Hierarchical Planning for Resource Allocation in Emergency Response Systems

Geoffrey Pettet
Vanderbilt University
Nashville, TN
geoffrey.a.pettet@vanderbilt.edu

Ayan Mukhopadhyay
Vanderbilt University
Nashville, TN
ayan.mukhopadhyay@vanderbilt.edu

Mykel Kochenderfer
Stanford University
Stanford, CA
mykel@stanford.edu

Abhishek Dubey
Vanderbilt University
Nashville, TN
abhishek.dubey@vanderbilt.edu

ABSTRACT

A classical problem in city-scale cyber-physical systems (CPS) is resource allocation under uncertainty. Spatial-temporal allocation of resources is optimized to allocate electric scooters across urban areas, place charging stations for vehicles, and design efficient on-demand transit. Typically, such problems are modeled as Markov (or semi-Markov) decision processes. While online, offline, and decentralized methodologies have been used to tackle such problems, none of the approaches scale well for large-scale decision problems. We create a general approach to hierarchical planning that leverages structure in city-level CPS problems to tackle resource allocation under uncertainty. We use emergency response as a case study and show how a large resource allocation problem can be split into smaller problems. We then create a principled framework for solving the smaller problems and tackling the interaction between them. Finally, we use real-world data from a major metropolitan area in the United States to validate our approach. Our experiments show that the proposed approach outperforms state-of-the-art approaches used in the field of emergency response.

KEYWORDS

dynamic resource allocation, large-scale CPS, planning under uncertainty, hierarchical planning, semi-Markov decision process

Accepted for publication in the proceedings of the 12th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS-2021).

1 INTRODUCTION

Dynamic resource allocation (DRA) in anticipation of uncertain demand is a canonical problem in city-scale cyber-physical systems (CPS). In such a scenario, the decision-maker optimizes the spatial location of resources (typically called *agents*) to maximize utility over time while satisfying constraints specific to the domain of the CPS. Consider the problem of emergency response. Governments and private agencies must optimize the location of ambulances to minimize response times to emergency calls. The decision-maker has constraints on the number of available ambulances and locations where they can be stationed. An associated problem is resource dispatch, which manifests itself in situations where agents need to spatially move from their location to the location of the demand to address the task at hand. Ambulances must move to the scene of the incidents to provide assistance to patients. In this paper, we address the problem of dynamical resource allocation and dispatch in large-scale systems.

Our approach can model many different resource allocation problems at the intersection of urban management, CPS, and multi-agent systems. The problem of resource allocation under uncertainty manifests in allocating electric scooters [8], optimizing locations of charging stations for vehicles [1], designing on-demand transit [2], and emergency response management (ERM) [18]. We focus on ERM in this paper for several reasons. First, it is a critical problem faced by communities across the globe. Responders must attend to many critical incidents dispersed across space and time using limited resources. Mukhopadhyay et al. [18] describe the critical nature of emergency response and the intricate pipeline that first-responders follow to ensure timely and effective service. Second, ERM pipelines are a classic example of human-in-the-loop CPS (H-CPS), which introduces structure and constraints that let us take into account important real-world considerations. Finally, emergency response presents us the scope to evaluate multi-agent resource allocation and dispatch problems with high-quality real-world data.

Dynamic resource allocation and dispatch problems are typically modeled as Markov decision processes (MDP). The goal of the decision-maker is to find an optimal *policy*, which is a mapping between states of the system and actions that need to be taken. Problems pertaining to CPS in urban areas evolve in continuous time. The integration of dispatch into the problem makes state-transitions non-memoryless, so the dynamics of the underlying continuous-time stochastic process are actually semi-Markovian [20]. We show a broad spectrum of approaches to address resource allocation under uncertainty in figure 1. The most direct approach is to represent the problem as a single MDP, shown on the left in figure 1. However, when real-world problems are modeled as MDPs, the state transitions are difficult to estimate in closed-form. The standard approach to address this issue is to use a black-box simulator which is relatively simple to construct [20]. Then, the simulator can be used to estimate an empirical distribution over the state transitions [20].

This work is sponsored by The National Science Foundation under award numbers CNS1640624 and IIS1814958 and a grant from Tennessee Department of Transportation. We also acknowledge support from Google through the Cloud Research Credits.

Given the transition distribution, an optimal policy can be learned using the well-known policy iteration algorithm [13]. Another approach is to use an online solution, where given a specific state, the simulator aids a heuristic search algorithm like Monte-Carlo tree search (MCTS) [17].

A different approach is to use a completely decentralized methodology, as shown in the extreme right in figure 1. In such an approach, each agent determines its own course of action. As the agents cooperate to achieve a single goal, they must estimate what other agents will do in the future as they optimize their own actions. For example, Claes et al. [4] show how each agent can explore the efficacy of its actions locally by using MCTS. While such approaches are significantly more scalable than their centralized counterparts, they are sub-optimal as agents' estimates of other agents' actions can be highly inaccurate. Note that high-fidelity models for estimating agents' actions limits scalability and therefore decentralized approaches rely on computationally cheap heuristics. Decentralized approaches are useful in disaster scenarios where communication networks can break down, but agents in urban areas (ambulances) typically have access to reliable networks and communication is not a constraint. Therefore, approaches that ensure scalability but do not fully use all available information during planning are not suitable for emergency response in urban areas, especially when fast and effective response is critical.

An orthogonal approach to solve large-scale MDPs is to use hierarchical planning [9], which focuses on learning local policies, known as *macros*, over subsets of the state space. We use hierarchical planning to address resource allocation for emergency response by leveraging structure in the problem. Our idea is also motivated by the concept of *jurisdictions* or *action-areas* used in public policy, which create different zones (typically spatial) to segregate and better manage infrastructure.

We design a principled algorithmic approach that segregates the spatial area under consideration. We then treat resource allocation in each resulting sub-area (called regions) as individual planning problems which are smaller than the original problem by construction. While this ensures scalability, it naturally results in performance loss, as agents constrained in one region might be needed in the other region. To tackle this, we show how hierarchical planning can be used to facilitate transfer of agents across regions. Specifically, the top-level planner, called the inter-region planner, identifies and detects states where "interaction" between regions is necessary and finds actions such that the overall utility of the system can be maximized. The low-level planner, called the intra-region planner, works to tackle the problem of allocation and dispatch within a region.

Contributions: **1)** We leverage structure in resource allocation problems to design a hierarchical planning approach that scales significantly better than prior approaches. The key idea in our approach comes from the concept of *macros* in decision-theoretic systems [9], which focus on finding policies for subsets of the state-space, thereby ensuring scalability. **2)** We show how exogenous constraints in real-world resource allocation problems can be used to naturally segregate the overall decision-problem into sub-problems. We create a low-level planner which focuses on finding optimal policies for the sub-problems. **3)** We show how a high-level planner can facilitate exchange of resources between the sub-problems (spatial areas in our case). **4)** We use real-world emergency response data from a major metropolitan area in USA to evaluate our approach, and show that it performs better than state-of-the-art approaches both in terms of efficiency and scalability.

The paper is organized as follows. We describe ERM and a mathematical formulation of our problem in section 2. Section 3 describes the overall approach, the high-level, and the low-level planner. We present experimental results in section 4 and summarize the paper in section 7.

