Decentralized reinforcement learning for multi-agent patrol routing

Avijit Roy
Indian Institute of Technology, Kanpur
Kanpur, India
avijit@iitk.ac.in

Nisheeth Srivastava
Indian Institute of Technology, Kanpur
Kanpur, India
nsrivast@iitk.ac.in

ABSTRACT

We model the placement of first-responder patrol vehicles on a city map as a multi-agent reinforcement learning problem, where individual agents learn desirable locations for parking based on dynamically updated geo-localized emergency call records. The model is able to outline reasonable patrol locations and routes, adapting to changes in the geographical pattern of call locations, and permits optimization of routes accommodating fuel economy and other cost-based concerns into account in a principled way. We also present an actual patrolling system we have developed around this model, and present simulated results comparing it's performance vis-a-vis centroid-based patrol location prediction and judgments made by humans.

KEYWORDS

Reinforcement learning, Multi-Agent System, Path Prediction, Human Computer Interaction

ACM Reference Format:
Avijit Roy and Nisheeth Srivastava. 2021. Decentralized reinforcement learning for multi-agent patrol routing . In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), London, UK, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

The urgency of first-response in emergency situations is a critically important aspect of public services. It is a common adage that, when you need the police immediately, they tend to be just five minutes away. Emergency response services (henceforth EMS), therefore, are tasked with responding as quickly to emergencies as possible given existing resources [20]. Since existing resources are increasingly constrained, EMS have to optimize first response resources intelligently [15].

The problem of optimizing first responder resources is most commonly studied in the context of police responses [9]. The need for firefighting services is too sparse in modern society to necessitate proactive placement of resources on the part of fire departments, and the need for emergency medical services is fulfilled by a patchwork of state and private agencies. Policing, in contrast, remains the one activity that, across governments, is controlled entirely by the state, and remains perpetually in high demand.

In the policing context, first responders are usually patrol officers. The placement of patrol officers is therefore, an important topic of study in criminology [10]. Large-scale field studies attest to a substantial deterrence effect from placing patrol officers prominently in known *hot spots* of criminal activity [21, 25].

Researchers commonly address this problem as one concerning the optimal design of patrol routes [5, 22], being in turn a subset of the more general multi-agent patrol routing problem [1]. A common underlying premise in theoretical approaches to this problem is to map the area of interest as a graph, and design walks on the graphs that ensure that all (or important) vertices of the graph are visited frequently [11, 26]. Graph partitioning and network flow optimization approaches are commonly used to solve the problem [8].

However, it is commonly recognized that such deterministic strategies are not good fits for the underlying problem in policing contexts [5]. This is for three reasons. One, deterministic fixing of patrolling schedules permit criminals to reflexively schedule crimes coordinating with the patrol schedules. Two, these approaches are not adaptive to situational changes on the ground affecting the number and distribution of potential first-responders. Three, the solutions proposed by such methods are frequently considered excessive and wasteful by practitioners. In any realistic setting involving motorized patrols, planners want to balance fast response times with managing fleet fuel consumption [31].

To tackle the problem of reflexive crime scheduling, researchers have studied patrol planning as two-player zero-sum games [22, 28]. To tackle the problem of adaptation, heuristic-based probabilistic optimization schemes have been developed, including proposals deploying genetic algorithms [16] and ant colony optimization [13]. Domain-specific heuristics are used for optimizing police patrols in [5] and ambulances in [14]. No approaches, to the best of our knowledge, currently explicitly tackle the problem of managing the trade-off between response times and patrolling resource consumption.

Reinforcement learning appears to be a natural fit for the multi-agent patrol routing problem [27], and has in fact been proposed for this purpose in the literature [24]. However, existing treatments of this problem using reinforcement learning are noted to be complex, and seek to optimize the graph-theoretic notion of patrol optimality that omits several important situational nuances important to real-world EMS requirements [5].

In this paper, we present a decentralized reinforcement learning solution for the multi-agent patrolling problem particularized for the context of policing EMS. Our solution is simpler than previous reinforcement learning solutions for this task, highly scalable with respect to the number of patrolling units, and addresses the three concerns identified above that have bedeviled the adoption of previous proposals in real-world deployments. Notably, we use call

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, London, UK.

records from a real police EMS for designing and testing our system, and present a final solution that is currently being field-tested for adoption within the same real-world EMS.

2 DECENTRALIZED REINFORCEMENT LEARNING FOR MULTI-AGENT PATROL ROUTE PLANNING

Reinforcement learning is a natural setting for the multi-agent patrolling problem, balancing exploration of an unknown environment with performing actions assigned high rewards by the system designer. The notion of states, actions and rewards maps easily to geographical locations, patrol movement and emergency calls in this problem domain. Traffic congestion adds dynamic uncertainty to the state transition probability, and the varying criticality of calls received from different locations make it more realistic to treat rewards as unknown. Thus, it seems reasonable to model the multi-agent patrolling problem using reinforcement learning.

