Hierarchical Planning for Dynamic Resource Allocation in Smart and Connected Communities

GEOFFREY PETTET, AYAN MUKHOPADHYAY, MYKEL J. KOCHENDERFER, and ABHISHEK DUBEY

Resource allocation under uncertainty is a classical problem in city-scale cyber-physical systems. Consider emergency response as an example; urban planners and first responders optimize the location of ambulances to minimize expected response times to incidents such as road accidents. Typically, such problems deal with sequential decision-making under uncertainty and can be modeled as Markov (or semi-Markov) decision processes. The goal of the decision-maker is to learn a mapping from states to actions that can maximize expected rewards. While online, offline, and decentralized approaches have been proposed to tackle such problems, scalability remains a challenge for real-world use-cases. We present a general approach to hierarchical planning that leverages structure in city-level CPS problems for resource allocation. We use emergency response as a case study and show how a large resource allocation problem can be split into smaller problems. We then use Monte-Carlo planning for solving the smaller problems and managing the interaction between them. Finally, we use data from Nashville, Tennessee, 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.

1 INTRODUCTION

Dynamic resource allocation (DRA) in anticipation of uncertain demand is a canonical problem in city-scale cyber-physical systems (CPS) [30]. 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 problem domain. This optimization requires design and development of procedures that can estimate how the system will evolve and evaluate the long-term value of actions. DRA manifests in many problems at the intersection of urban management, CPS, and multi-agent systems. These include emergency response management (ERM) for ambulances and fire-trucks [31], allocating helper trucks [43], designing on-demand transit [4], and positioning electric scooters [16]. All such use-cases typically deal with incidents of interest which correspond to specific calls for service. In anticipation of such calls, planners proactively optimize over allocating resources. The allocation problem can be modeled as a sequential decision-making problem under uncertainty, which can be solved to maximize a domain-specific utility function. For example, the maximizing the total demand that can be serviced or minimizing the expected response time to incidents are commonly used objectives.

While we present a general approach for dynamic resource allocation in urban areas, we focus particularly on the problem of emergency response as a case-study since it is a critical problem faced by communities across the globe. With road accidents accounting for 1.25 million deaths globally and 240 million emergency medical services (EMS) calls made in the USA each year, there is a critical need for a proactive and effective response to these emergencies [1, 31]. In addition to responding to frequent incidents each day, emergency response management (ERM) addresses large-scale disasters due to natural hazards (climate-driven disasters caused more than $90 billion of losses in the US in 2018 [12]) and man-made attacks. The lack of active and timely response to emergencies constitutes a threat to human lives and has resulted in mounting costs. Minimizing the

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.

Authors' address: Geoffrey Pettet, geoffrey.a.pettet@vanderbilt.edu; Ayan Mukhopadhyay, ayan.mukhopadhyay@vanderbilt.edu; Mykel J. Kochenderfer, mykel@stanford.edu; Abhishek Dubey, abhishek.dubey@vanderbilt.edu.

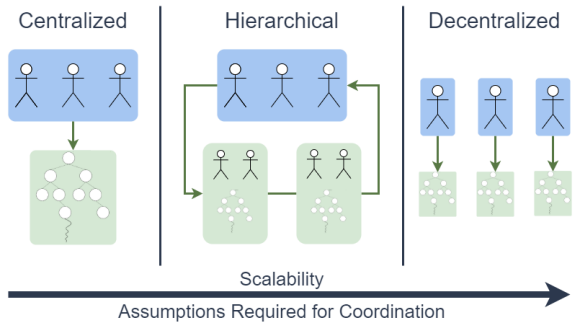


Fig. 1. A spectrum of approaches to solve a dynamic resource allocation problem under uncertainty. A completely centralized approach uses a monolithic state representation. In a completely decentralized approach, each agent simulates what other agents do and performs its own action estimation. Our hierarchical approach segments the planning problem into sub-problems to improve scalability without agents estimating other agents' actions.

time to respond to emergency incidents is critical in emergency response. Therefore, governments and private agencies often try to proactively optimize ambulance locations while considering constraints on the number of available ambulances and locations where they can be stationed. In addition, they address the problem of resource dispatch, in which agents[1] must be selected and asked to spatially move from their location to the location of the demand to address the task at hand. For example, ambulances must move to the scene of the incidents to provide assistance to patients. Effectively, ERM pipelines are an important example of human-in-the-loop CPS (H-CPS), which introduces problem-specific structure and constraints.

State of the art: DRA and dispatch problems are typically modeled as Markov decision processes (MDP) [31, 33]. The decision-maker's goal is to find an optimal *policy*, which is a mapping between system states and actions to be taken. There is a broad spectrum of approaches available to address resource allocation under uncertainty as shown in figure 1. In our previous work, we applied the extremes of this continuum to ERM, with each approach having strengths and weaknesses [30, 33, 35]. The most direct approach is to model the state of the MDP as a monolithic entity in the MDP that captures the entire system in consideration, shown on the left in figure 1. When real-world problems are modeled as MDPs, state transitions are typically unknown. The standard method to address this is to use a simulator (a model of the world) to estimate an empirical distribution over the state transitions [33], and use this to learn a policy using the well-known policy iteration algorithm [23]. Unfortunately, this offline approach does not scale to realistic problem settings [30]. Another method is to use an online solution like Monte-Carlo tree search (MCTS) [30]. While adaptable to dynamic environments, this approach still suffers from poor scalability, taking too long to converge for realistic scenarios.

On the other side of the spectrum is 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. [6] 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'

[1]The unit of decision-making actions are referred to as agents (e.g., ambulances)

actions limits scalability and therefore decentralized approaches rely on inexpensive 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 available information during planning are not suitable for emergency response in urban areas, especially when fast and effective response is critical.

We explore hierarchical planning [17], 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 spatial 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 to partition and better manage infrastructure. We design a principled algorithmic approach that partitions the spatial area under consideration into smaller areas. 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 because agents constrained in one region might be needed in the other region. We show how hierarchical planning can be used to facilitate transfer of agents across regions. A 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. A low-level planner, called the intra-region planner, addresses allocation and dispatch within a region.

A challenge with hierarchical planning is that the high-level planner must be able to estimate rewards further along the planning pipeline in order to make decisions. However, such rewards can only be computed *after* the low-level planner optimizes its objective function; this defeats the purpose of hierarchical planning because it does not segregate the overall planning problem between two different levels. We leverage the structure of DRA and dispatch problems to address this challenge.