2 PROBLEM FORMULATION

Emergency response management (ERM) deals with responding to spatial-temporal calls for service in a specified spatial area. The agents, ambulances in this case, respond to calls for medical aid. Once an incident is reported, responders are dispatched by a human agent to the scene of the incident (guided by some algorithmic approach). If no free responder is available, the incident typically enters a waiting queue, and is responded to when an agent becomes free (we use "agent" and "responder" inter-changeably throughout the paper). Each responder is typically housed at specific locations called depots, which are distributed in the spatial area under consideration (these could be fire-stations or rented parking spots, for example). Once a responder finishes servicing an incident, it is directed back to a depot and becomes available for dispatch. Therefore, there are two broad actions that the decision maker can optimize: (1) which responder to dispatch once an incident occurs (dispatching action) and (2) which depots to send the responders to in anticipation of future incidents (allocation action). A key aspect of emergency response is that if any free responders are available when an incident is reported, then one must be dispatched to attend to the incident. This constraint is a direct consequence of the bounds within which emergency responders operate, as well as the critical nature of the incidents [18].

To model the problem of emergency response management, we begin with several assumptions on the problem structure and information provided *a priori*. First, we assume that we are given a spatial map broken up into a finite collection of equally sized cells G, and a set of agents Λ that need to be allocated across these cells and dispatched to demand points. We also assume that we have access to a spatial-temporal model of demand over G, and that within each cell the temporal demand distribution is homogeneous. Our third assumption is that agent allocation is restricted to *depots* D, that are located in a fixed subset of cells. Each depot $d \in D$ has a fixed capacity $C(d)$ number of agents it can accommodate.

While the state space in this resource allocation problem evolves in continuous-time, it is convenient to view the dynamics as a set of specific decision-making states. As an example, an ambulance moving through an area continuously changes the state of the *world*, but presents no scope for decision-making unless an event occurs that needs response or the planner redistributes agents. As a result, the decision-maker only needs to find optimal actions for a subset of the state space that provides the opportunity to take actions.

A key component of response problems is that agents physically move to the site of the request. This property makes temporal transitions between decision-making states non-memoryless, and is a crucial consideration in designing decision-theoretic models

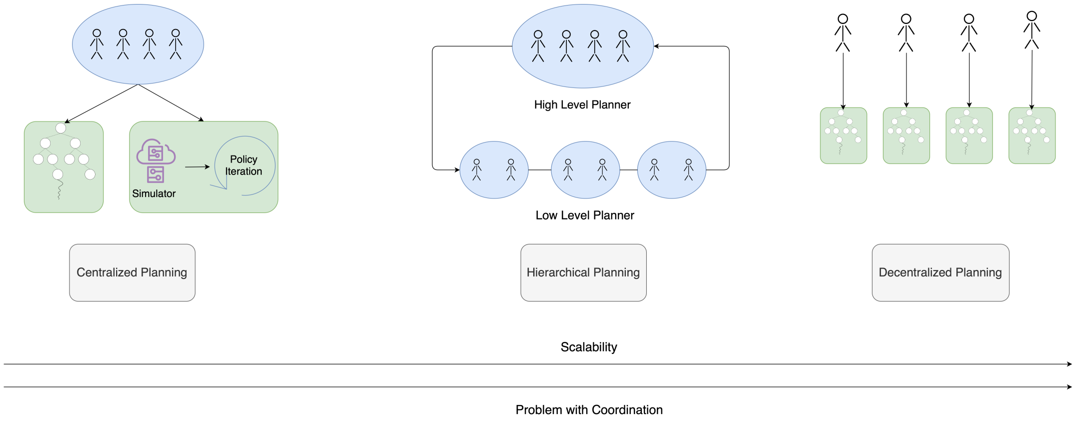


Figure 1: Different approaches to solve a dynamic resource allocation problem under uncertainty. A completely centralized approach (leftmost) deals with a monolithic state representation. In a completely decentralized approach each agent simulates what other agents do and performs its own action estimation (rightmost). Our approach (middle) segments the original planning problem into multiple sub-problems to improve scalability without agents estimating other agents' actions.

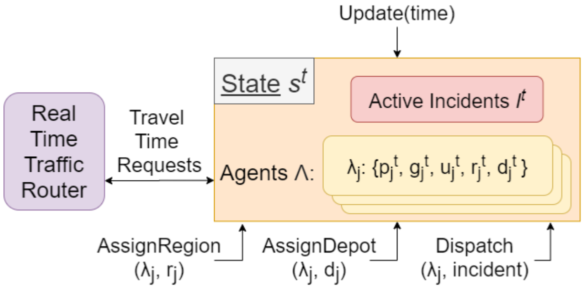


Figure 2: System State and the actions.

since it causes the underlying stochastic process that governs the evolution of the system to be semi-Markovian. The dynamics of a set of agents working to achieve a common goal can be modeled as a Multi-Agent Semi-Markov Decision Process (MSMDP) [24], which can be represented as the tuple $(S, \Lambda, \mathcal{A}, P, T, \rho(s,a), \alpha, \mathcal{T})$, where S is a finite state space, $\rho(s,a)$ represents the instantaneous reward for taking action a in state s, P is a state transition function, T is the temporal distribution over transitions between states, α is a discount factor, and Λ is a finite collection of agents where $\lambda_j \in \Lambda$ denotes the jth agent. The action space of the jth agent is represented by A_j, and $\mathcal{A} = \prod_{i=1}^{m} A_j$ represents the joint action space of all agents. We assume that the agents are cooperative and work to maximize the overall utility of the system. $\mathcal{T}$ represents a termination scheme; note that since agents each take different actions that could take different times to complete, they may not all terminate at the same time [24]. We focus on asynchronous termination, where actions for a particular agent are chosen as and when the agent completes its last assigned action.[1]

States: A state at time t is represented by s^t and consists of a tuple $(I^t, Q(s^t))$, where I^t is a collection of cell indices that are waiting to be serviced, ordered according to the relative times of incident occurrence. $Q(s^t)$ corresponds to information about the set of agents at time t with $|Q(s^t)| = |\Lambda_r|$. Each entry $q_j^t \in Q(s^t)$ is a set $\{p_j^t, g_j^t, u_j^t, r_j^t, d_j^t\}$, where p_j^t is the position of agent λ_j, g_j^t is the destination cell that it is traveling to (which can be its current position), u_j^t is used to encode its current status (busy or available), r_j^t is the agent's assigned region, and d_j^t is it's assigned depot, all observed at the state of our world at time t. A diagram of the state and how it can be interacted with is shown in figure 2 and discussed in detail in appendix A.

We assume that no two events occur simultaneously in our system model. In such a case, since the system model evolves in continuous time, we can add an arbitrarily small time interval to segregate the two events and create two separate states.

Actions: Actions correspond to directing agents to valid cells to either respond to demand or wait at a depot. For a specific agent $\lambda_i \in \Lambda_r$, valid actions for a specific state s_i are denoted by $A^i(s_i)$ (some actions are naturally invalid, for example, if an agent is at cell k in the current state, any action not originating from cell k is unavailable to the agent). Actions can be broadly divided into two categories: *dispatching* actions which direct agents to service an active demand point and *allocation* actions which assign agents to wait in particular depots in anticipation of future demand.

Here it is important to note that in the ERM case study, dispatch actions are greedily prescribed [18, 22]. In practice, the severity of incidents can not be gauged from calls for service and therefore, the closest available responder must be dispatched to the incidents. We consider that dispatch actions are greedy by default, and therefore do not optimize over such actions. Therefore, the problem we consider focuses on proactively redistributing agents across a spatial area under future demand uncertainty. Nonetheless, dispatch actions are still necessary to model since they are the foundation of our reward function.

