graph with nodes $\mathcal{N}$ (locations) and
       edge $\mathcal{E}$; $\mathcal{V}$: Set of vehicles; $\mathcal{T}^{shift}$: Maximum shift duration;
       $\mathcal{T}^{rest}$: Required rest period between shifts; $\mathcal{T}^{patrol}$:
       Minimum patrol time per location; $\mathcal{P}^{emergency}$: Probability
       of emergency; $\mathcal{W}_{l,l'}$: Travel time matrix between locations
Output: Optimized vehicle routing policy $\pi_\theta$
1  Initialize $\mathcal{P}^4\mathcal{O}$ Environment with the given inputs;
2  Initialize $\mathcal{P}^4\mathcal{O}$ Agent with an actor-critic network;
3  for episode $= 1$ to $\mathcal{K}$ do
4      Reset environment parameters;
5      $\mathcal{R}_{episode} \leftarrow 0$;
6      while not all shifts completed do
7          foreach vehicle $v \in \mathcal{V}$ do
8              Observe current state $s_v$;
9              Sample next location $a_v \sim \pi_\theta(s_v)$;
10             if an emergency event occurs with probability
                 $\mathcal{P}^{emergency}$ then
11                 Identify emergency location $l_e$;
12                 foreach vehicle $v \in \mathcal{V}$ do
13                     if $v$ has remaining shift time then
14                         $last\_location \leftarrow$ last visited location
                             of $v$ (or starting point if none);
15                         $travel\_time \leftarrow \mathcal{W}(last\_location, l_e)$;
16                         Subtract $travel\_time$ from $v$'s
                             remaining shift time;
17                         if remaining shift time $\leq 0$ then
18                             Move $v$ to the next shift and
                                 subtract the rest period $\mathcal{T}^{rest}$;
19                         end
20                         Append $l_e$ to $v$'s route;
21                     end
22                 end
23                 continue to the next decision step;
24             end
25             Execute action $a_v$ subject to all operational
                 constraints;
26             if revisit time constraint is violated then
27                 Apply a penalty to the reward;
28             end
29             Observe reward $r_v$;
30             $\mathcal{R}_{episode} \leftarrow \mathcal{R}_{episode} + r_v$;
31         end
32         Compute returns $\mathcal{G}_t$ using discounted rewards;
33         Estimate advantages $\mathcal{A}_t$;
34         Update policy $\pi_\theta$ using the clipped objective;
35     end
36 end
37 return $\pi_\theta$;
```

a probability of $\mathcal{P}^{emergency}$ (Line 10), an emergency location l_e is identified in Line 11, and a replanning procedure is initiated for all vehicles (Lines 12-19). Specifically, for each vehicle that has remaining shift time (Line 13), the algorithm determines its last visited location (or starting point if none exists) in Line 14, calculates the travel time to l_e using $\mathcal{W}(last\_location, l_e)$ in Line 15, and deducts this travel time from the vehicle's remaining shift time (Line 16). If the remaining shift time becomes non-positive, the vehicle moves to the next shift with the rest period $\mathcal{T}^{rest}$ subtracted, and l_e is added to its route (Lines 17 and 18). When a vehicle's remaining shift time is exhausted while responding to an emergency, it immediately transitions to the next shift after deducting the mandatory rest period, thereby upholding operational constraints. Simultaneously, the emergency location is appended to its route, ensuring that urgent incidents are addressed without compromising regulatory requirements. This transition indicates that the current shift has been fully utilized and requires the vehicle to take its necessary rest before resuming operations. The algorithm then proceeds to the next decision step in Line 20. In the absence of an emergency, each vehicle's next location is determined by sampling from the policy $\pi_\theta(s_v)$, where s_v represents the current state and the policy network produces a probability distribution over all candidate locations. Once the action a_v is selected, the vehicle proceeds by updating its route, deducting the corresponding travel and on-site service times from its remaining shift duration, while strictly enforcing operational constraints such as the minimum revisit interval (with penalties for violations) and the mandatory rest period between shifts, which prompts a shift transition when the available time is exhausted (Line 25). The algorithm checks if the revisit time constraint is violated, and if it is, a penalty is applied to the reward in Lines 26 and 27 (the penalty for this violation is -50). This reward structure encourages the agent to avoid significant penalties. The immediate reward r_v, which accounts for both positive outcomes and penalties incurred, is observed and added to $\mathcal{R}_{episode}$ (Lines 29 and 30).