Contributions: **1)** We propose a hierarchical planning approach for resource allocation under uncertainty for smart and connected communities that scales significantly better than prior approaches. The key idea in our approach comes from the concept of *macros* in decision-theoretic systems [17], 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 partition the overall decision-problem into sub-problems. We create a low-level planner that 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 propose two models, a data-driven surrogate model and a queue-based approximation, that can estimate rewards to aid the high-level planner. In practice, responders can remain busy even after they leave the scene of the incident (e.g., ambulances transport victims of accidents to hospitals). Unlike prior work [30, 36], the data-driven approach we propose can naturally accommodate such drop-off times. **5)** We use real-world emergency response data from Nashville, Tennessee, a major metropolitan area in the United States, to evaluate our approach and show that it performs better than state-of-the-art approaches both in terms of efficiency, scalability, and robustness to agent failures.

The paper is organized as follows. We present a general decision-theoretic formulation for spatial-temporal resource allocation under uncertainty in section 2. Section 3 describes the overall approach, the high-level, and the low-level planner. Section 4 details the implementation of our decision support system for ERM responder allocation. We present our experimental design in

Table 1. 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
p_j	Number of agents assigned to region r_j
γ_j	Total incident rate in region r_j
$w_j(p_j, \gamma_j)$	Expected waiting time for incidents in region r_j
D_j	Set of depots in region r_j
G_j	Set of cells in region r_j

section 5 and the results in section 6. We present related work in section 7 and summarize the paper in section 8. Table 1 summarizes the notation used throughout the paper.

2 PROBLEM FORMULATION

Resource allocation in anticipation of spatial-temporal requests is a common problem in urban areas. For example, ambulances respond to road accidents and emergency calls, helper trucks respond to vehicle failures, taxis provide service to customers, and fire trucks respond to urban fires. We refer to responders as *agents* to be consistent with the terminology used in multi-agent systems [44]. Once an incident is reported, agents are dispatched to the scene of the incident. The decision to select an agent to be dispatched can either be performed by a human expert (e.g., dispatching towing trucks), human-algorithm collaboration (e.g., dispatching ambulances), or completely by an algorithmic approach (e.g., taxis). If no free agent is available, the incident typically enters a waiting queue, and is responded to when an agent becomes free. Each agent is typically housed at specific locations distributed throughout the spatial area under consideration (these could be fire stations or rented parking spots, for example). The number of such locations can vary by the type of incident and agent in consideration. For example, while taxis can wait at arbitrary locations or move around areas with high historical demand, ambulances are typically housed at rented parking

lots or designated stations. We refer to such locations as *depots*. Once an agent finishes servicing an incident (

which might involve transporting the victim of the incident to a nearby hospital), it becomes available for dispatch. If there are no pending incidents, 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 agent to dispatch once an incident occurs (dispatching action) and (2) which depots to send the agents to in anticipation of future demand (allocation action). Next, we introduce the assumptions we make and the notation we use for problem formulation. While we present a formulation that is broadly application to resource allocation problems under uncertainty in urban areas, we use emergency response as a case study to throughout to provide concrete examples and use-cases.

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, a set of agents Λ that need to be allocated across these cells and dispatched to demand points, and a travel model that describes how the agents move throughout G. 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 $C(d)$, which is the 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 finite decision making states that evolve in discrete time. For example, an agent 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, which makes temporal transitions between decision-making states non-memoryless. This causes the underlying stochastic process governing 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) [38], 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 [38]. We focus on asynchronous termination, where actions for a particular agent are chosen as and when the agent completes its last assigned action.[2]

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 its assigned

[2]Different termination schemes are discussed in the theoretical analysis by Rohanimanesh and Mahadevan [38].

depot, all observed at time t. A diagram of the state is shown in figure 4 and discussed in detail in section 4.

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 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.

The manner in which agents are dispatched to the scene of the incidents varies with the type of incidents. For example, a key aspect of emergency response is that if any free agents are available when an incident is reported, then the nearest one must be greedily 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 [29, 31, 35]. Taxis, on the other hand, can optimize dispatch based on long-term rewards. We focus on emergency response in our experiments and validate the approach using data collected from ambulances. As a result, 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 resource allocation system model evolves through several stochastic processes. Incidents occur at different points in time and space governed by some arrival distribution. We assume that 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 [31]. 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 agents. 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 an 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. The benefits of allocation actions 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.

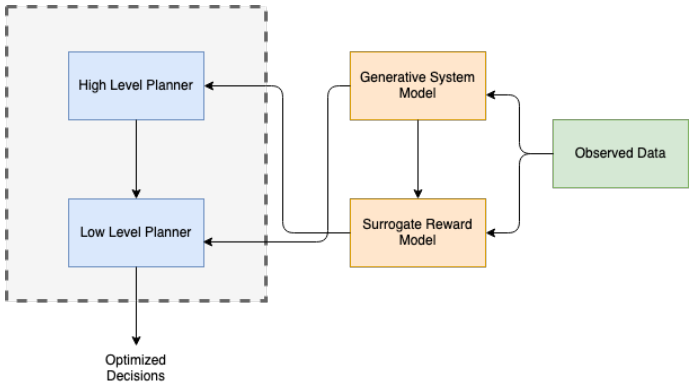


Fig. 2.
An overview of the proposed planning approach. We use the observed data to learn a generative model over when and where incidents occur. The generative model lets us simulate agent behavior, which aids the creation of a surrogate model for the high-level planner to segment the overall problem into a set of smaller problems. The low-level planner tackles each sub-problem independently. As we show, the surrogate reward model can also be estimated based on closed-form expressions of waiting times based on a queuing model.

3 APPROACH

Our approach to spatiotemporal resource allocation divides planning tasks into a two stage hierarchy: "high-level" and "low-level" planning. First, the high-level planning task is to divide the overall decision-theoretic problem into smaller sub-problems. Our high-level planner accomplishes this by creating meaningful spatial clusters (which we call regions), and then optimizing the distribution of agents among these regions.

In order to do so, it is imperative for the high-level planner to assess the quality of a specific distribution, which requires the low-level planner itself. Indeed, the actual reward generated from an allocation depends on how the low-level planner optimizes the distribution of responders *within each region*. Clearly, this dependency is detrimental to creating an approach that seeks to divide the overall planning problem into two stages to achieve scalability. In order to tackle this challenge, we learn a surrogate model over waiting times that can be used by the high-level planner to estimate rewards. The high-level planner can perform inference using the model to estimate rewards as it optimizes over 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. We assume the availability of an integrated simulation framework that models the dynamics of agents as they travel throughout the environment, as well as a probabilistic generative model of incident occurrence learned from historical incident data. Our simulation framework and incident model are described in detail in section 4.