Transitions: The system model of ERM evolves through several stochastic processes. Incidents occur at different points in time and space governed by some arrival distribution. We assume that

[1] Different termination schemes are discussed in the theoretical analysis by Rohanimanesh and Mahadevan [24].

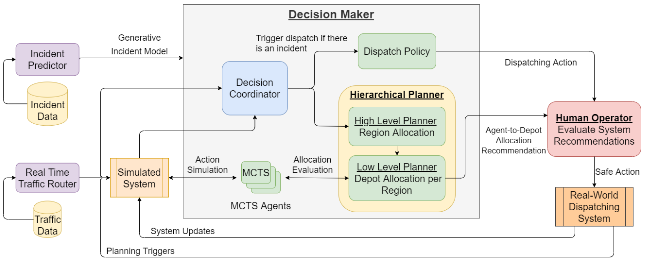


Figure 3: Solution Approach

the number of incidents in a cell $r_j \in R$ per unit time can be approximated by a Poisson distribution with mean rate γ_j (per unit time), a commonly used model for spatial-temporal incident occurrence [18]. Agents travel from their locations to the scene of incidents governed by a model of travel times. We assume that agents then take time to service the incident at some exogenously specified velocity. The system itself takes time to plan and implement allocation of responders. We refrain from discussing the mathematical model and expressions for the temporal transitions and the state transition probabilities, as our algorithmic framework only needs a generative model of the world (in the form of a black-box simulator) and not explicit estimates of transitions themselves.

Rewards: Rewards in SMDP usually have two components: a lump sum instantaneous reward for taking actions, and a continuous time reward as the process evolves. Our system only involves the former, which we denote by $\rho(s, a)$, for taking action a in state s. Rewards are highly domain dependent; the metric we are concerned with in our ERM case study is *incident response time* t_r, which is the time between the system becoming aware of an incident and when the first agent arrives on scene. Therefore, our reward function is

$$\rho(s,a) = \begin{cases} \alpha^{t_h}(t_r(s,a)) & \text{if dispatching} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where α is the discount factor for future rewards, t_h the time since the beginning of the planning horizon t_0, and $t_r(s, a)$ is the response time to the incident due to a dispatch action. There is no immediate reward for allocation actions; their benefits are inferred from improved future dispatching.

Problem Definition Given state s and a set of agents Λ, our goal is to find an action recommendation set $\sigma = \{a_1, ..., a_m\}$ with $a_i \in A^i(s)$ that maximizes the expected reward. The ith entry in σ contains a *valid* action for the ith agent. In our ERM case study, this corresponds to finding an allocation of agents to depots that minimizes the expected response times to incidents.

3 APPROACH

We show a schematic representation of our decision support system in Fig. 3. We assume the availability of historical incident data and a probabilistic generative model of incident occurrence learned from the data. We also assume access to a traffic router that can simulate traffic in the concerned urban area (for example, see prior work done by Mukhopadhyay et al. [17]). We divide tasks pertaining to planning in a hierarchical manner. We refer to the two stages in our hierarchy as the "high-level" and the "low-level". The high-level planner divides the overall decision-theoretic problem into smaller sub-problems by creating meaningful spatial clusters, which we call regions. It also optimizes the distribution of agents among the regions. Then, an instance of the low-level planner is instantiated for each of the regions. The low-level planner for a specific region optimizes the spatial locations of agents within that region. Finally, a human operator accesses the planning mechanism to act as an interface with a real-world computer-aided dispatch system.

An important consideration in designing approaches to resource allocation under uncertainty in city-level CPS problems is to adapt to the dynamic environment in which such systems evolve. In our decision support system, a decision coordinator (an automated module) invokes the high-level planner at all states that allow the scope for making decisions. For example, consider that a responder is unavailable due to maintenance. The coordinator triggers the high-level planner and notifies it of the change. The high-level planner then checks if the spatial distribution of the responders can be optimized to best respond to the situation at hand. We describe the exact optimization functions, metrics, and approaches that we use to design the planners below.

3.1 High-Level Planner

Through the high-level planner, we seek to decompose the overall MSMDP into a set of smaller problems that can be solved tractably and distribute agents among the regions. Recall that the overall goal of the parent MSMDP is to reduce expected response times. Response times to emergency incidents consist of two parts: a) the time taken by an agent to travel to the scene of the incident, and b) the time taken to service the incident. We assume that incidents are homogeneous, meaning that the time taken by responders to service incidents follows the same distribution. This assumption means that the sole criterion that a planner needs to optimize is overall travel time of the agents to incidents (achieving zero waiting times is clearly infeasible in practice, so we seek to minimize waiting times). Therefore, the high-level planner seeks to distribute responders to different regions such that incident waiting time is minimized. There are other factors that contribute to waiting times (traffic, for example). However, such factors are not affected by the distribution of the agents themselves.

Consider that the high level planner seeks to divide the overall problem in to k regions, denoted by the set $R = \{r_1, r_2, \ldots, r_k\}$, where $r_i \in R$ denotes the ith region. To achieve this, we use historical data of incident occurrence to partition the set G into clusters using a standard clustering algorithm (we use the k-means algorithm in our experiments). We denote the average waiting time for incidents in region $r_j \in R$ by $w_j(x_j)$, where x_j is the number of responders assigned to the region r_j. We model waiting times in a region by a multi-server queue model. Recall that incident arrivals are distributed according to a Poisson distribution, thereby making inter-incident times exponentially distributed. We make the standard assumption that service times are exponentially distributed [19]. One potential issue with using well-known queuing models to estimate waiting times in emergency response is that travel times are not memoryless. We use an approximation from prior work to tackle this problem [19]. Specifically, travel times to emergency incidents are typically much smaller than service times. Thus, the sum of travel times and service times can be considered to be approximated by a memoryless distribution (provided that the service time itself can be modeled by a memoryless distribution). The average waiting-time for a region $r_j \in R$ can then be estimated by considering a $m/m/c$ queuing model (using Kendall's notation [10]), where $c = x_j$.

Given a method for estimating the waiting times for incidents in each region, the high-level planner seeks to minimize the cumulative response times across all regions. The optimization problem that the high-level planner seeks to solve can be represented as

$$\min_x \sum_{j=1}^{k} w_j(x_j) \tag{2a}$$

$$\text{s.t.} \quad \sum_{i=1}^{k} x_i = |\Lambda| \tag{2b}$$

$$x_i \in \mathbb{Z}^{0+} \; \forall i \in \{1, \ldots, k\} \tag{2c}$$

Let the average rate of incident occurrence be λ_j at region $r_j \in R$. Let the average service time be T_s and let $\mu = 1/T_s$ denote the mean service rate. Then, the mean waiting time $w_j(x_j)$ can be represented as [25]:

$$w_j(x_j) = \frac{P_0(\frac{\lambda}{\mu})^{x_j}\lambda}{c!(1-\rho)^2 c}$$

where P_0 denotes the probability that there are 0 incidents waiting for service and can be represented as

$$P_0 = 1/\Big[\sum_{m=0}^{x_j-1} \frac{(x_j\rho)^m}{m!} + \frac{(c\rho)^c}{c!(1-\rho)}\Big]$$

A challenge in solving mathematical program 2 is that the objective function is non-linear and non-convex. We tackle this problem by using an iterative greedy approach shown in algorithm 1. We begin by sorting regions according to total arrival rates. Since the arrival process is assumed to be Poisson distributed, the overall rate for region $r_j \in R$, denoted by γ_j can be calculated as $\sum_{g_i \in G} \mathbb{1}(g_i \in r_j)\gamma_i$, where $\mathbb{1}(g_i \in r_j)$ denotes an indicator function which checks if cell g_i belongs to region $r_j \in R$. Let this sorted list be R_s. Then, we assign responders iteratively to regions in order of decreasing arrival rates (step 3). After assigning each responder to a region $r_j \in R$, we compare the overall service rate (x_j times the mean service rate by one responder) and the incident arrival rate for the region (step 5). Essentially, we try to ensure that given a pre-specified service rate, the expected length of the queue is not arbitrarily large. Once a region is assigned enough responders to sustain the arrival of incidents, we move on the next region in the sorted list R_s (step 6). Once all regions are assigned responders in this manner, we check if there are surplus responders (step 9). The surplus responders are assigned iteratively according to the incremental benefit of each assignment. Specifically, for each region, we calculate the marginal benefit J of adding one agent to the existing allocation (step 11). Then, we assign an agent to the region that gains the most (in terms of reduction in waiting times) by the assignment.