Ideally, we would like multiple agents to coordinate their activity in jointly beneficial ways. A centralized view of the multi-agent problem as a single agent with multiple actuators is attractive for its theoretical simplicity, but is practically infeasible because of the exponential growth in the state space with the number of patrolling units [3]. Sophisticated approaches for finding globally optimal solutions for a cooperative group of agents are known, but are infeasible because of quadratic growth in the number of communications needed to coordinate an increasing number of patrolling units [2].

Thus, realistically, we are restricted to solving the problem in a decentralized manner, considering each of the patrolling units to be 'independent learners' [6]. Notably, it is known that for a wide variety of settings, independent learners tend to do just as well or even better than more coordinated 'joint action learners' [6].

The use of independent learners in the multi-agent setting introduces both theoretical and practical difficulties [19]. In theory, since the action of any agent in the previous time step can influence any other agent's action in the next time step, the system cannot be described as Markovian in the true sense. In a practical sense, this problem boils down to how to get agents to identify local policies that don't cause them to get in other agents' way.

Theoretical analyses of these problems have revealed that the difficulties are not as formidable as they may seem. Claus & Boutilier have reported that independent learners fare no worse than joint action learners in terms of convergence to the globally optimal solution [6]. In Lauer & Riedmiller's optimistic distributed Q-learning algorithm, all penalties due to non-coordination of agents are ignored in each individual agent's update. Their algorithm still finds the global optima in deterministic cooperative multi-agent systems [17] and, with minor alterations, does quite well in stochastic settings also [18]. Additionally, practical applications of decentralized reinforcement learning simply update each agent's policy using Q-learning assuming the other agents don't exist, and still achieve reasonable performance [7].

Thus, while the theoretical problems facing decentralized reinforcement learning remain formidable, cumulative experience shows the existence of considerable emergent self-coordination in even the simplest practical implementations, a fact beginning to find support in theory as well [30]. We, for our part, attempt to solve the coordination problem by embedding a proximity penalty for our independently learning agents, as we describe further below.

3 IMPLEMENTATION

In this section we describe the actual implementation of our approach. At the very outset, it is important to provide background into the situational grounding of our project. We have designed the system described below specifically to assist in planning patrol vehicle locations for a fleet of about 3600 patrol vehicles possessed by one of the largest police departments under a single command in the world, responsible for providing emergency first response services for a very large population. The emergency dispatch unit of this department handles upwards of 50000 calls per day, resulting in approximately 5000 effective patrol vehicle dispatch requests per day. The vehicles in this fleet currently move an average of 20 kms per vehicle per day, resulting in a fuel cost of approximately $ 105 million per year (purchasing power parity adjusted to 2015 USD), and average response times of approximately 20 minutes per emergency request.

Our problem statement was to use geo-localized information about the source of emergency calls from the past to identify hourly patrol points for the entire fleet of vehicles, allowing officers discretion to change these points based on their ground experience, and allowing administrators flexibility in emphasizing or de-emphasizing fuel costs while determining patrolling points. For ease of exposition, we restrict our analysis and results below to one district in the entire state. This restriction does not change our results in any material way, since the police EMS is also administratively partitioned by districts, such that each district's patrol routing is performed independently.

Emergency police services operate round the clock in the district in question. A fleet of 25 patrol vehicles move around the district and when some emergency call comes to the control room, they attend to the case. These vehicles should ideally always be near to the crime-prone areas and they should cover as much area as possible with minimal movement. Movement of these emergency vehicles incur fuel cost, so moving all the vehicles all the time is undesirable, though this is the primary optimality criterion several earlier approaches have sought to optimize [11, 24].

Our partner police EMS made available to us a unique dataset, containing call records for all calls received by the EMS over the past three months in the state. The data contains descriptions of alleged crimes, as assessed by dispatchers, spatial and temporal coordinates of the call, and additional textual descriptions of events and sensitive meta-data. As we mention above, this paper focuses on subset of this data, corresponding to $\approx 200k$ calls received in one district over three months, leading to approximately 100 dispatch events per day within the district.

They also made available to us the vehicular locations of their patrolling fleets for the same time duration. Since all their patrolling vehicles are GPS-enabled, this information is available in real-time to the EMS.

4 MODEL

To generate the route of the vehicles, vehicles must decide which direction the vehicle should move in the next timestamp. Patrolling points are automatically generated from the hourly location of the vehicles obtained as part of the route. The movement of the vehicle is directed by the reward it receives from each step. Reward is dependent on two factors. First, if the vehicle reaches a crime zone quickly, then that will produce a high reward. Second, keeping distance from other vehicles incur rewards, more the distance, higher the reward. Below we formalize the model:

We use a Markov Decision Process (MDP) to represent the model. In Markov Decision Process, the next state is only dependent on the current state and the action taken at that state. That is, at time t,for state s, reward r and action a:

$$Pr\{R_{t+1} = r, S_{t+1} = s^{'} | S_t, A_t\}$$

$$= Pr\{R_{t+1} = r, S_{t+1} = s^{'} | S_0, A_0, R_1, ..., R_t, S_t, A_t\}$$

We model our agents, states, actions and rewards as follows:

4.1 Agent

Each vehicle is represented by an agent. So, for v vehicles, the agents will be denoted as, $Agent \in \{0, 1, 2, ..., v-1\}$

4.2 State

We divide the entire area into a rectangular grid of size $M \times N$.Let us consider the state space $S = \{s_0, s_1, s_2, ...\}$, where $s_i \in \{0, M\} \times \{0, N\}$ denoting the location of an agent in the 2-D grid-world.