Segmenting allocation into smaller sub-problems significantly reduces its complexity compared to a centralized problem structure. Consider an example city with $|\Lambda| = 20$ agents and $|D| = 30$ depots, each of which can hold one agent. With a centralized approach, any agent can go to any depot, so there are $Permutations(|D|, |\Lambda|) = \frac{|D|!}{(|D|-|\Lambda|)!} = \frac{30!}{10!} = 7.31 \times 10^{25}$ possible assignments at each decision epoch. Now consider a hierarchical structure where the problem is split into 5 evenly sized sub-problems, so $|\Lambda_h| = 4$ and $|D_h| = 6$ for each region. There are now $P(|D_h|, |\Lambda_h|) = 360$ possible allocations in each region, so there are $360 * 5 = 1800$ possible actions across all regions.

This is a reduction in complexity of about 22 orders of magnitude compared to the centralized problem, at the cost of abstracting the interactions between agents in different regions through the high level planner. Alternatively, a decentralized approach in which each responder plans only its own actions reduces the complexity further to only $|D| = 30$ possible allocations for each agent [35]. However, this reduction in complexity comes at a cost as each agent must make assumptions regarding other agent behavior, which can lead to sub-optimal planning. Hierarchical planning offers a balance between decentralized and centralized planning by having agents accurately model other nearby agents' behavior while having reasonable decision complexity.

An important consideration when designing approaches for resource allocation under uncertainty in city-scale CPS problems is adapting to the dynamic environments 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 an agent 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 agents can be optimized to best adapt 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 tractable sub-problems. A natural decomposition for spatiotemporal resource allocation is to divide the overall problem's spatial area into discrete regions, and perform allocation separately for each region. This allows the low-level allocation planner to focus on evaluating potential interactions between agents that are nearby and therefore likely to interact. The first goal of the high-level planner is to define these spatial regions. Consider that the high-level planner seeks to divide the overall problem in to m regions, denoted by the set $R = \{r_1, r_2, \ldots, r_m\}$, where $r_i \in R$ denotes the ith region. To achieve this goal, we use historical data of incident occurrence to partition the set G into clusters.

Intuitively, we want to create regions based on *hotspots* of incident occurrence [3]. Recall that our goal is to identify spatial areas in which planning (the allocation and distribution of agents) can be performed independently. By identifying spatial clusters, we achieve the following: 1) areas close to each other that have similar patterns of incident occurrence are grouped together and 2) areas of high incident occurrence that are further apart get segregated from each other. While any standard spatial clustering approach can be used to achieve this goal, we use the well-known k-means algorithm in our analysis [25]. The k-means algorithm partitions a given set of points in $\mathbb{R}^d$ into k clusters such that each point belongs to the cluster with the nearest cluster center (or mean). While the problem is known to be NP-hard even when $d = 2$ (our case) [27], heuristic-based iterative approaches can be used to achieve promising solutions. Typically, the solution process repeatedly performs two steps after initializing an initial set of clusters. In the first step, each point is assigned to cluster whose center is the closest to the point. Then, given an assignment, the centers of the clusters are computed again. The process is repeated until convergence. We use such clusters as separate sub-problems for low-level planning.

The high-level planner's second task is to determine the distribution of agents across these spatial regions. If the regions are homogeneous, agents could simply be split evenly across them. In practice, however, regions will differ in properties such as size, incident rate, and depot distributions, all of which impact the number of responders needed to cover each region. For example, the number of agents assigned to cover a dense downtown area should likely be different from sparsely populated suburbs. Recall that the overall goal of the parent MSMDP is to reduce the expected incident 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. If we assume that incidents are homogeneous, meaning that the time taken by agents to service incidents follows the same distribution, then the sole criterion that a planner needs to optimize is overall travel time of the agents to incidents, which we refer to as *waiting times* (achieving zero waiting times is clearly infeasible in practice, so we seek to minimize waiting times). Therefore, the high-level planner seeks to distribute agents to different regions such that the expected incident waiting time is minimized.

We denote the expected waiting time for incidents in region $r_j \in R$ by $w_j(p_j, \gamma_j)$, where p_j is the number of responders assigned to the region r_j and γ_j is the total incident rate across r_J. Since the arrival process is assumed to be Poisson distributed, 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 that checks if cell g_i belongs to region $r_j \in R$. We consider two approaches to estimate $w_j(p_j, \gamma_j)$: a *queuing model* that approximates the system using an *m/m/c* queuing formulation, and a *surrogate model* that uses machine learning. We detail these models in sections 3.1.1 and 3.1.2, and then describe our high-level agent distribution algorithm, which uses an iterative greedy approach to assign agents across regions, in section 3.1.3.

3.1.1 Queuing Model. One approach to model waiting times in a region is using 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 [32]. One potential issue with using well-known queuing models to estimate waiting times in emergency response is that travel times are not memoryless. In this approach, we use an approximation from prior work to tackle this problem [32]. 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 [20]), where $c = p_j$.

Let the average service time be T_s and let $\mu = 1/T_s$ denote the mean service rate. Then, the mean waiting time $w_j(p_j, \gamma_j)$ is [39]:

$$w_j(p_j, \gamma_j) = \frac{P_0(\frac{\gamma_j}{\mu})^{p_j}\gamma_j}{c!(1-\rho)^2 c}$$

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

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

While this approach is straightforward, computationally efficient, and works 'out of the box' with any city of interest, it it abstracts away many environmental dynamics that can effect response times. Factors such as the agents' allocation within a region, travel times due to traffic, and behaviors such as dropping off patients at the hospital can all affect the response times observed in the real world. To capture such factors, we explore an alternative method using machine learning to create a surrogate model over expected waiting times.

3.1.2 Surrogate Model. Another approach to estimate waiting times in a region is to use simulated data to learn a model over waiting times conditional on a set of relevant covariates. We simulate emergency response in a region under various conditions (namely the number of agents assigned to the region and the incident rate in the region) and record the simulated incident response times. After generating many such samples, we use a supervised learning approach to fit a function to

maximize the likelihood of the simulated data. The trained model can then be used to estimate waiting times for a region given a number of responders. The advantage of this approach is that it captures subtle environmental dynamics that are ignored by the queue model, such as the travel times of agents as they respond to incidents as well as their location within the region due to factors such as depot locations and dropping off patients at hospitals.

The first step to creating the surrogate model is generating response time samples by simulating emergency response in each region with different incident rates γ_j and available agents. We sample incidents using the model described in section 4, which uses historical rates of incidents to create a sampling distribution. We refer to sampled incidents as *chains*. For each such chain, region r_j, and potential number of agents $p_j \in [1 : |D_j|]$ (where D_j is the set of depots in region r_j), we simulate response to the incidents occurring within r_j using the simulation framework described in section 4. This provides us with labeled training data where each observation captures the response time (output) given the number of responders and the rate of incident occurrence (inputs).