Algorithm 1: High-Level Planner

```
   input  : Sorted Regions $R_s$, Arrival Rates $\{\gamma_1, \gamma_2, \ldots, \gamma_k\}$,
            Service Rate $\eta$
   output : Responder Allocation $X = \{x_1, x_2, \ldots, x_k\}$
 1 assigned := 0, $i$ := 0, $j$ := 0;
 2 while assigned ≤ | $\Lambda$ |  and $i \leq k$ do
 3     $x_i := x_i + 1$;
 4     assigned := assigned + 1;
 5     if $\eta \times (x_i) \geq \sum_{g_i \in G} \mathbb{1}(g_i \in r)\gamma_i$ then
 6         $i := i + 1$;
 7     end
 8 end
 9 while assigned ≤ | $\Lambda$ | do
10     Calculate $J$ where $j_i = w_i(x_i) - w_i(x_i + 1)$;
11     $r^* = \arg\max_{r_i \in R} J$;
12     $x_{r^*} := x_{r^*} + 1$;
13     assigned := assigned + 1;
14 end
```

3.2 Low-Level Planner

The fine-grained allocation of agents to depots within each region is managed by the low level planner, which induces a decision process for each region that is smaller than the original MSMDP problem described in section 2 by design. The MSMDP induced by each region $r_j \in R$ contains only state information and actions that are relevant to r_j, i.e. the depots within r_j, the agent's assigned to r_j by the inter-region planner, and incident demand generated within r_j.

Decomposing the overall problem makes each region's MSMDP tractable using many approaches, such as dynamic programming, reinforcement learning (RL), and Monte Carlo Tree Search (MCTS). Each approach has advantages and tradeoffs which must be examined to determine which is best suited with respect to the specific problem domain that is being addressed.

Spatial-temporal resource allocation has a key property that informs the solution method choice — a highly dynamic environment that is difficult to model in closed-form. To illustrate, consider an agent travel model. While there are certainly long term trends for travel times, precise predictions are difficult due to complex interactions between features such as traffic, weather, and events occurring in the city. A city's traffic distribution also changes over time as the road network and population shifts, so it needs to be updated periodically with new data. This dynamism is true for many pieces of the domain's environment, including the demand distribution of incidents. Importantly, it is also true of the system itself: agents can enter and leave the system due to mechanical issues or purchasing decisions, and depots can be closed or opened.

Whenever underlying environmental models change, the solution approach must take the updates into account to make correct recommendations. Approaches that require long training periods such as reinforcement learning and value iteration are difficult to apply since they must be re-trained each time the environment changes. This motivates using Monte Carlo Tree Search (MCTS), a probabilistic search algorithm, as our solution approach. Being an anytime algorithm, MCTS can immediately incorporate any changes in the underlying generative environmental models when making decisions.

MCTS represents the planning problem as a "game tree", where states are represented by nodes in the tree. The decision-maker is given a state of the world, and is tasked with finding a promising action for the state. The current state is treated as the root node, and actions that take you from one state to another are represented as edges between corresponding nodes. The core idea behind MCTS is that this tree can be explored asymmetrically, with the search being biased toward actions that appear promising. To estimate the value of an action at a state node, MCTS simulates a "playout" from that node to the end of the planning horizon using a computationally cheap *default policy* (our simulated system model is shown in figure 2 and described in detail in appendix A). This policy is generally not very accurate (a common method is random action selection), but as the tree is explored and nodes are revisited, the estimates are re-evaluated and will converge toward the true value of the node. This asymmetric tree exploration allows MCTS to search very large action spaces quickly.

When implementing MCTS, there are a few domain specific decisions to make — the *tree policy* used to navigate the search tree and find promising nodes to expand, and the *default policy* used to quickly simulate playouts and estimate the value of a node.

Tree Policy: When navigating the search tree to determine which nodes to expand, we use the standard Upper Confidence bound for Trees (UCT) algorithm [14], which defines the score of a node n as

$$\text{UCB(n)} = \overline{u(n)} + c\sqrt{\frac{\log(\text{visits}(n))}{\text{visits}(n')}} \tag{3}$$

where $\overline{u(n)}$ is the estimated utility of state at node n, visits(n) is the number of times n has been visited, and n' is n's parent node. When deciding which node to explore in the tree, the child node with the maximum UCB score is chosen. The left term $\overline{u(n)}$ is the exploitation term, and favors nodes that have been shown to be promising. The right term is the exploration term, and benefits nodes with low visit counts, which encourages the exploration of under-represented actions in the hope of finding an overlooked high value path. The constant c controls the tradeoff between these two opposing objectives, and is domain dependent.

Default Policy: When working outside the MCTS tree to estimate the value of an action, i.e. rolling out a state, a fast heuristic *default policy* is used to estimate the score of a given action. Rather than using a random action selection policy, we exploit our prior knowledge that agents generally stay at their current depot unless large shifts in incident distributions occur. Therefore, we use greedy dispatch without any redistribution of responders as our heuristic default policy.

It is important to note that performing MCTS on one sampled chain of events is not enough, as traffic incidents are inherently sparse. Any particular sample will be too noisy to make robust claims regarding the value of an action. To handle this uncertainty, we use *root parallelization.* We sample many incident chains from the prediction model, and instantiate separate MCTS trees to process each. We then average the scores across trees for each potential allocation action to determine the optimal action.

Algorithm 2: Low-Level Planner

```
   input  : Regions $R$, State $s$, Generative Demand Model $E$,
            Number of Samples $n$
   output : Recommended Allocation Actions $\sigma_r \ \forall r \in R$
 1 for region $r_j \in R$ do
 2     Decompose $s$ into region specific state $s_j$;
 3     Action Score Map $\widetilde{\mathcal{A}} := \emptyset, i := 0$;
 4     eventChains := $E$.sample($s_j, n$);
 5     action scores $A$ := MCTS($s_j$, eventChains);
 6     for action $a \in A$ do
 7         $\widetilde{\mathcal{A}}[a]$.append(score($a$));
 8     end
 9     $\overline{\mathcal{A}} := \emptyset$;
10     for potential action $a \in \widetilde{\mathcal{A}}$ do
11         $\overline{\mathcal{A}}[a]$ = mean($\widetilde{\mathcal{A}}[a]$);
12     end
13     Recommended action $\sigma_r$ := $\text{argmax}_a \ \overline{\mathcal{A}}[a]$
14 end
```