4.3 Action

For each state s, action $a \in \mathcal{A}(s) = \{L, R, U, D, S\}$, where, L = Left, R = Right, U = Up, D = Down, S = Same. If taking an action takes the agent out of the grid-world, then that action is not allowed.

4.4 Reward

We design our reward keeping in mind three factors. First, agents need to be close to the crime location. Second, agents need to keep distance among themselves. Third, movement is costly. So the total reward is defined as,

$$r(S_t) = r^p(S_t) + r^i(S_t) - C$$

Here, $r(S_t)$ is total reward, C is a per step movement penalty that is zero if the agent remains in the same state on the next turn and a fixed positive value α otherwise, $r^p(S_t)$ is proximity reward, and $r^i(S_t)$ is instant reward.

Proximity Reward: To promote distance among the agents, we use this part of the reward

$$r^p(S_t) = e^{\beta_1 \times d}$$

. Here, β_1 is a parameter controlling the reward magnitude, and d is the sum of the euclidean distance of this agent from other agents.

Instant Reward: This part of the reward function is to help the agent find states which have crime locations.

$$r^i(S_t) = \beta_2^{\tau} \times reward(S_t)$$

Where, β_2 is a parameter manipulating reward magnitude, τ is time elapsed since a call was recorded from state S_t, and $reward(S_{t-\tau})$ is the reward available at the state because of a crime committed at time $t - \tau$. Calls arise randomly from different states. A reward is available at state S_t when crime occurs at that state. and the reward decays exponentially with time to zero because of the effect of β_2. The decay in the instant reward incentivizes faster responses to call locations.

4.5 Algorithm

Algorithm 1: Q-learning (off-policy TD control)

Algorithm parameters: step size $\alpha \in (0, 1]$, $\epsilon = 0.4$;
Initialize $Q(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily;
foreach *episode* **do**
  Initialize S;
  foreach *step of episode* **do**
    Choose A from S using policy derived from Q (e.g., ε-greedy);
    Take action A, observe R, S';
    New $Q(S, A) \leftarrow$
     $Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$;
    $S \leftarrow S'$;
  end foreach
end foreach

We use simple Q-Learning to solve the MDP [29], as schematized in Algorithm 1. Consider $Q^*(s, a)$ as the expected value of taking action a from state s and then following the optimal policy. $Q(s, a)$ denotes the current estimate of $Q^*(s, a)$.

We initialize the Q-table with all 0. We create an $(m \times n)$ grid-world to keep track of the environment. We initialize random positions for different vehicles (0 to $v - 1$).

We choose actions using an ϵ-greedy strategy. We choose the value of the ϵ as 0.4 by trial and error. If there is a draw in Q-values, we choose the action uniformly randomly among the drawn candidates.

We then take the step according to the action chosen. We update the new location of the agent. Then we calculate the reward.

Finally we update the Q-Table using the following equation:

$$NewQ(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$$

5 SYSTEM DESCRIPTION

The final version of this project will link to a live deployment of the system we have created. We describe it using text and screenshots. Figure 1 provides an overview of the application.

Under the hood, we divide the entire area covered by the EMS into district-wise zones, approximately identified by rectangular bounding boxes. We create routes for different zones separately by running the algorithm once for each zone. A web-based application allows officers to monitor and update vehicle routes. We built the web-application on the MERN stack to interact with the model. We use human interaction to improve the routes over time, in the sense that our algorithm replaces its own selected patrol point with a human-generated patrol point stored in the database if the human