A crucial factor that affects simulated response times is the initial location of the agents within each region. Recall that in our proposed framework, the location of the agents is optimized by the low-level planner which is naturally infeasible to use for every step during sampling. As a result, we solve the standard p-median problem [9], which is commonly applied to ambulance allocation, to determine the initial conditions of the simulation. The objective of the p-median formulation is to locate p_j agents (in region r_j) such that the average demand-weighted distance between demand points and their nearest agent is minimized. Formally, we solve the following optimization problem to allocate the agents to depots:

$$\min \sum_{i=1}^{|G_j|} \sum_{k=1}^{|D_j|} a_i d_{ik} Y_{ik} \tag{2a}$$

$$\text{s.t.} \sum_{k=1}^{|D_j|} Y_{ik} = 1, \quad \forall i \in \{1, \ldots, |G_j|\} \tag{2b}$$

$$\sum_{k=1}^{|D_j|} X_k = p \tag{2c}$$

$$Y_{ik} \leq X_k, \quad \forall i \in \{1, \ldots, |G_j|\}, \forall k \in \{1, \ldots, |D_j|\} \tag{2d}$$

$$X_k, Y_{ik} \in \{0, 1\}, \quad \forall i \in \{1, \ldots, |G_j|\}, \forall j \in \{1, \ldots, |D_j|\} \tag{2e}$$

where G_j is the set of cells in region r_j, D_j is the set of depots in r_j, p_j is the number of agents to be located in r_j, a_i is the likelihood of accident occurrence at cell $g_i \in G_j$, and d_{ik} is the distance between cell $g_i \in G_j$ and location $d_k \in D_j$. Y_{ik} and X_k are two sets of decision variables; $X_k = 1$ if an agent is located at $d_k \in D_j$ and 0 otherwise, and $Y_{ik} = 1$ if cell $g_i \in G_j$ is covered by an agent located at $d_k \in D_j$ (i.e. the agent at d_k is the nearest placed agent to g_i) and 0 otherwise.

The p-median problem is known to be NP-hard [19], therefore heuristic methods are employed to find approximate solutions in practice. We use the Greedy-Add algorithm [7] to optimize the locations of agents. We show the algorithm in Algorithm 12. First, we initialize the iteration counter z and the set of allocated agent locations X_z to the empty set (step 1). Then, as long as there are responders awaiting allocation, we iterate through the following loop: (1) update counter z to the current iteration (step 3), (2) for each potential location not already in the allocation, compute the p-median score (equation 2a) of the allocation which includes the potential location (steps 5 - 7), and (3) find the location that minimizes the p-median score (step 9) and add it to the set of allocated agent locations (step 10). After locating the agents to depots within the region, we simulate

Algorithm 1: Greedy-Add Algorithm

input : Region cells G_j, region depots D_j, cell incident likelihoods $a_i \ \forall g_i \in G_j$, cell to depot distances $d(i,k) \ \forall g_i \in G_j, \forall d_k \in D_j$, number of agents p_j

output : Agent depot assignments X

Initialize $z := 0, X_z := \emptyset$;
while $z < p_j$ **do**
  $z := z + 1$;
  for location $d_{k'} \in D_j$, where $k' \notin X_{z-1}$ **do**
    $X'_z := X_{z-1} \cup d_{k'}$;
    Find nearest facilities $y_i \ \forall g_i \in G_j$, where $y_{g_i} \in X'_z$;
    Compute $U^z_{k'} := \sum_{g_i \in G_j} a_i d(g_i, y_i)$;
  end
  Best location $d^*_k := \text{argmin}_k \ U^z_k$;
  $X_z := X_{z-1} \cup k^*$;
  Return X_z
end

emergency response using greedy dispatch and record the average response times $w(r_j, p_j, \lambda_j)$. The complexity of this algorithm is $O(p_j|D_j||G_j|)$, as assigning each agent requires evaluating each potential depot in D_j, which requires summing over the weighted distances between each cell in G_j and its nearest populated depot.

Given a set of samples of waiting times (w), agent allocations (p), and incident rates (γ), the second step in creating the surrogate model is to learn an estimator over p given p and γ. We learn a different model for each region to capture any latent features that can effect response times such as the region's depot distribution and roadway network. We use random forest regression [2]. Random forests are an ensemble learning method that is based on constructing a several decision trees at training time. During inference, the average prediction of the trained trees is used as output (for regression problems).

3.1.3 Optimization. Given estimated waiting times for incidents in each region, the high-level planner seeks to minimize the cumulative response times across all regions. The optimization problem can be represented as

$$\min_{p} \sum_{j=1}^{m} w_j(p_j) \tag{3a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} p_i = |\Lambda| \tag{3b}$$

$$p_i \in \mathbb{Z}^{0+} \ \forall i \in \{1, \ldots, m\} \tag{3c}$$

The objective function in mathematical program 3 is non-linear and non-convex. We use an iterative greedy approach shown in algorithm 2. We begin by sorting regions according to total arrival rates. Let this sorted list be R_s. Then, we assign agents iteratively to regions in order of decreasing arrival rates (step 3). After assigning each agent to a region $r_j \in R$, we compare the overall service rate (p_j times the mean service rate by one agent) 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

Algorithm 2: High-Level Planner

input : Sorted Regions R_s, Arrival Rates $\{\gamma_1, \gamma_2, \ldots, \gamma_m\}$, Service Rate η
output: Responder Allocation $P = \{p_1, p_2, \ldots, p_m\}$

1 assigned := 0, i := 0, $J := \emptyset$;
2 **while** assigned $\leq$ | Λ | **and** $i \leq m$ **do**
3 $\quad p_i := p_i + 1$;
4 $\quad$ assigned := assigned + 1;
5 $\quad$ **if** $\eta \times (p_i) \geq \sum_{g_i \in G} \mathbb{1}(g_i \in r)\gamma_i$ **then**
6 $\quad\quad i := i + 1$;
7 $\quad$ **end**
8 **end**
9 **while** assigned $\leq$ | Λ | **do**
10 $\quad$ **for** $i \in [1, m]$ **do**
11 $\quad\quad J[i] := w_i(p_i, \gamma_i) - w_i(p_i + 1, \gamma_i)$;
12 $\quad$ **end**
13 $\quad r^* := \arg\max_{i \in [1,m]} J[i]$;
14 $\quad p_{r^*} := p_{r^*} + 1$;
15 $\quad$ assigned := assigned + 1;
16 **end**

assigned agents in this manner, we check if there are surplus agents (step 9). The surplus agents 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 13). Then, we assign an agent to the region that gains the most (in terms of reduction in waiting times) by the assignment. The complexity of the algorithm is $O(|\Lambda| m \xi)$, where ξ is the complexity of the wait time estimation method, as the algorithm computes the potential wait times $w_j(r_j, p_j, \lambda_j)$ from adding an agent to each region when assigning said agent. When using the queuing model, the overall complexity is $O(|\Lambda|^2 m)$, since equation P_0 sums over all assigned agents to estimate w_j. The overall complexity when using the random forest surrogate model is $O(|\Lambda| m \epsilon \beta)$ where ϵ is the number of trees in the forest and β is the maximum tree depth.