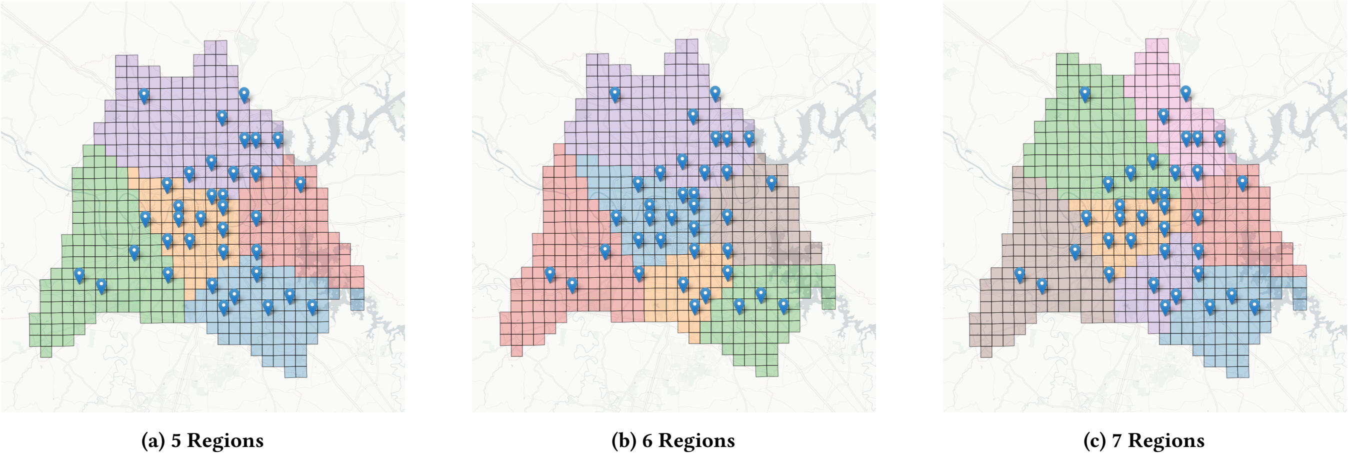
Our low-level planning approach is shown in algorithm 2. The inputs for low level planning are the regions R, the current overall system state s (which includes each agent's region assignment), a generative demand model E, and the number of chains to sample and average over for each region n. For each region $r_j \in R$, we first extract the state s_j in the region's MSMDP from the current overall system state s (step 2). Then we perform root parallelization by sampling n incident chains from the demand model E and performing MCTS on each to score each potential allocation action (step 4). It is important to note that the sampled incident chains are specific to the region under investigation, and demand is only generated from the cells that are in that region. We then average the scores across samples for each action, and choose the allocation action with the maximum average score (step 13).

4 EXPERIMENTS

We evaluate the proposed hierarchical framework's effectiveness on emergency response data obtained from a major metropolitan area in the USA, with a population of approximately 700,000 (we suppress the exact name of the area for blind review). We use actual historical incident data, depot locations, and operational data. We construct a grid representation of the city using 1x1 mile square cells. This choice was a consequence of the fact that a similar granularity of discretization is followed by local authorities. These cells, as well as the city's 35 depot locations, can be seen in figure 4.

We make a few important assumptions when configuring our experiments. First, we limit the capacity of each depot $C(d)$ to 1. This encourages responders to be geographically spread out to respond quickly to incidents occurring in any region of the city, and it models the usage of ad-hoc stations by responders, which are often temporary parking spots.[2] We assume there are 26 available responders to allocate, which is the actual number of responders in the urban area under consideration.

[2] In theory, we could always add dummy depots at the same location to extend our approach to a situation where more than one responder per depot is needed.



(a) 5 Regions

(b) 6 Regions

(c) 7 Regions

Figure 4: Maps presenting the various spatial regions under consideration. Pins on the map represent the locations of depots, and different colors represent different spatial regions. Each map shows a different number of regions.

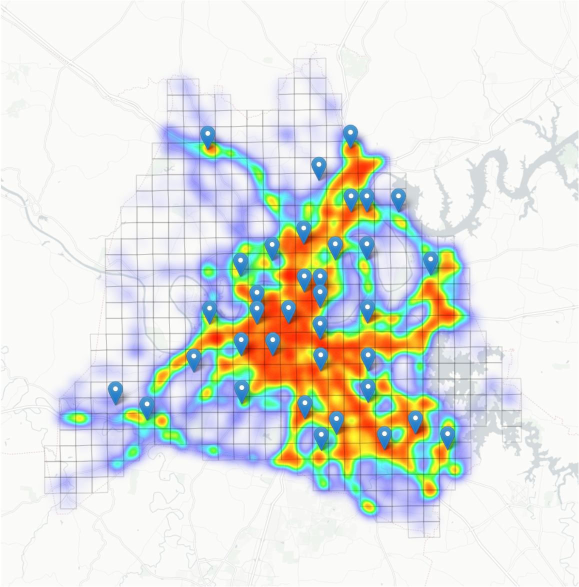


Figure 5: Historic incident density overlaid on the spatial grid environment. Pins represent depots locations.

Our experimental hyper-parameter choices are shown in table 1. In our experiments, we vary the number of spatial regions to examine how their size and distribution effects performance of the hierarchical planner; the resulting region configurations can be seen in figure 4. We use the k-means algorithm [15] implemented in scikit-learn [21] on historical incident data, which consists of 47862 incidents that occurred from January 2018 to May 2019 in our city of interest obtained from the local government.

We assume that the mean rate to service an incident is 20 minutes based on actual service times in practice in the area of consideration (we hold this constant in our experiments to directly compare the planning approaches). As mentioned in section 3, we assume that incidents are homogeneous. The number of MCTS iterations performed when evaluating potential actions on a sampled incident chain is set to 1000 and the number of samples from the incident model that are averaged together using root parallelization during each decision step is set to 50. We run the hierarchical planner after each test incident to re-allocate responders. Further, if the planner is not called after a pre-configured time interval, we call it to ensure that the allocations are not stale. In our experiments this maximum time between allocations is set to 60 minutes. We ran experiments on an intel i9-9980XE based system, which has 38 logical processors running at a base clock of 3.00 GHz, and 64 GB RAM.

Table 1: Experimental hyper-parameter choices

Parameter	Value(s)
Number of Regions	{5, 6, 7}
Maximum Time Between Re-Allocations	60 Minutes
Incident Service Time	20 Minutes
Responder Speed	30 Mph
MCTS Iteration Limit	1000
Discount Factor	0.99995
UCT Tradeoff Parameter c	1.44
Number of Generated Incident Samples	50

Our incident model is learned from the 47862 real incidents discussed eariler. For each cell, we learn a Poisson distribution over incident arrival based on historical data. The maximum likelihood estimate (MLE) of the rate of the Poisson distribution is simply the empirical mean of the number of incidents in each unit of time. To simulate our system model, we access the Poisson distribution of each cell and sample incidents from it. In reality, emergency incidents might not be independently and identically distributed; however, the incident arrival model (and the blackbox simulator of the system in general) is completely exogenous to our model and does not affect the functioning of our approach. To validate the robustness of our approach, we create three separate test beds based on domain knowledge and preliminary data analysis of historical incident data.