After processing the vehicles at the decision step, the agent calculates the returns $\mathcal{G}_t$ using discounted rewards ($\mathcal{R}_{episode}$) and estimates the advantages $\mathcal{A}_t$, which measure the relative benefit of the selected actions. In this context, $\mathcal{G}_t$ signifies the cumulative sum of discounted future rewards from the current time step onward, reflecting the expected long-term return (Lines 32 and 33). The policy is then updated using the PPO clipped objective, a crucial aspect of the PPO algorithm, which restricts the policy update by clipping the ratio of new to old action probabilities. The combination of $\mathcal{G}_t$ and $\mathcal{A}_t$ is utilized to compute a surrogate loss that is maximized during the update process, promoting stable and conservative learning as noted in Line 34. Ultimately, after $\mathcal{K}$ episodes, the algorithm produces the optimized routing policy π_θ, which effectively manages revisit constraints, emergency situations, and overall route coverage.

The $\mathcal{P}^4\mathcal{O}$ simulation employs the ReLU activation function and the Adam optimizer. The policy update clipping parameter is set to 0.18, while the discount factor γ is 0.97, and the bias-variance trade-off parameter λ is 0.95. The learning rate α is fixed at 5e-4. Training is conducted over 5000 episodes ($\mathcal{K}$), with a maximum of 500 steps per episode. The neural network consists of four hidden layers, each with 128 neurons. Policy updates are performed over four epochs per iteration. We chose PPO because it effectively addresses the dynamic challenges of the VPS problem, striking a balance between stability and adaptability in environments with fluctuating routes and unexpected emergencies. Its capability to handle both continuous and discrete states and actions, along with neural network-based policy optimization and a clipping mechanism for stable training, makes it particularly well-suited for this task. PPO is notable for its efficiency in learning from limited interactions while maintaining robustness across various strategic scenarios and hyperparameter settings, requiring minimal tuning.

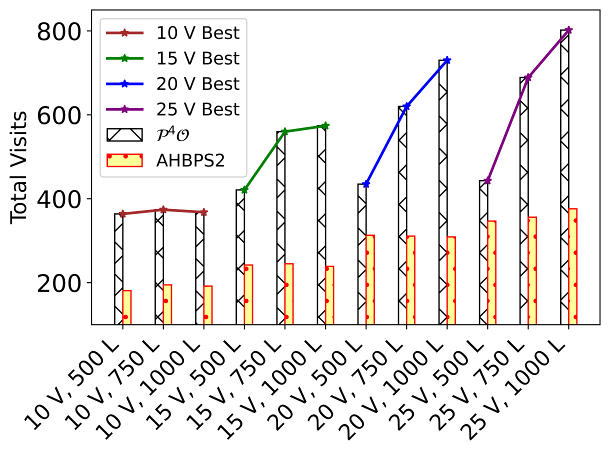
V. Performance Evaluation

In this section, we compare the performance of the proposed methods, $\mathcal{P}^4\mathcal{O}$ and AHBPS2, across a variety of scenarios, including tests on large-scale graphs representing complex urban networks. The comparison is carried out under different scenarios, using varying fleet sizes, represented by the number of vehicles (V = 10, 15, 20, 25), and different scales of the patrol environment, defined by the number of locations (L = 500, 750, 1000). In the scenarios, V represents vehicles and L represents locations. Our evaluation focuses on dynamic environments, assessing their performance regarding total visits and emergency handling success rate.

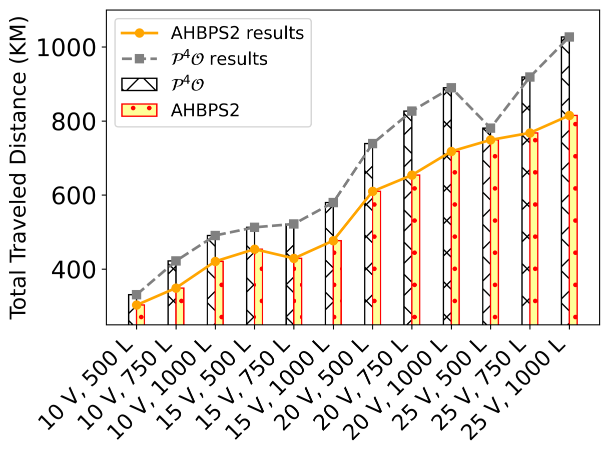
The graphs used for testing were generated using real-world maps extracted from Open Source Maps (OSM). By assigning varying numbers of nodes to represent target locations (e.g., banks) on the same map size, we created diverse graph instances with different configurations. This allowed us to simulate a variety of real-world scenarios to test the three approaches comprehensively. Using the NetworkX and OSMNX libraries in Python, we processed the extracted maps and converted them into graph structures to serve as inputs for the algorithms. We use Python3 to simulate the operation of our algorithms. We run our program on AMD Ryzen 7 7700X, 8 cores, and 16 Threads, amplified by NVIDIA GeForce RTX 4070 Super GPU, and 64.0 GB RAM.