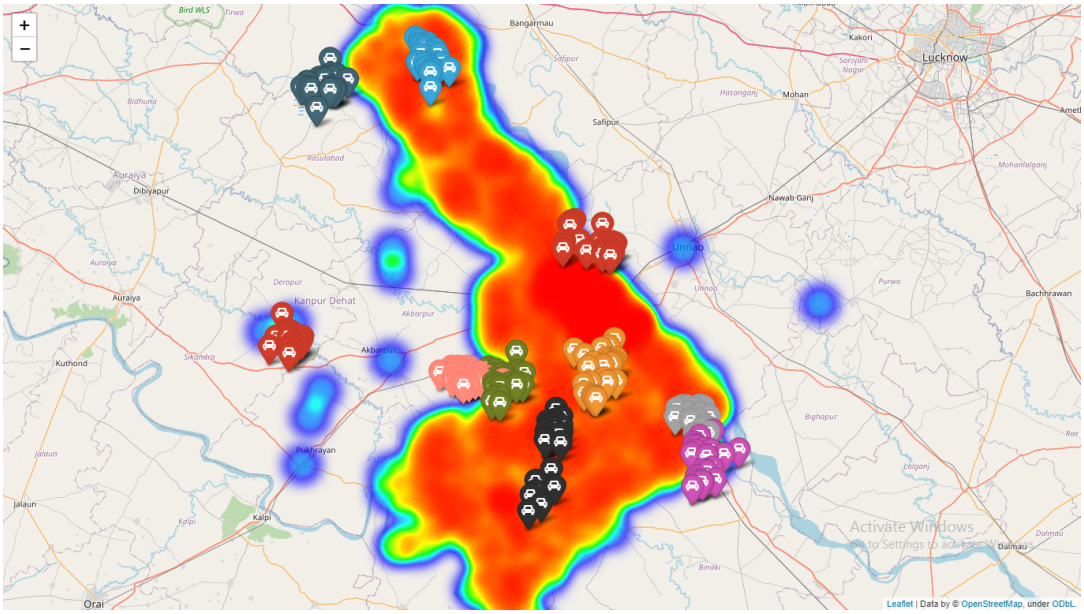


Figure 1: This figure show an overview of our Application. The markers shows locations of vehicles at different points in time. The heatmap reports density of call records at different locations in the map.

suggestion is within a certain radius of the algorithmically proposed point.

We initially generate a route in Python using the algorithm mentioned above. We save the generated route in the database. Then we show the route and take user feedback using the web-application. The following are the main components of the application.

5.1 Database

We use MongoDB for this application. We have created the following collections:

5.1.1 Zones. This collection has the information about all the zones that the total area is divided into. The total area is divided into rectangular zones. The fields of this collection are:

- Name: Name of the zone
- Zone: Unique Identifier for Zone
- LeftLat: Latitude of the top left corner
- LeftLng: Longitude of the top left corner
- RightLat: Latitude of the bottom right corner
- RightLng: Longitude of the bottom right corner

5.1.2 Crimes. This collection stores the crimes occurring at different locations. The fields in this collection are:

- Type: Describes the type of the crime
- Zone: Zone Id for zone in which the crime occurred.
- Lat: Latitude of crime location.
- Lng: Longitude of crime location.

5.1.3 Vehicles.

- VehicleId: Unique Identification number for each vehicle
- Zone: Zone Id of the vehicle.
- locations: An array containing all the points in the route of the vehicle

5.1.4 NearestLocations.

- Zone: Zone of the location
- Location: Locations given by the users during interaction in GeoJSon format.

5.2 Server

We have built the server using NodeJS and Express. The task of the server is to connect and serve required data from database to the client. In the server we implement the following main features.

5.2.1 Authentication. We only allow authentic users to see and update the data. We use token based authentication to authenticate a user via username and password.

5.2.2 Zone Filtering. Every user is associated with a zone. We filter all the data based on the zone of the user logged in and show it to the user. This way we separate concerns of every zone.

5.3 Client

In this part we implement our User Interface using ReactJS and Bootstrap. The main features of this layer are as follows:

5.3.1 All Vehicles View. In this view we show the current location of all the vehicles. It helps the control room to redirect the nearest vehicle to the crime location.

We also provide option to show Heat-Map for crimes occurring at different locations. It shows the frequency of crime at the entire zone.

5.3.2 Single Vehicle View. On clicking a vehicle in All Vehicles View, we redirect to Single Vehicle View. In this view we show the route for 24 hours. We show what the vehicle's position will be at each hour.

User can update each location in the route to a more suitable location. This location will be saved in the NearestLocations collection in the database and will be used next time to generate the route, complementing the algorithm's judgment with human judgment.

6 EXPERIMENTAL EVALUATION

We present two sets of results using our algorithm. The first set of results are generated *in silico* and the second set is generated in a real-world environment. The *in silico* evaluation seeks to examine whether the individual agents are jointly learning policies that are globally reasonable given the domain. The real-world evaluations attempt to discern whether the output of the system is realistically usable by the EMS.

6.1 Simulation Based Results

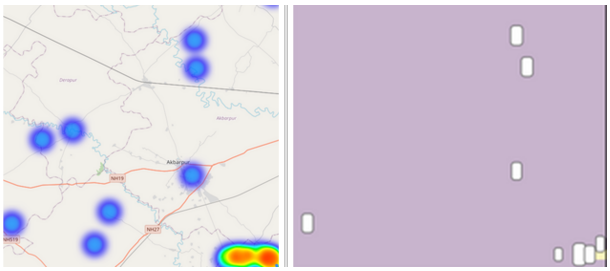


Figure 2: Left: Real World, Right: Simulation. This figure describes how crimes in real world gets mapped to the simulation world.

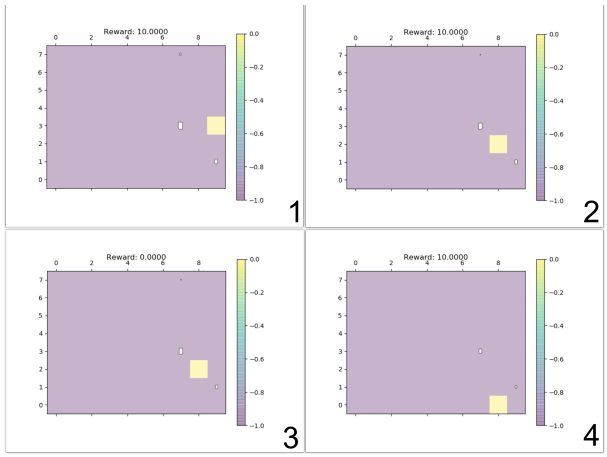


Figure 3: This shows the Setup for all the Experiments. One agent is shown for illustration. Rewards keep decaying with time as shown.