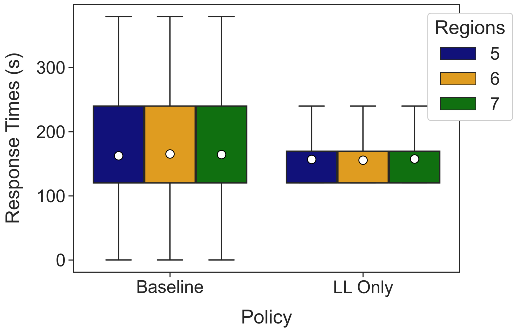


Figure 6: The response time distributions for the baseline and low level planners (LL Only) when applied to incidents sampled from a stationary rate distribution. The boxplot represents the data's Inter-Quartile Range (IQR = $Q_3 - Q_1$), and the whiskers extend to 1.5IQR

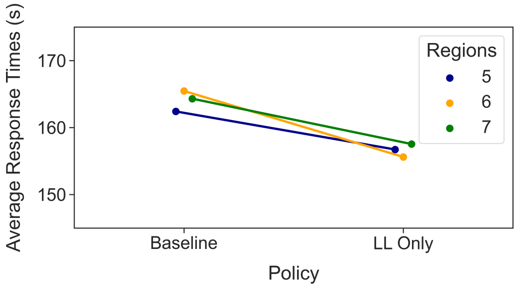


Figure 7: A zoomed in view of the average response times for the baseline and low level planners (LL Only) when applied to incidents sampled from a stationary rate distribution.

Stationary incident rates: We start with a scenario where our forecasting model samples incidents from a Poisson distribution that is stationary (for each cell), meaning that the rate of incident occurrence for each cell is the empirical mean of historical incident occurrence per unit time in the cell. This means that the only utility of the high-level planner in such a case is to divide the overall spatial area into regions and optimize the initial distribution of responders among them. Since the rates are stationary, the initial allocation is maintained throughout the test period under consideration. This scenario lets us test the proposed low-level planning approach in isolation. The experiments were performed on five chains of incidents sampled from the stationary distributions, which have incident counts of {939, 937, 974, 1003, 955} respectively (for a total of 4808 incidents), and are combined to reduce noise.

Non-stationary incident rates: We test how our model reacts to changes in incident rates. We identify different types of scenarios that cause the dynamics of spatial temporal incident occurrence and traffic to change in specific areas of the concerned metropolitan area. We look at rush-hour traffic on weekdays (which affects the center of the county), football game days (which affects the area around the football stadium, typically on Saturdays), and Friday evenings (which affects the downtown area). Then, we synthetically simulate spikes in incident rates in the specific areas at times when the areas are expected to see spikes. To further test whether our approach can deal with sudden spikes, we randomly sample the spikes from a Poisson distribution with a rate that varies between two to five times the historical rates of the regions. We create five different trajectories of incidents with varying incident rates, which have incident counts of {873, 932, 865, 862, 883} respectively (for a total of 4415 incidents).

Responder failures: An important consideration in emergency response is to quickly account for situations where some ambulances might be unavailable due to maintenance and breakdowns. We randomly simulate failures of ambulances lasting 8 hours to understand how our approach deals with such scenarios.

We compare our approach with a baseline policy that has no responder re-allocation. This baseline emulates current policies in use by cities in which responders are statically assigned to depots and rarely move. The initial responder placement is determined using our proposed high-level policy to ensure all the policies begin with similar responder distributions. The baseline uses the same greedy dispatch policy as our approach.

Our experiments and software framework were programmed using python; our code is anonymously hosted at https://anonymous.4open.science/repository/66e0e199-0f2c-41d6-94f4-c31a53d402a2/ for review.

5 RESULTS

Stationary Incident Rates: The results of experiments comparing the baseline policy with the proposed low-level planner on incidents sampled from stationary incident rates are shown in figures 6 and 7. Our first observation is that using the low level planner reduces response times for all region configurations, improving upon the baseline by **7.5** seconds on average. This is a significant improvement in the context of emergency response since it is well-known that paramedic response time affects short-term patient survival [16]. We also observe a significant shift in the distribution of response times, with the upper quartile of the low level results being reduced by approximately **71 seconds** for each region configuration. This reduction in variance indicates that the proposed approach is more consistent. As a result, lesser number of incidents experience large response times.

Non-Stationary Incident Rates We now examine the results of experiments using incidents generated from non-stationary incident distributions, which are shown in figures 8 and 9. Our first observation is that response times generally increase relative to the stationary experiments for both the baseline and the proposed approach. This result is expected since response to incidents sampled from a non-stationary distribution are more difficult to plan for. However, we also observe that our approach is better able to adapt to the varying rates. The low level planner in isolation improves upon the baseline's response times by **18.6 seconds** on average. Introducing the complete hierarchical planner (i.e. both the high level and low level planners) improves the result further, reducing response times by **3 seconds** compared to using only the low level planner, and **21.6 seconds** compared to the baseline. We again observe that the region configuration has a small effect on the efficiency of the proposed approach. This result shows that our approach reduces lower response times irrespective of the manner in which the original problem is divided into regions. Finally, we

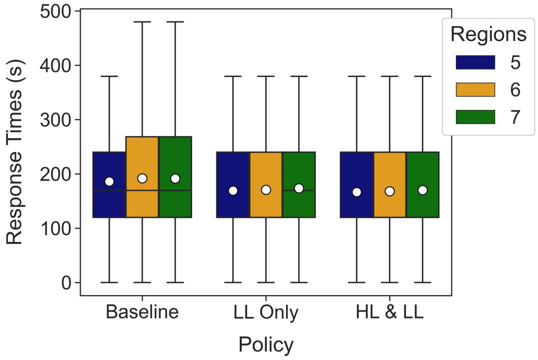


Figure 8: The response time distributions for the baseline, low level planner (LL Only), and complete hierarchical planner (HL & LL) when applied to incidents sampled from a non-stationary rate distribution.

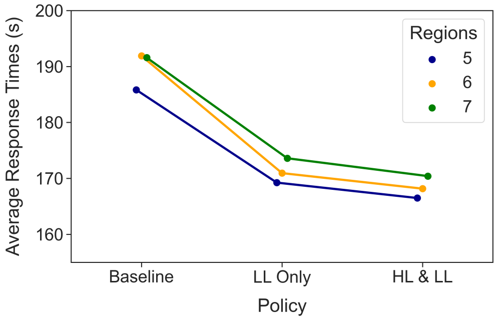


Figure 9: A zoomed in view of the average response times for the baseline, low level planner (LL Only), and complete hierarchical planner (HL & LL) when applied to incidents sampled from a non-stationary rate distribution.

also observe that the variance of the response time distributions achieved by the proposed method is not as low as compared to the stationary experiments, which is likely due to the high strain placed on the system from the non-stationary incident rates.

Responder Failures: Experimental results on the non-stationary incident distribution demonstrate the effectiveness of the hierarchical planner when there are shifts in the spatial distribution of incidents. We now examine its response to equipment failures within the ERM system. Figure 12 illustrates an example (from our experiments) of how the planner can adapt to equipment failures. When a responder in the green region fails, the high level planner determines that imbalance in the spatial distribution of the responders. Intuitively, due to the failure incidents occurring in the upper left cells of the green region could face long response times. Therefore, the planner reallocates a responder from the orange region to the green region.

To examine how equipment failures impact the proposed approach, we simulated several responder failures and compared system performance using our three allocation strategies. We show the results in figures 10 and 11. Naturally, as the number of failures increases, response times increase as there are fewer responders. However, we observe that the proposed approach intelligently allocates the remaining responders to outperform the baseline methods.