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 4 and described in detail in section 4). 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 [24], which defines the score of a node n as

$$\mathrm{UCB}(n) = \overline{u}(n) + c\sqrt{\frac{\log(\mathrm{visits}(n))}{\mathrm{visits}(n')}} \tag{4}$$

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

Algorithm 3: 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 $\widetilde{\mathcal{A}} := \emptyset$;
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 $a \in A$ **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**

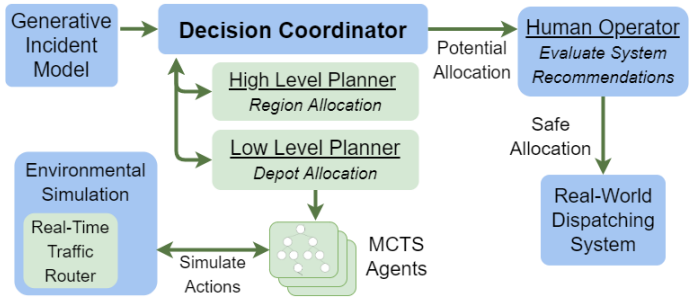


Fig. 3. Emergency Response Decision Support Framework

to process each. We then average the scores across trees for each potential allocation action to determine the optimal action.

Our low-level planning approach is shown in algorithm 3. 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 INTEGRATION FRAMEWORK

Figure 3 shows a schematic representation of our decision support system. Realizing a such as system for online emergency responder allocation requires a framework of interconnected processes, including

- A traffic routing model to support routing requests (section 4.1).
- A probabilistic generative model of incident occurrence (section 4.2).
- A model of the ERM system and its environment, including the dynamics of responders, depot locations, and hospital locations (section 4.3).
- A simulation of the ERM system built on the above components (section 4.4).
- A hierarchical decision process that makes allocation and dispatching recommendations based on the current state of the environment, responder locations, and projected incident distributions (section 3).
- A human operator to access the planning mechanism and act as an interfaced with a real-world computer-aided dispatch system.

We discussed our hierarchical approach to planning in section 3; in this section we detail the rest of our decision framework.

4.1 Travel Model

We consider two travel time models in our implementation. The first model is based on the Euclidean distance between two points of interest (the centers of the cells in the grid). We also develop a more principled travel model that uses contraction hierarchies [14] and an open source routing machine engine [18] to lookup travel times at different times of the day from the center of each cell in the spatial grid to other cells. We collect such travel times across a week. The final travel model considers the median of the accumulated travel times and develops a lookup table which can be used by the overall planning process to query the travel times. This approach is better than a Euclidean distance based travel time as this considers the average road congestion and the maximum travel speed across the road. The pipeline we propose and our framework is flexible to accommodate other travel time models; for example, a modular component that estimates travel times based on advanced graphical neural networks can be used with a richer set of covariates to provide estimates that are sensitive to time and weather.

4.2 Incident Prediction

Recall that we need samples of incidents for the low-level planner as well as learning the surrogate model for the high-level planner. We assume that incidents are generated by a Poisson distribution. A Poisson model has been widely used to model the occurrence of accidents [31]. The Poisson distribution is a discrete probability distribution over the number of events in a fixed interval of time. We learn a separate Poisson model for each cell $g_i \in G$. The rate parameter of each model can be learned by maximizing the likelihood of historical data in the respective cell. The learned Poisson model can then be used to sample incidents in a given period of time.

4.3 ERM System Model

As shown in figure 4, our system state at 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 could not 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

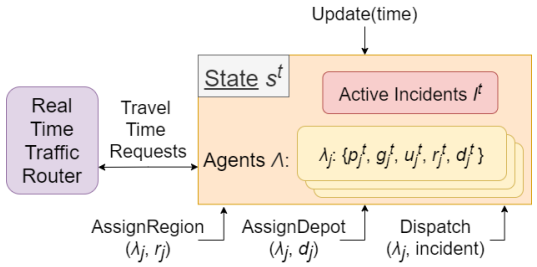


Fig. 4. System State and Actions.

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.

4.4 Simulation Framework

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 its 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.

5 EXPERIMENTS

We evaluate the proposed hierarchical framework's effectiveness on emergency response data obtained from Nashville, Tennessee, a major metropolitan area in the United States, with a population of approximately 700,000. We use historical incident data, depot locations, and operational data provided by the Nashville Fire Department [8]. We construct a grid representation of the city using 1×1 mile square cells. This choice was a consequence of the fact that a similar granularity of

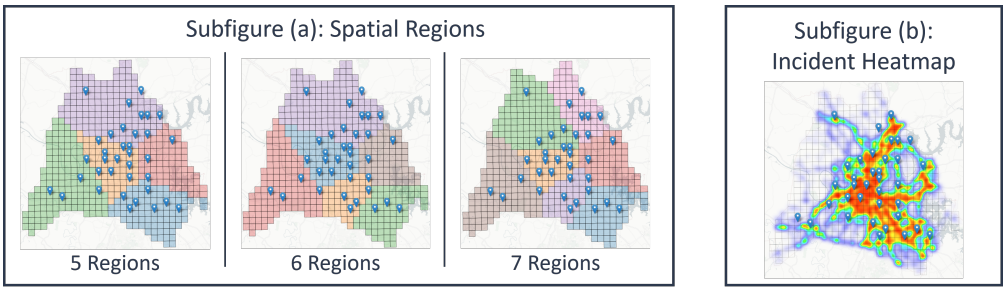


Fig. 5. Subfigure (a) – The various spatial regions under consideration. Pins on the map represent depot locations, and different colors represent different spatial regions. Subfigure (b) – Nashville's historic incident density from January 2018 to May 2019 overlaid on the spatial grid environment.

discretization is followed by local authorities. These cells, as well as the city's 35 depot locations, can be seen in figure 5.