For every experiment we use the same setup. We define a grid-world by taking minimum and maximum latitude and longitude of the district, defining a rectangular grid-world using this and then dividing the entire rectangular grid-world into $100m \times 100m$ grid locations.

As mentioned in the model description, we have three set of rewards. For Instant Reward, we generate the rewards at certain grid location with a probability proportional to the number of calls received from that location in the past. To do this, we take real crime data and map it into the grid-world by assigning the crimes to a grid that is falls into. Instant rewards decay with each step in the simulation. For Proximity Reward, we calculate the Euclidean distance from each of the other vehicles and calculate the reward. For step penalty we add a penalty(α) as required for every step other than the same step. Agents train on the grid-world using multiple training episodes, as is conventional in Q-learning. An episode terminates after a fixed number of steps.

6.1.1 Training helps agents. In our first experiment, we train the model for different number of iterations and then run an on-policy iteration to get the total accumulated rewards in that iteration. We run this experiment for 25 runs for each training epoch length and report the mean accumulated reward in Figure 4.

We see that, as expected, with increasing number of iterations the agents are learning to accumulate higher rewards together. This in turn means that the agents are learning to adapt the behaviour that we intend them to adopt via rewards.

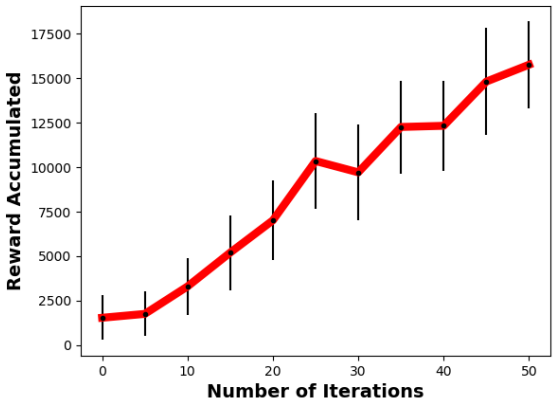


Figure 4: Training time vs reward obtained. The positive trend shows that the training helps the fleet of bots better achieve the designers' objectives. Error bars represent 95% CI measured over 25 experiment runs per x-axis point.

6.1.2 Diminishing returns in reward. In our second experiment, we measure total accumulated rewards with an increasing number of agents in the simulation. We run the experiment in the same setup for every number of agents. We increase the number of agents and observe the total accumulated rewards. We average the rewards over 25 runs per experiment condition.

From Figure 5, we observe that with increasing number of agents, the accumulated reward increases. Importantly, the curve shows diminishing returns, reflecting the real-world expectation that it is counter-productive to place too many patrolling units in any given locality.

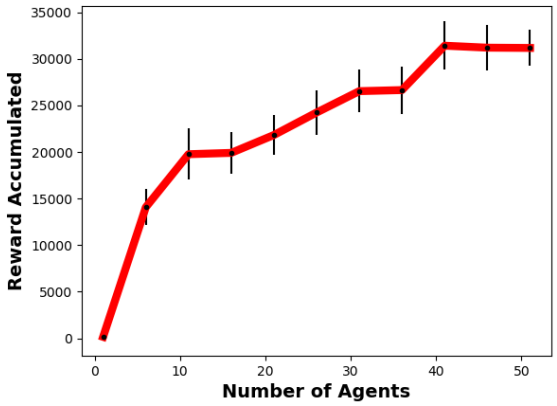


Figure 5: Number of Agents vs Reward Accumulated. The plot shows diminishing returns, suggesting that adding too many patrolling units is unwise. Error bars represent 95% CI measured over 25 experiments runs per x-axis point.

6.1.3 Diminishing returns in response time. In our third experiment, we measure response time to calls with respect to the number of agents in the system. We use distance of the nearest agent, when a call occurs, as a proxy for the response time. For this *in silico* environment, we disregard real-world considerations of road locations, traffic density etc. We observe from Figure 6 that the Response-Time decreases drastically initially, then the rate of decrease reduces slowly and the curve flattens, and looks reasonably exponential. Taken in conjunction with the results reported in Figure 5, these results clearly indicate the possibility of optimizing the number of actively patrolling agents without compromising very much on response time. Thus, these results support the basic premise of our system - that we can cut down on miles driven without affecting response times very much.

6.1.4 The step penalty works. Finally, we measure response time with respect to the distance traversed by all the agents for different values of step penalty. We observe from Figure 7 that when there is no step penalty, the agents tend to move more and it decreases with increasing step penalty. By adding step penalty we can see that, the fleet as a whole travels less. More importantly, a more static fleet actually sees lower response times on average.