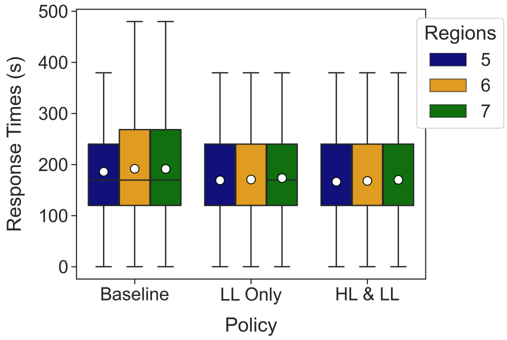


Figure 10: The response time distributions for the baseline, low level planner (LL Only), and complete hierarchical planner (HL & LL) when subjected to increasing numbers of simultaneous equipment failures.

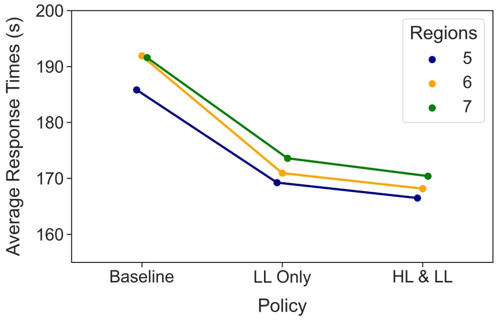


Figure 11: A zoomed in view of the average response times for the baseline, low level planner (LL Only), and complete hierarchical planner (HL & LL) when subjected to increasing numbers of simultaneous equipment failures.

Indeed, when there are three simultaneous failures, using the hierarchical planning improves response times by over **20 seconds** compared to both the baseline and using only the low level planner.

Allocation Computation Times: To be viable, the system must be able to perform allocation computations in a reasonable amount of time. The distribution of computation times can be seen in figure 13. Decisions using the proposed approach take **180.29** seconds on average. Note that this is the time that our system takes to optimize the allocation of responders. Dispatch decisions are greedy and occur instantaneously. Hence, our system can easily be used by first responders on the field without hampering existing operational speed.

6 RELATED WORK

Markov decision processes can be directly solved by using dynamic programming when the transition dynamics of the system are known [13]. Typically, in large-scale resource allocation problems in complex environments like urban areas, the transition dynamics are unknown [18]. To alleviate this, the Simulate-and-Transform (*SimTrans*) algorithm [20] can be used that performs canonical Policy Iteration with an added computation. In order to estimate values (utilities) of states, the algorithm simulates the entire system of incident occurrence and responder dispatch and keeps track of

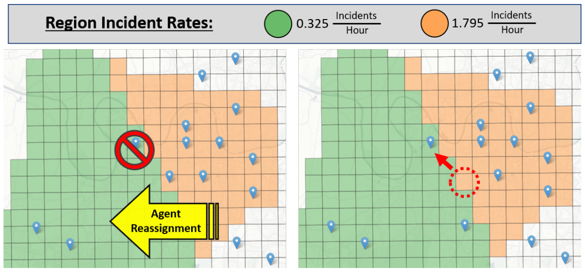


Figure 12: Example of the high level planner resolving an equipment failure. In sub-figure (left), the agent positioned at the depot marked by the red circle in the green region fails, and the high-level planner determines there is an imbalance across regions. In sub-figure (right), we see the planner move an agent from the depot marked by the red dotted circle to the green region to ensure that the upper left of the region can be serviced.

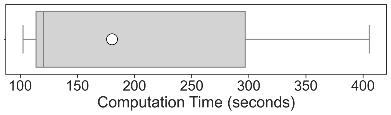


Figure 13: Distribution of Computation times

all states transitions and actions, and gradually builds statistically confident estimates of the transition probabilities.

While such an approach finds a close approximation of the optimal policy (assuming that the estimates of the transition probabilities are close to the true probabilities), this process is extremely slow and unsuited to dynamic environments. As an example, even if a single agent (ambulance in this case) breaks down, the entire process of estimating transition probabilities and learning a policy must be repeated. To better react to dynamic environmental conditions, decentralized and online approaches have been explored [4, 17]. For example, Claes et al. [4] entrust each agent to build its own decision tree and show how computationally cheap models can be used by agents to estimate the actions of other agents as the trees are built.

A somewhat orthogonal approach to solve large-scale MDPs is to use hierarchical planning [9]. Such an approach focuses on learning local policies, known as *macros*, over subsets of the state space. The concept of macro-actions was actually introduced separately from hierarchical planning, as means to reuse a learned mapping from states to actions to solve multiple MDPs when objectives change [23, 27]. Later, the macro-policies were used in hierarchical models to address the issue of large state and action spaces [6, 9].

We also briefly describe how allocation and dispatch are handled in the context of emergency response. First, note that the distinction between allocation and response problems can be hazy since any solution to the allocation problem implicitly creates a policy for response (greedy response based on the allocation) [18]. We use a similar approach in this paper since greedy response satisfies the constraints under which first responders operate. The most commonly used metric for optimizing resource allocation is coverage [3, 7, 28]. Waiting time constraints are often used as constraints in approaches that maximize coverage [19, 26]. Decision-theoretic models have also been widely used to design ERM systems. For example, Keneally et al. [11] model the resource allocation and dispatch problem in ERM as a continuous-time MDP and Mukhopadhyay et al. [20] use a semi-Markovian process. Allocation in ERM can also be addressed by optimizing distance between facilities and demand locations [17], and explicitly optimizing for patient survival [5, 12].

7 CONCLUSION

We have presented a hierarchical planning approach for dynamic resource allocation in city scale cyber-physical system (CPS). We model the overall problem as a Multi-Agent Semi-Markov Decision Process (MSMDP), and show how to leverage the problem's spatial structure to decompose the MSMDP into smaller and tractable sub-problems. We then detail how a hierarchical planner can employ a low-level planner to solve these sub-problems, while a high-level planner identifies situations in which resources must be moved across region lines. Our experiments show that our proposed hierarchical approach offers significant improvements when compared to the state-of-the-art in emergency response planning, as it maintains system fairness while significantly decreasing average incident response times. We also find that it is robust to equipment failure, and is computationally efficient enough to be deployed in the field without hampering existing operational speed.

REFERENCES

[1] Fouad Baouche, Romain Billot, Rochdi Trigui, and Nour-Eddin El Faouzi. 2014. Efficient allocation of electric vehicles charging stations: Optimization model and application to a dense urban network. *IEEE Intelligent Transportation Systems Magazine* 6, 3 (2014), 33–43.

[2] Olfa Chebbi and Jouhaina Chaouachi. 2015. Modeling on-demand transit transportation system using an agent-based approach. In *IFIP International Conference on Computer Information Systems and Industrial Management*. Springer, 316–326.

[3] Richard Church and Charles ReVelle. 1974. The maximal covering location problem. In *Papers of the Regional Science Association*, Vol. 32. Springer-Verlag, 101–118.

[4] Daniel Claes, Frans Oliehoek, Hendrik Baier, and Karl Tuyls. 2017. Decentralised online planning for multi-robot warehouse commissioning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 492–500.

[5] Erhan Erkut, Armann Ingolfsson, and Güneş Erdoğan. 2008. Ambulance location for maximum survival. 55, 1 (2008), 42–58.

[6] J-P Forestier and Pravin Varaiya. 1978. Multilayer control of large Markov chains. *IEEE Trans. Automat. Control* 23, 2 (1978), 298–305.

[7] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. 1997. Solving an ambulance location model by tabu search. 5, 2 (1997), 75–88.

[8] Stefan Gössling. 2020. Integrating e-scooters in urban transportation: Problems, policies, and the prospect of system change. *Transportation Research Part D: Transport and Environment* 79 (2020), 102230.