Configuration and hyper-parameters: 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.[3] We assume there are 26 available responders to allocate, which is the actual number of responders in the urban area under consideration [8]. We assume that the mean rate to service an incident is 20 minutes based on actual service times in practice in Nashville (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, which has 38 logical processors running at a base clock of 3.00 GHz, and 64 GB RAM. Our experimental hyper-parameter choices are shown in table 2. 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 5.

Incident Model: Our incident model is learned from the 47862 real incidents discussed earlier. 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.

[3]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.

Table 2. Experimental hyper-parameter choices

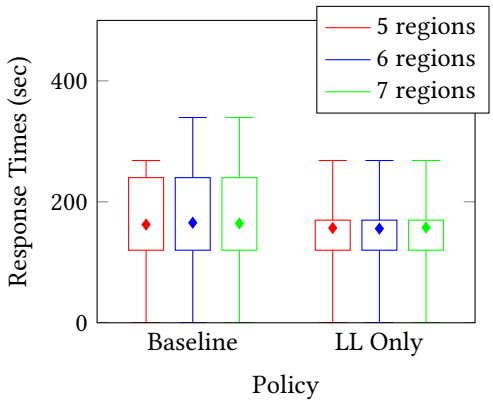
Parameter	Value(s)
Number of Regions	{5, 6, 7}
Maximum Time Between Re-Allocations	60 Minutes
Incident Service Time	20 Minutes
Responder Speed	30 Mph
MCTS Iteration Limit	1000
Discount Factor	0.99995
UCT Tradeoff Parameter c	1.44
Number of Generated Incident Samples	50

Region Segmentation: We use the k-means algorithm [26] implemented in scikit-learn [34] on historical incident data provided by the Tennessee Department of Transportation, which consists of 47862 incidents that occurred from January 2018 to May 2019 in Nashville. We vary the parameter k to divide the total area in consideration into 5, 6, and 7 regions. The cluster centers are initialized uniformly at random from the observed data and we use the classical expectation-minimization based iterative approach to compute the final clusters [34].

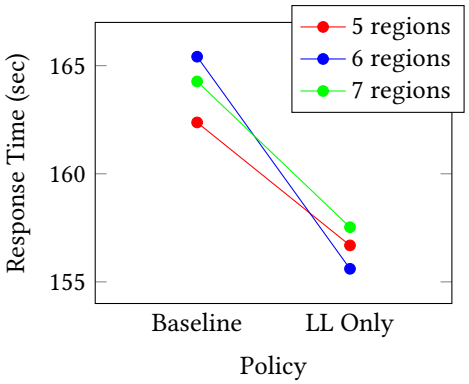
Surrogate Model: To learn the surrogate model over waiting times conditional on the number of responders in a region and mean incident arrival rate, we use the random forest regression model [2]. We use the mean squared error (MSE) to measure the quality of a node split, use 150 estimators, and consider $\sqrt{|n|}$ random features for each split, where n is the number of features. The following hyper-parameters were tuned using a grid search: the maximum depth of each tree, the minimum number of observations in a node required to split it, and the minimum number of samples required to be at a leaf node to split its parent.

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 Nashville. 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). In these experiments we compare using the low-level planner with fixed responder distributions across regions to a full deployment that incorporates the high-level planner to dynamically balance responders across regions.



(a) Response time distributions.



(b) Average response times.

Fig. 6. Results when applying the baseline and low-level planners to incidents sampled from a stationary rate distribution. Sub-figure (a) presents the full response time distributions; the boxplot represents the data's Inter-Quartile Range (IQR = $Q_3 - Q_1$), and the whiskers extend to the 9^{th} and 91^{st} percentiles. Sub-figure (b) presents a zoomed in view of the average response times.

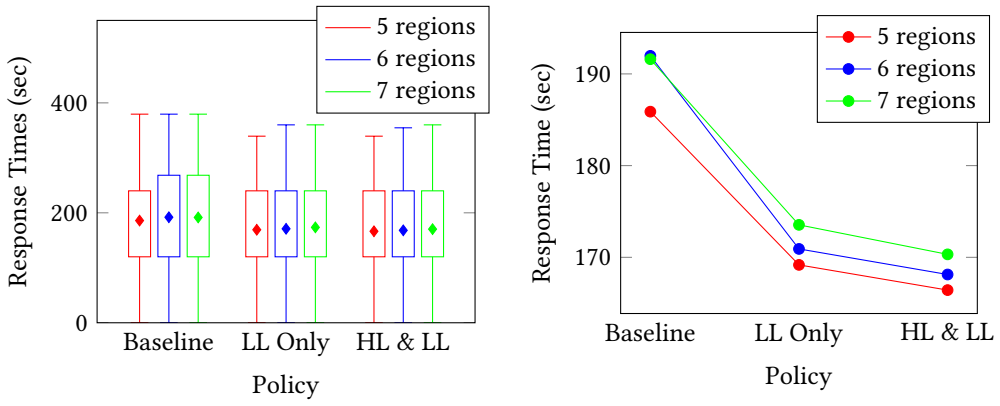
Responder failures: An important consideration in emergency response is to quickly account for situations where some responders might be unavailable due to maintenance and breakdowns. We randomly simulate failures of responders lasting 8 hours to understand how our approach deals with such scenarios.

Baseline Policy: 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.

6 RESULTS

Stationary Incident Rates: We begin by comparing the baseline policy with the proposed low-level planner on incidents sampled from stationary incident rates. Instead of using the data-driven surrogate model and travel-time model, we test the low-level planner in isolation; we use the simpler queue-based model for initial allocations and a travel-time model based on Euclidean distance (we present results with the data-driven models later, but start with the faster to compute Euclidean model). The results are shown in figure 6. 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 [28]. 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 figure 7. Again, we begin by using the simple queue based allocation. Our first observation is that response times generally increase relative to the stationary experiments for both the baseline and the proposed approach.



(a) Response time distributions.

(b) Average response times.

Fig. 7. Results when applying the baseline, low-level planner (LL Only), and complete hierarchical planner (HL & LL) when applied to incidents sampled from a non-stationary rate distribution. Sub-figure (a) presents the full response time distributions; the boxplot represents the data's Inter-Quartile Range (IQR = $Q_3 - Q_1$), and the whiskers extend to the 9^{th} and 91^{st} percentiles. Sub-figure (b) presents a zoomed in view of the average response times.

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 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.