This observation occurs because of the specific spatial distribution of call records, likely representative of criminological trends in areas such as the one covered by our partner EMS. Specifically, the district under consideration in our analysis consists of multiple urban clusters separated by physical obstacles (a river, railway lines), and rural clusters dispersed further away. Trying to cover

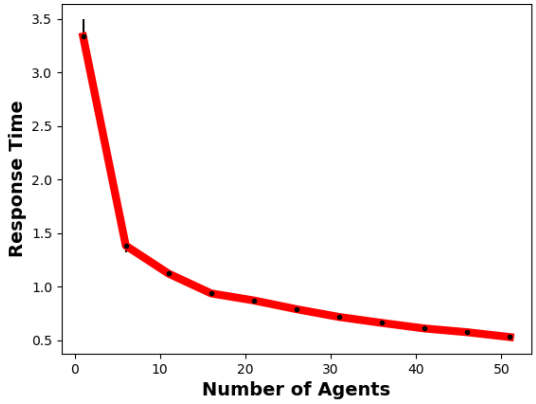


Figure 6: Number of Agents vs Response-Time. The plot shows an exponential decline in response time with the introduction of more agents. Error bars represent 95% CI measured over 25 experiment runs per x-axis point.

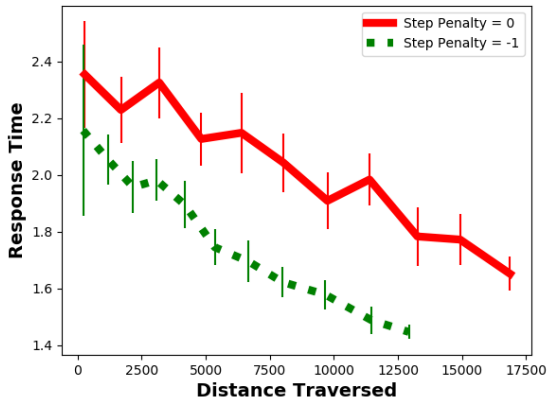


Figure 7: Distance Traversed vs Response-Time. When the step penalty is higher, agents drive around less, and manage to have lower response times overall anyway because the topology of call records in our dataset privileges static beats over dynamic beats. Error bars represent 95% CI measured over 25 experiment runs per x-axis point.

the entire district while patrolling is less efficient than staying close to the urban clusters that report most of the crime.

6.2 Real Environment Results

6.2.1 Qualitative comparison. In this exercise, we compare routes for a single patrol vehicle generated for three separate areas within the district by three separate mechanisms - by a human expert, by a density estimation algorithm, and by our RL-based method. Our observations are visually summarized in Figure 8.

The first column in Figure 8 documents patrol points marked by a human expert. We can see that for the sparsely populated rural area

examined in the top row, the human is actually rather inefficient in placing patrol points - they believe in patrolling along the highway for the most part, while most of the crime is reported from the rural cluster a little removed from the highway. The human's judgments are better in placing patrol points for the dense urban cluster seen in the middle row, and the sub-urban area shown in the bottom row.

The second column shows patrol points generated using DBSCAN, an extremely robust and powerful density estimation algorithm for spatial data [12]. DBSCAN does well on the top and middle rows, but spreads the patrol points too much in the bottom row, expecting the agent to travel a long distance to patrol two points in the extreme north of the area for little profit.

The third column shows patrol points generated using our algorithm. Notice that the algorithm places patrol points really close together for the rural area, widely dispersed for the urban area, and at an intermediate level of density for the sub-urban area. Adaptive variation in plotting routes is one of the key advantages of RL-based approaches over more graph-theoretic approaches, and is evident in our results.

6.2.2 Actualizing a resource-response trade-off. We chose a small area within the district and ran 3 experiments with high, middle and low fuel consumption imperatives for a single vehicle by controlling the step penalty and show the resultant routes. We visualize our observations in Figure 9. The key point to note is that, as fuel consumption becomes increasingly more prioritized, the patrolling route reduces in length, covering lesser area, and prioritizing the more likely locations of call origination.

7 DISCUSSION

We present a detailed implementation of decentralized reinforcement learning for solving an interesting real-world variant of the multi-agent patrol routing problem, with additional first response ability and fuel efficiency requirements. Our system is trained using real emergency call records and vehicle position data, and yields results qualitatively comparable to those generated by a human route planner. This system is currently being field-tested by the police department that instigated this project.

Prior work has already acknowledged the value of reinforcement-learning in solving such problems [5], but uptake has been slow because of inadequate evaluation and absence of connection with real data [24]. Recently however, algorithmic patrolling solutions validated on real data have been reported from Singapore [4] and Israel [23].

For example, the STREETS program designed for traffic police patrolling in the Singapore Central Business District was validated with simulations using traffic volume and traffic violation counts data as scale parameters [4]. Closer in granularity to our own evaluation, the recent system reported by [23] is validated against a record of traffic accidents for Israeli road networks, cross-referenced against additional GIS-based and weather information. [23] use this data to shape the risk calculation component of their patrolling system, and a separate database of Dallas police cars' location in real-time, cross-referenced with Dallas traffic accident data, to shape their system's understanding of how effective the presence of police cars is on preventing car accidents.

The problem we try to solve is distinctly different from the patrolling problem in that patrolling cars are not just meant to deter crime, but also to act as effective first-responders. Additionally, our system tries to solve for fuel efficiency in combination with achieving these objectives. Training the system using real emergency call records data, and comparing predictions of our model with those of human domain experts, we find that our decentralized reinforcement learning works reasonably well. Further operational refinement of the algorithm is naturally possible via improvements in the design of the reward function to include consideration of crime severity and immediacy.