[9] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas L Dean, and Craig Boutilier. 2013. Hierarchical solution of Markov decision processes using macro-actions. *arXiv preprint arXiv:1301.7381* (2013).

[10] David G Kendall. 1953. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain. *The Annals of Mathematical Statistics* (1953), 338–354.

[11] Sean K Keneally, Matthew J Robbins, and Brian J Lunday. 2016. A markov decision process model for the optimal dispatch of military medical evacuation assets. *Health Care Management Science* 19, 2 (2016), 111–129.

[12] V. A. Knight, P. R. Harper, and L. Smith. 2012. Ambulance allocation for maximal survival with heterogeneous outcome measures. 40, 6 (2012), 918–926.

[13] Mykel J Kochenderfer. 2015. *Decision Making Under Uncertainty: Theory and Application*. MIT Press.

[14] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European Conference on Machine Learning (ECML)*. Springer, 282–293.

[15] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1. Oakland, CA, USA, 281–297.

[16] Jonathan D Mayer. 1979. Emergency medical service: delays, response time and survival. *Medical Care* (1979), 818–827.

[17] Ayan Mukhopadhyay, Geoffrey Pettet, Chinmaya Samal, Abhishek Dubey, and Yevgeniy Vorobeychik. 2019. An Online Decision-theoretic Pipeline for Responder Dispatch. In *International Conference on Cyber-Physical Systems (ICCPS)* (Montreal, Quebec, Canada). ACM, 185–196. https://doi.org/10.1145/3302509.3311055

[18] Ayan Mukhopadhyay, Geoffrey Pettet, Sayyed Vazirizade, Yevgeniy Vorobeychik, Mykel Kochenderfer, and Abhishek Dubey. 2020. A Review of Emergency Incident Prediction, Resource Allocation and Dispatch Models. arXiv:2006.04200 [cs.AI]

[19] Ayan Mukhopadhyay, Yevgeniy Vorobeychik, Abhishek Dubey, and Gautam Biswas. 2017. Prioritized Allocation of Emergency Responders based on a Continuous-Time Incident Prediction Model. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 168–177.

[20] Ayan Mukhopadhyay, Zilin Wang, and Yevgeniy Vorobeychik. 2018. A Decision Theoretic Framework for Emergency Responder Dispatch. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 588–596.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[22] Geoffrey Pettet, Ayan Mukhopadhyay, Mykel Kochenderfer, Yevgeniy Vorobeychik, and Abhishek Dubey. 2020. On algorithmic decision procedures in emergency response systems in smart and connected communities. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[23] Doina Precup and Richard S Sutton. 1998. Multi-time models for temporally abstract planning. In *Advances in Neural Information Processing Systems (NIPS)*. 1050–1056.

[24] Khashayar Rohanimanesh and Sridhar Mahadevan. 2003. Learning to take concurrent actions. In *Advances in Neural Information Processing Systems (NIPS)*. 1651–1658.

[25] John F Shortle, James M Thompson, Donald Gross, and Carl M Harris. 2018. *Fundamentals of Queueing Theory*. Wiley.

[26] Francisco Silva and Daniel Serra. 2008. Locating emergency services with different priorities: the priority queuing covering location problem. *Journal of the Operational Research Society* 59, 9 (2008), 1229–1238.

[27] Richard S Sutton. 1995. TD models: Modeling the world at a mixture of time scales. In *International Conference on Machine Learning (ICML)*. Elsevier, 531–539.

[28] Constantine Toregas, Ralph Swain, Charles ReVelle, and Lawrence Bergman. 1971. The Location of Emergency Service Facilities. 19 (1971), 1363–1373.

Appendices

A SIMULATOR

Here we detail our simulation of the emergency response system and its environment which is used by the low-level planner to estimate the effect of various actions. As shown in figure 2, our system state at some time t is captured by a queue of active incidents I^t and agent states Λ. I^t is the queue of incidents that have been reported but not yet serviced, and allows the system to keep track of any incidents that couldn't be immediately responded to. The state of each agent $\lambda_j \in \Lambda$ consists of the agent's current location p_j^t, status u_j^t destination g_j^t, assigned region r_j^t, and assigned depot d_j^t. Each agent can be in several different internal states (represented by u_j^t), including *waiting* (waiting at a depot), *in_transit* (moving to a new depot and not in emergency response mode), *responding* (the agent has been dispatched to an incident and is moving to its location), and *servicing* (the agent is currently servicing an incident). These states dictate how the agent is updated when moving the simulator forward in time, as discussed below.

To determine the travel times between locations in the environment, the simulator uses a traffic router. In our experiments, we use a Euclidean distance based router, which assumes all agents travel in straight lines between locations. If deployed to a real world system, a more advanced router can be used that uses information about the roadway network and current traffic conditions to accurately estimate travel times.

Our simulator is designed as a discrete event simulator, meaning that the state is only updated at discrete time steps when interesting events occur. These events include incident occurrence, re-allocation planning steps, and responders becoming available for dispatch. Between these events, the system evolves based on prescribed rules. Using a discrete event simulator saves on valuable computation time as compared to a continuous time simulator.

At each time step when the simulator is called, the system's state is updated to the current time of interest. First, if the current event of interest is an incident occurrence, it is added to the active incidents queue I^t. Then each agent's state and locations are updated to where they would be at the given time, which depends on their current state. For example, agents that are in the *waiting* state stay at the same position, while agents that are *responding* or *in_transit* will check to see if they have reached their destination. If they have, they will update their state to *servicing* or *waiting* respectively and update their locations. If they have not reached their destination, they interpolate their current location using the travel model. If an agent is in the *servicing* state and finishes servicing an incident, it will enter the *in_transit* state and set its destination g_j^t to its assigned depot.

After the state is updated, a planner has several actuation's available to control the system. The *Dispatch(λ_j, incident)* function will dispatch the agent λ_j to the given incident which is in I^t. Assuming the responder is available, the system sets λ_j's destination g_j^t to the incident's location, and it's status u_j^t is set to *responding*. The incident is also removed from I^t since it is being serviced, and the response time is returned to the planner for evaluation. The planner can also change the allocation of the agents. *AssignRegion(λ_j, r_j)* assigns agent λ_j to region r_j by updating λ_j's r_j^t. *AssignDepot(λ_j, d_j)* similarly assigns agent λ_j to depot d_j by updating λ_j's d_j^t and setting its destination g_j^t to the depots location. These functions allow a planner to try different allocations and simulate various dispatching decisions.

B NOTATION

We provide notation definitions for convenience in table 2.

Table 2: Notation lookup table

Symbol	Definition
Λ	Set of agents
D	Set of depots
$C(d)$	Capacity of depot d
G	Set of cells
R	Set of regions
S	State space
A	Action space
P	State transition function
T	Temporal transition distribution
α	Discount factor
$\rho(s,a)$	Reward function given action a taken in state s
$\mathcal{A}$	Joint agent action space
$\mathcal{T}$	Termination scheme
s^t	Particular state at time t
I^t	Set of cell indices waiting to be serviced
$Q(\Lambda)$	Set of agent state information
p_j^t	Position of agent j
g_j^t	Destination of agent j
u_j^t	Current status of agent j
s_i, s_j	Individual states
σ	Action recommendation set
η	Service Rate
γ_g	Incident rate at cell g
t_h	Time since beginning of planning horizon
$t_r(s,a)$	Response time to an incident given action a in state s