We now evaluate the surrogate model and its effect on the planner. First, we show how the model performs while forecasting waiting times in unseen test data with respect to the queuing model. We show the results in figure 8. We observe that the surrogate model based on random forest regression significantly outperforms the queuing based model; this improvement is expected as the regression model takes into account travel times as well as the time taken to drop victims to hospitals through the simulated data. Even though the queuing based model has large prediction errors, we show that using it as a heuristic to guide the high-level planner outperforms the baseline approach, most likely because such an estimator learns the proportion of waiting times among the regions fairly well.

Finally, we use test the entire hierarchical planning pipeline (with both the surrogate model as well as the queuing based model for initial allocation) and compare it with the baseline approach. We also use the data-driven travel time router to replicate realistic travel times. We present the results in figure 9. We see that usually (in all cases except one), the hierarchical planner with the data-driven surrogate model outperforms the other approaches. On average, it improves response times by about **23 seconds** with respect to the baseline model and by about **6 seconds** with respect

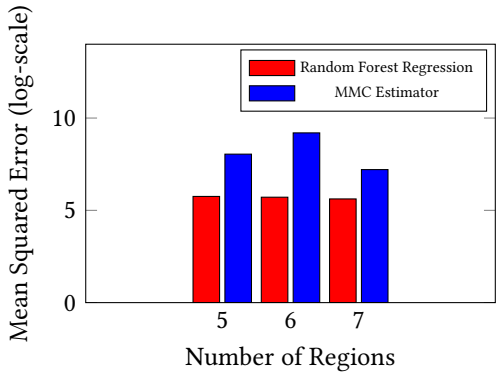
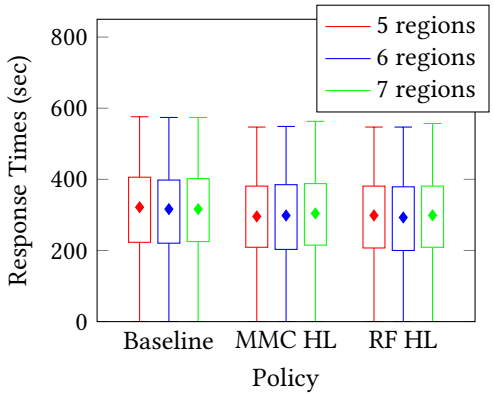
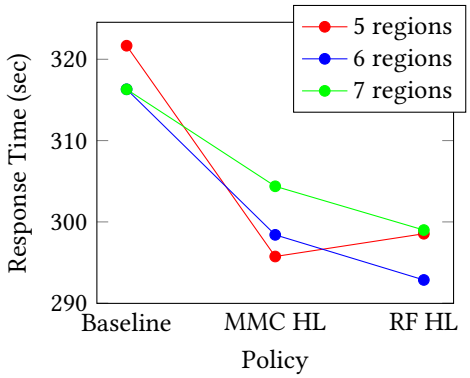


Fig. 8.
Mean Squared Error in logarithmic scale for the proposed estimators. The random forest regression model performs significantly better in comparison to the queuing based estimator. However, as the queuing based estimator learns the proportion of wait times among the regions fairly well, it serves as a meaningful heuristic to guide the high-level planner.



(a) Response time distributions.

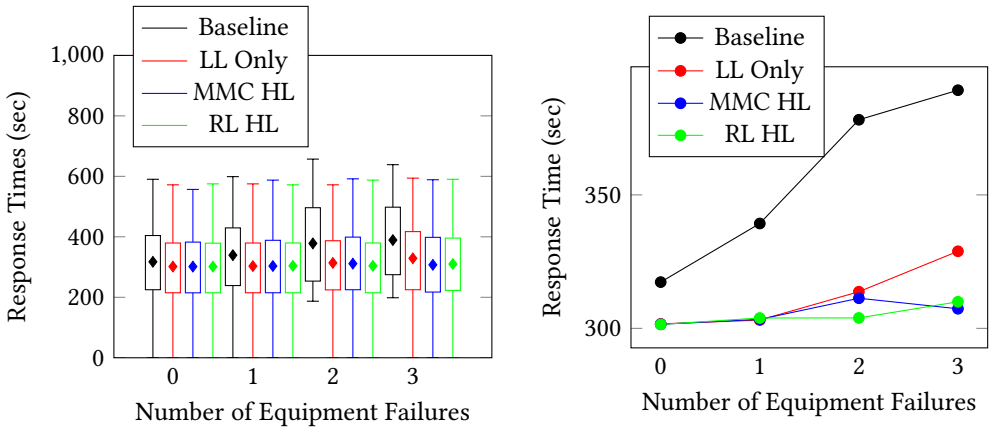


(b) Average response times.

Fig. 9.
Results when applying the baseline, complete hierarchical planner using the MMC queuing high level planner (MMC HL), and complete planner using the surrogate model high level planner (RF HL) when applied to incidents sampled from a non-stationary rate distribution and using a data-driven travel time router. Sub-figure (a) presents the full response time distributions; the boxplot represents the data's Inter-Quartile Range (IQR = $Q_3 - Q_1$), and the whiskers extend to the 9^{th} and 91^{st} percentiles. Sub-figure (b) presents a zoomed in view of the average response times.

to a hierarchical planner that uses a queuing model for the high-level planner when using each model's best region segmentation.

Responder Failures: 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 11 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



(a) Response time distributions.

(b) Average response times.

Fig. 10.
Results when applying the low-level planner only (LL Only), complete hierarchical planner using the MMC queuing high level planner (MMC HL), and complete planner using the surrogate model high level planner (RF HL) when subjected to increasing numbers of simultaneous equipment failures. Sub-figure (a) presents the full response time distributions; the boxplot represents the data's Inter-Quartile Range (IQR = $Q_3 - Q_1$), and the whiskers extend to the 9^{th} and 91^{st} percentiles. Sub-figure (b) presents a zoomed in view of the average response times.

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 the baseline policy, the low-level planner in isolation, the full hierarchical approach using the MMC queue high-level planner, and the full approach using the surrogate model. We show the results in figure 10. Naturally, as the number of failures increases, response times increase as there are fewer responders. However, we observe that the proposed hierarchical approach intelligently allocates the remaining responders to outperform the baseline and low-level planner in isolation. Indeed, when there are three simultaneous failures, using the MMC queuing based hierarchical planner improves response times by about **82 seconds** compared to the baseline policy and about **22 seconds** compared to using only the low-level planner.

Allocation Computation Times: 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.