The *sui generis* nature of our practical problem statement prevents a comparison with more sophisticated algorithmic approaches than density-based assignment, which we acknowledge as a limitation of this paper. Since this system is actually meant for deployment, we expect results from an ongoing field test by the police department to yield further evidence of it's value.

ACKNOWLEDGMENTS

We acknowledge sustained and intensive assistance from the police emergency management system we are working with for developing this system.

REFERENCES

[1] Alessandro Almeida, Geber Ramalho, Hugo Santana, Patrícia Tedesco, Talita Menezes, Vincent Corruble, and Yann Chevaleyre. 2004. Recent advances on multi-agent patrolling. In *Brazilian Symposium on Artificial Intelligence*. Springer, 474–483.

[2] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research* 27, 4 (2002), 819–840.

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

[4] Matthew Brown, Sandhya Saisubramanian, Pradeep Varakantham, and Milind Tambe. 2014. Streets: game-theoretic traffic patrolling with exploration and exploitation. (2014).

[5] Huanfa Chen, Tao Cheng, and Sarah Wise. 2017. Developing an online cooperative police patrol routing strategy. *Computers, Environment and Urban Systems* 62 (2017), 19–29.

[6] Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI* 1998, 746-752 (1998), 2.

[7] Robert H Crites and Andrew G Barto. 1998. Elevator group control using multiple reinforcement learning agents. *Machine learning* 33, 2-3 (1998), 235–262.

[8] Reginald Dewil, Pieter Vansteenwegen, Dirk Cattrysse, and Dirk Van Oudheusden. 2015. A minimum cost network flow model for the maximum covering and patrol routing problem. *European Journal of Operational Research* 247, 1 (2015), 27–36.

[9] Maite Dewinter, Christophe Vandeviver, Tom Vander Beken, and Frank Witlox. 2020. Analysing the Police Patrol Routing Problem: A Review. *ISPRS International Journal of Geo-Information* 9, 3 (2020), 157.

[10] John E Eck and Ronald V Clarke. 2019. Situational Crime Prevention: Theory, Practice and Evidence. In *Handbook on Crime and Deviance*. Springer, 355–376.

[11] Yehuda Elmaliach, Noa Agmon, and Gal A Kaminka. 2009. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence* 57, 3-4 (2009), 293–320.

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*, Vol. 96. 226–231.

[13] James Guo Ming Fu and Marcelo H Ang. 2009. Probabilistic ants (pants) in multi-agent patrolling. In *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 1371–1376.

[14] Caroline J Jagtenberg, Sandjai Bhulai, and Robert D van der Mei. 2015. An efficient heuristic for real-time ambulance redeployment. *Operations Research for Health Care* 4 (2015), 27–35.

[15] Christopher S Koper. 1995. Just enough police presence: Reducing crime and disorderly behavior by optimizing patrol time in crime hot spots. *Justice quarterly* 12, 4 (1995), 649–672.

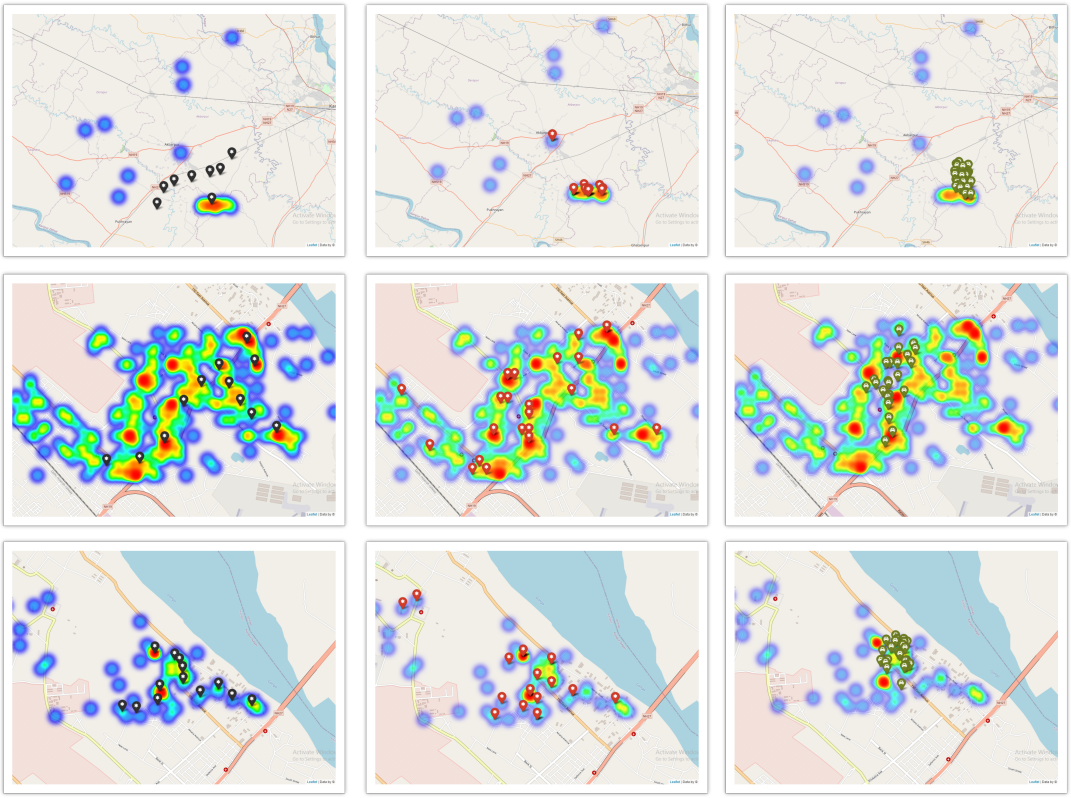
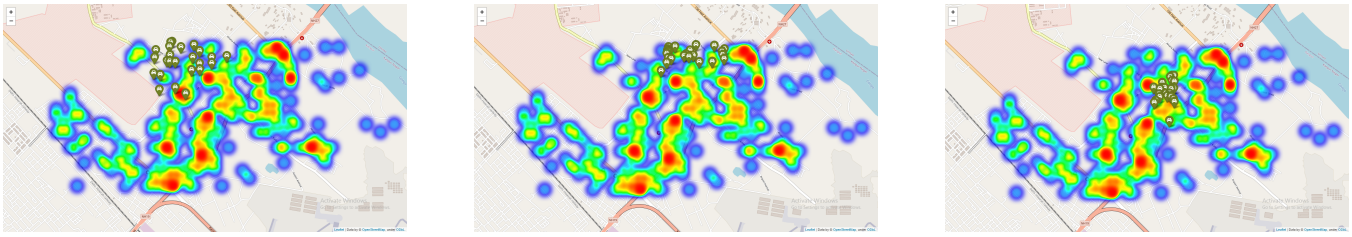