The travel times between nodes are estimated based on distances extracted from the generated maps. A normal distribution with a mean of 15 minutes and a standard deviation of 5 minutes is considered to reflect typical urban travel time variability, which was also observed with most of the examined maps. To introduce realistic fluctuations in traffic conditions, we apply a stochastic adjustment of ± 2 minutes in 25% of the runs, ensuring that the model accounts for varying traffic densities without introducing excessive unpredictability. This accounts for scenarios like low traffic (reduced travel time) or unforeseen incidents (increased travel time). This methodology effectively mitigates randomness, providing a clearer, more accurate reflection of algorithm performance across varying conditions and underscoring the efficiency of the methods. Initially, we fix certain parameters while testing various instances, and after analyzing the results, we adjust inputs to assess their impact. Unless specified otherwise, the rest time is set to 10 minutes, patrol time per location is 5 minutes, and the travel time distribution maintains a mean of 15 minutes. The results are averaged over 1000 runs for AHBPS2, and over ten runs for $\mathcal{P}^4\mathcal{O}$.

Fig. 2(a) shows the performance of the proposed methods under the mentioned scenarios. As illustrated, $\mathcal{P}^4\mathcal{O}$ consistently outperforms AHBPS2 by great differences, as expected in all scenarios. This is because $\mathcal{P}^4\mathcal{O}$ can adaptively learn and optimize through trial and error in a way that AHBPS2, which relies on predefined rules, cannot. $\mathcal{P}^4\mathcal{O}$ proves its efficiency over all sorts of networks. For instance, in the scenario with



(a)



(b)

Fig. 2: Comparing the (a) total visits & (b) traveled distance of $\mathcal{P}^4\mathcal{O}$ and AHBPS2 under the mentioned scenarios.

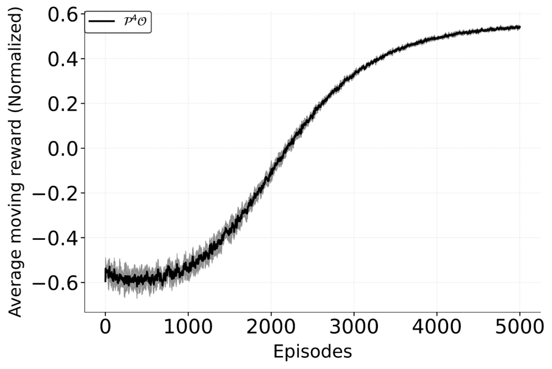


Fig. 3: Training progress for $\mathcal{P}^4\mathcal{O}$

25V, 750L, $\mathcal{P}^4\mathcal{O}$ covered 689 locations safely out of 750, which is around 92% coverage, and can be utilized and adjusted by companies based on their constraints to meet their demand. These results emphasize how adaptive, data-driven methods capable of iterative learning are better equipped to exploit the added flexibility presented by more extensive and variable networks. Consequently, these findings further confirm the suitability of $\mathcal{P}^4\mathcal{O}$ for real-world applications, where operational scales and coverage requirements are likely to expand over time.

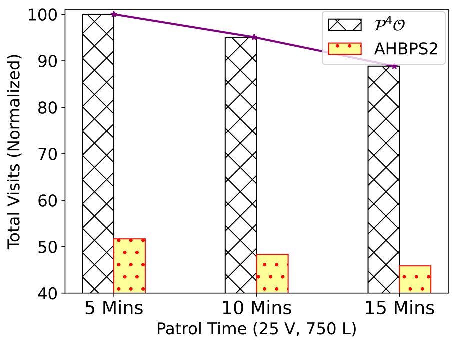
Fig. 3 shows the average reward received by the $\mathcal{P}^4\mathcal{O}$'s agent throughout 3000 episodes. As can be noticed, initially, due to exploration, the agent incurs large negative rewards; however, it eventually discovers an optimal patrolling plan, and the reward stabilizes after approximately 200 episodes. The shaded area around the reward shows the standard deviation. As episodes progress, the uncertainty (standard deviation) tends to diminish.