7 RELATED WORK

Markov decision processes can be directly solved using dynamic programming when the transition dynamics of the system are known [23]. Typically, for resource allocation problems in complex environments like urban areas, the transition dynamics are unknown [31]. To alleviate this, our Simulate-and-Transform (*SimTrans*) algorithm [33] can be used which 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 all

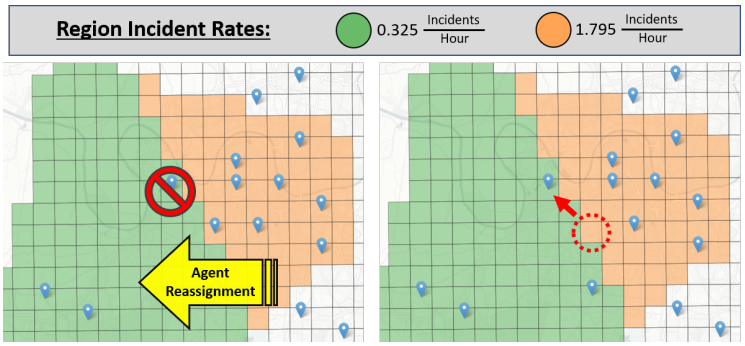


Fig. 11. Example of the high-level planner resolving an equipment failure. In sub-figure (left), the agent positioned at the depot marked by the red circle in the green region fails, and the high-level planner determines there is an imbalance across regions. In sub-figure (right), we see the planner 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.

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

While it 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 [6, 30]. For example, Claes et al. [6] 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.

An orthogonal approach to solve large-scale MDPs is using hierarchical planning [17]. 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 [37, 41]. Later, the macro-policies were used in hierarchical models to address the issue of large state and action spaces [13, 17].

We also 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) [31]. 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 [5, 15, 42]. Waiting time constraints are often used as constraints in approaches that maximize coverage [32, 40]. Decision-theoretic models have also been widely used to design ERM systems. For example, Keneally et al. [21] model the resource allocation and dispatch problem in ERM as a continuous-time MDP, while we have previously used a semi-Markovian process [33]. Allocation in ERM can also be addressed by optimizing distance between facilities and demand locations [10, 30], and explicitly optimizing for patient survival [11, 22].

8 CONCLUSION

We present a hierarchical planning approach for dynamic resource allocation in city scale cyber-physical system (CPS). We formulate a general decision-theoretic problem for that can be used in a variety of resource allocation settings. 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. We use emergency response as a case-study and validate the proposed approach with data from a major metropolitan area in the USA. 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. While this work demonstrates the potential of hierarchical decision making, several non-trivial technical challenges remain, including how to optimally divide a spatial area such that the solutions of the sub-problems maximize the overall utility of the original problem. We will explore these challenges in future work.

REFERENCES

[1] National Emergency Number Association. 2021. 911 Statistics. www.nena.org/page/911Statistics.

[2] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[3] Spencer Chainey, Svein Reid, and Neil Stuart. 2002. *When is a hotspot a hotspot? A procedure for creating statistically robust hotspot maps of crime.* Taylor & Francis, London, England.

[4] 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.

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

[6] 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.

[7] Mark S Daskin. 1995. *Network and discrete location: models, algorithms, and applications.* John Wiley & Sons.

[8] Nashville Fire Department. 2018. Private Communication.

[9] Michael Dzator and Janet Dzator. 2013. An effective heuristic for the p-median problem with application to ambulance location. *Opsearch* 50, 1 (2013), 60–74.

[10] Michael Dzator and Janet Dzator. 2013. An effective heuristic for the P-median problem with application to ambulance location. *OPSEARCH* 50, 1 (March 2013), 60–74.

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

[12] AON Impact Forecasting. 2018. *Weather, Climate and Catastrophe Insight.* Technical Report. AON Impact Forecasting.

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

[14] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. 2008. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Workshop on Experimental and Efficient Algorithms*. 319–333.

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

[16] 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.

[17] 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).

[18] Stephan Huber and Christoph Rust. 2016. Calculate travel time and distance with OpenStreetMap data using the Open Source Routing Machine (OSRM). *The Stata Journal* 16, 2 (2016), 416–423.

[19] Oded Kariv and S Louis Hakimi. 1979. An algorithmic approach to network location problems. I: The p-centers. *SIAM J. Appl. Math.* 37, 3 (1979), 513–538.

[20] 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.

[21] 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.

[22] V. A. Knight, P. R. Harper, and L. Smith. 2012. Ambulance allocation for maximal survival with heterogeneous outcome measures. 40, 6 (2012), 918–926.
[23] Mykel J Kochenderfer. 2015. *Decision Making Under Uncertainty: Theory and Application.* MIT Press.
[24] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European Conference on Machine Learning (ECML)*. Springer, 282–293.
[25] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.
[26] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1. 281–297.
[27] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. 2009. The planar k-means problem is NP-hard. In *International Workshop on Algorithms and Computation*. Springer, 274–285.
[28] Jonathan D Mayer. 1979. Emergency medical service: delays, response time and survival. *Medical Care* (1979), 818–827.
[29] Ayan Mukhopadhyay, Geoffrey Pettet, Mykel Kochenderfer, and Abhishek Dubey. 2020. Designing Emergency Response Pipelines: Lessons and Challenges. *arXiv preprint arXiv:2010.07504* (2020).
[30] 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)*. 185–196.
[31] 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]
[32] 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.
[33] 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.
[34] 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.
[35] 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)*. 1046–1054.
[36] Geoffrey Pettet, Ayan Mukhopadhyay, Mykel J Kochenderfer, and Abhishek Dubey. 2021. Hierarchical planning for resource allocation in emergency response systems. In *ACM/IEEE International Conference on Cyber-Physical Systems*. 155–166.
[37] Doina Precup and Richard S Sutton. 1998. Multi-time models for temporally abstract planning. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1050–1056.
[38] Khashayar Rohanimanesh and Sridhar Mahadevan. 2003. Learning to take concurrent actions. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1651–1658.
[39] John F Shortle, James M Thompson, Donald Gross, and Carl M Harris. 2018. *Fundamentals of Queueing Theory.* Wiley.
[40] 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.
[41] Richard S Sutton. 1995. TD models: Modeling the world at a mixture of time scales. In *International Conference on Machine Learning (ICML)*. 531–539.
[42] Constantine Toregas, Ralph Swain, Charles ReVelle, and Lawrence Bergman. 1971. The Location of Emergency Service Facilities. *Operations Research* 19 (1971), 1363–1373.
[43] Sayyed Mohsen Vazirizade, Ayan Mukhopadhyay, Geoffrey Pettet, Said El Said, Hiba Baroud, and Abhishek Dubey. 2021. Learning Incident Prediction Models Over Large Geographical Areas for Emergency Response Systems. *arXiv preprint arXiv:2106.08307* (2021).
[44] Michael Wooldridge. 2009. *An introduction to multiagent systems.* John wiley & sons.