Figure 8: This shows the three Methods of Patrolling point generation. Left: Manual, Middle: DBSCAN, Right: Our Algorithm



(a) Low fuel use restriction. Route length = 9.5kms

(b) Medium fuel use restriction. Route length = 7.8kms

(c) Heavy fuel use restriction. Route length = 4.5kms

Figure 9: Visualizing a resource-response time trade-off in a real-world scenario

[16] Henry CW Lau, George TS Ho, Yi Zhao, and WT Hon. 2010. Optimizing patrol force deployment using a genetic algorithm. *Expert Systems with Applications* 37, 12 (2010), 8148–8154.

[17] Martin Lauer and Martin Riedmiller. 2000. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer.

[18] Martin Lauer and Martin Riedmiller. 2004. Reinforcement learning for stochastic cooperative multi-agent systems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*. Citeseer, 1516–1517.

[19] Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. 2007. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 64–69.

[20] Stephen L Percy. 1980. Response time and citizen evaluation of police. *Journal of Police Science and Administration* 8, 1 (1980), 75–86.

[21] Jerry H Ratcliffe, Travis Taniguchi, Elizabeth R Groff, and Jennifer D Wood. 2011. The Philadelphia foot patrol experiment: A randomized controlled trial of police patrol effectiveness in violent crime hotspots. *Criminology* 49, 3 (2011), 795–831.

[22] Danilo Reis, Adriano Melo, André LV Coelho, and Vasco Furtado. 2006. GAPatrol: An evolutionary multiagent approach for the automatic definition of hotspots and patrol routes. In *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*. Springer, 118–127.

[23] Ariel Rosenfeld, Oleg Maksimov, and Sarit Kraus. 2020. When security games hit traffic: A deployed optimal traffic enforcement system. *Artificial Intelligence* 289 (2020), 103381.

[24] Hugo Santana, Geber Ramalho, Vincent Corruble, and Bohdana Ratitch. 2004. Multi-agent patrolling with reinforcement learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*. 1122–1129.

[25] Lawrence W Sherman and David Weisburd. 1995. General deterrent effects of police patrol in crime "hot spots": A randomized, controlled trial. *Justice quarterly*

12, 4 (1995), 625–648.

[26] Ruben Stranders, E Munoz De Cote, Alex Rogers, and Nicholas R Jennings. 2013. Near-optimal continuous patrolling with teams of mobile information gathering agents. *Artificial intelligence* 195 (2013), 63–105.

[27] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[28] Jason Tsai, Zhengyu Yin, Jun-young Kwak, David Kempe, Christopher Kiekintveld, and Milind Tambe. 2010. Urban security: Game-theoretic resource allocation in networked physical domains. In *National Conference on Artificial Intelligence (AAAI)*. Citeseer, 881–886.

[29] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[30] Chongjie Zhang, Victor R Lesser, and Sherief Abdallah. 2010. Self-organization for coordinating decentralized reinforcement learning.. In *AAMAS*, Vol. 10. 739–746.

[31] Shanjiang Zhu, Woon Kim, Gang-Len Chang, and Steve Rochon. 2014. Design and evaluation of operational strategies for deploying emergency response teams: Dispatching or patrolling. *Journal of Transportation Engineering* 140, 6 (2014), 04014021.