Fig. 2(b) presents the total traveled distance (in KM) across different scenarios. As shown, an increase in the number of vehicles or locations results in greater travel distances. This trend arises for two reasons: first, a larger fleet size leads to more vehicles actively patrolling, thereby increasing the total distance covered; second, a higher number of locations expands the patrol area, providing more destinations for vehicles to visit. Consequently, the overall traveled distance increases as both variables grow. The trends of each algorithm over the scenarios are also shown with different colors. For instance, in the scenario of 20V, 1000L, vehicles traveled around 850 km in total across the whole operation time, 12 hours, for $\mathcal{P}^4\mathcal{O}$. On the other hand, AHBPS2 traveled around 615 km. It can be explained because $\mathcal{P}^4\mathcal{O}$ covered more locations, and as a result, vehicles traveled more to cover these locations.

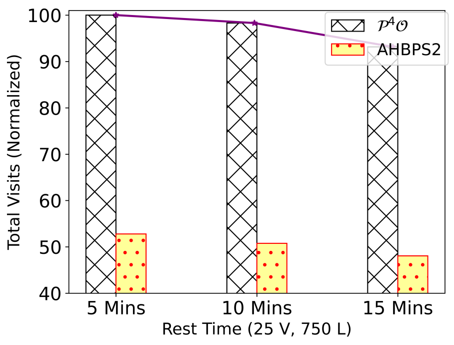
We also analyzed how different parameters affect the objective of maximizing visited locations. We applied $\mathcal{P}^4\mathcal{O}$ to large networks with 25 vehicles and 750 locations, as this scenario achieved the highest coverage of 92% among all examined cases. Specifically, we examined the effects of variations in $\mathcal{T}^{rest}$, $\mathcal{T}^{patrol}$, $\mathcal{P}^{emergency}$, and $\mathcal{W}_{l,l'}$ (road conditions). First, we compare different patrol durations of 10 and 15 minutes against the baseline value of 5 minutes. As expected, longer patrol times result in fewer visited locations, as vehicles spend more time at each location (see Fig. 4(a)).

Next, we assessed the impact of varying rest periods at the depot by adjusting the initial 10 minutes to 5 and 15 minutes (Fig. 4(b)). Reducing the rest time to 5 minutes yields a slight improvement, while increasing it to 15 minutes diminishes performance. However, shorter rest periods may have unexamined psychological effects, so adequate rest is recommended. Lastly, we test different travel time distributions with varying means while keeping the standard deviation at 5. We expect more locations to be visited when the mean travel time is lower, since it takes less time to travel from location l to l'. As shown in Fig. 4(c), shorter travel times indeed lead to increased coverage, which meets our expectations.

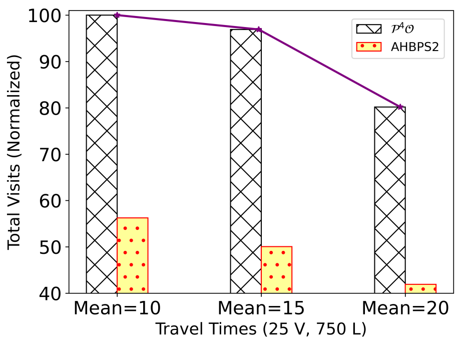
We also evaluated the effects of having different operation times. We compared two different values (10, & 8 hours) for operation time, and compared the results with the base value, 12 hours. As expected, 12 hours has better coverage when it comes to total visits. This is because the fleet has more time to secure locations. Fig 5(a) illustrates these changes. A similar trend is visible for both methods. We also tested our proposed methods with different probabilities of emergencies. During our simulations, the probability of an emergency happening



(a)

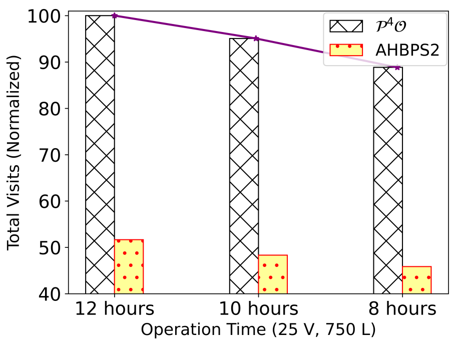


(b)

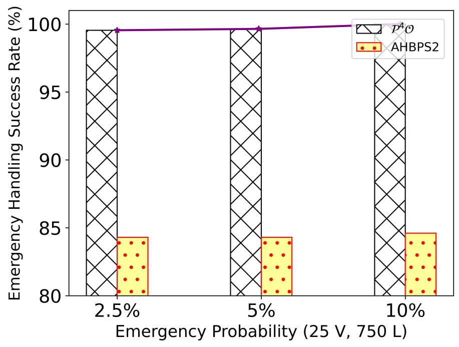


(c)

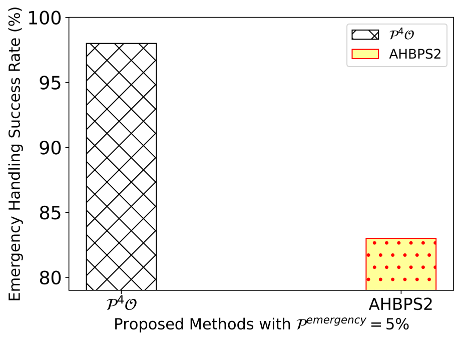
Fig. 4: Total visits comparison for the proposed methods with different (a) patrol time, (b) rest time, and (c) travel time distributions (standard deviation is 5)



(a)



(b)



(c)

Fig. 5: Total visits comparison for the proposed methods with (a) different operation time, (b) different emergency probability (c) Proposed methods emergency handling success rate

($\mathcal{P}^{emergency}$) was 5%. Fig 5(b) illustrates the effects of different values (2.5% and 10%) along with the baseline. As can be seen, the proposed methods performed very similarly across different probabilities. In higher probabilities of emergency, which is the case for areas with high crime rates, our methods performed slightly better, showcasing their abilities in handling emergencies. It must be mentioned that higher probability leads to lower coverage since the patrolling plans change and vehicles must travel to new locations. Fig. 5(c) reports the emergency handling coverage (%) for both methods. $\mathcal{P}^4\mathcal{O}$ effectively handles 98% of the emergency cases happened during the simulations. On the other hand, AHBPS2 performed well, responding in time to around 83% of the emergencies. To conclude, both methods performed very well in handling emergencies. In conclusion, $\mathcal{P}^4\mathcal{O}$ proves to be highly effective in all sorts of networks and different fleet sizes and demonstrates efficiency in real-time applications due to its high coverage and effective emergency handling system.

VI. Conclusion

This study addressed the Vehicle Patrol Scheduling (VPS) problem, crucial for urban safety, by proposing two approaches: the Adaptive Hill-Climbing-Based Patrol Scheduling (AHBPS2) heuristic and the Patrol Planning with Proximal Policy Optimization ($\mathcal{P}^4\mathcal{O}$) reinforcement learning algorithm. While AHBPS2 effectively manages routes and emergency responses across various scales, $\mathcal{P}^4\mathcal{O}$ leverages a Markov Decision Process and Proximal Policy Optimization to learn optimal patrol strategies. Extensive simulations on real-world urban maps demonstrated that $\mathcal{P}^4\mathcal{O}$ consistently outperforms AHBPS2 in terms of total visits and emergency handling. These results provide valuable insights for designing adaptive and efficient patrol scheduling systems tailored to diverse operational needs.

References

[1] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[2] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[3] P. Toth and D. Vigo, *The vehicle routing problem.* SIAM, 2002.

[4] N. Adler, A. S. Hakkert, J. Kornbluth, T. Raviv, and M. Sher, "Location-allocation models for traffic police patrol vehicles on an interurban network," *Annals of operations research*, vol. 221, pp. 9–31, 2014.

[5] R. Bautista Linde, "Deep reinforcement learning for uav patrolling in rescue operations," 2023.

[6] J. Perron, J. Hogan, B. Moulin, J. Berger, and M. Bélanger, "A hybrid approach based on multi-agent geosimulation and reinforcement learning to solve a uav patrolling problem," in *2008 Winter Simulation Conference*. IEEE, 2008, pp. 1259–1267.

[7] G. Kim, Y. S. Ong, T. Cheong, and P. S. Tan, "Solving the dynamic vehicle routing problem under traffic congestion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2367–2380, 2016.