An Online Decision-Theoretic Pipeline for Responder Dispatch

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

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

Chinmaya Samal
Vanderbilt University
Nashville, TN
chinmaya.samal.1@vanderbilt.edu

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

Yevgeniy Vorobeychik
Washington University
St Louis, MO
yvorobeychik@wustl.edu

ABSTRACT

The problem of dispatching emergency responders to service traffic accidents, fire, distress calls and crimes plagues urban areas across the globe. While such problems have been extensively looked at, most approaches are offline. Such methodologies fail to capture the dynamically changing environments under which critical emergency response occurs, and therefore, fail to be implemented in practice. Any holistic approach towards creating a pipeline for effective emergency response must also look at other challenges that it subsumes - predicting when and where incidents happen and understanding the changing environmental dynamics. We describe a system that collectively deals with all these problems in an online manner, meaning that the models get updated with streaming data sources. We highlight why such an approach is crucial to the effectiveness of emergency response, and present an algorithmic framework that can compute promising actions for a given decision-theoretic model for responder dispatch. We argue that carefully crafted heuristic measures can balance the trade-off between computational time and the quality of solutions achieved and highlight why such an approach is more scalable and tractable than traditional approaches. We also present an online mechanism for incident prediction, as well as an approach based on recurrent neural networks for learning and predicting environmental features that affect responder dispatch. We compare our methodology with prior state-of-the-art and existing dispatch strategies in the field, which show that our approach results in a reduction in response time of responders with a drastic reduction in computational time.

This work is sponsored by The National Science Foundation under award numbers CNS1640624 and IIS1814958. We thank our partners from Metro Nashville Fire Department and Metro Nashville Information Technology Services in this work.
* These authors had equal contribution in this work.

ICCPS '19, April 16–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6285-6/19/04…$15.00
https://doi.org/10.1145/3302509.3311055

CCS CONCEPTS

• **Mathematics of computing** → *Probabilistic representations*; *Markov processes*; • **Information systems** → **Decision support systems**; *Data stream mining*; *Spatial-temporal systems*; • **Applied computing** → *Decision analysis*;

KEYWORDS

Streaming Survival Analysis, Decision Support System, EMS Dispatch

ACM Reference Format:
Ayan Mukhopadhyay*, Geoffrey Pettet*, Chinmaya Samal, Abhishek Dubey, and Yevgeniy Vorobeychik. 2019. An Online Decision-Theoretic Pipeline for Responder Dispatch. In *Proceedings of 10th ACM/IEEE International Conference on Cyber-Physical Systems (with CPS-IoT Week 2019) (ICCPS '19).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3302509.3311055

ACRONYMS

AFT Accelerated Failure Model
ALT A* Search with Landmarks
DTMDP Discrete Time Markov Decision Process
EMS Emergency Medical Services
HCPS Human-in-the-Loop Cyber-Physical Systems
LSTM Long Short-Term Memory Neural Network
MDP Markov Decision Process
MLE Maximum Likelihood Estimation
MCTS Monte Carlo Tree Search
SMDP Semi-Markov Decision Process

1 INTRODUCTION

Emerging Trends and Challenges: Smart and connected communities are Human-in-the-Loop Cyber-Physical systems (H-CPS), with interactions between humans, the outside environment, and computational tools that assist in decision-making processes [6]. Analysis and optimization of H-CPS's is challenging primarily due to the inherent complexity and the sheer number of agents involved. Making accurate models is difficult, and simple rule based strategies often fail to capture the dynamics of the problem space.

Consider the classical problem of emergency response. The goal of responders is to minimize the variance in the operational delay between the time incidents are reported and when responders arrive on the scene. However, solving this problem requires not just sending the nearest emergency responder, but sometimes being proactive placing emergency vehicles in regions with higher incident likelihood. Sending the nearest available responder by euclidean distance ignores road networks and their congestion, as

well as where the resources are stationed. Greedily assigning resources to incidents can lead to resources being pulled away from their stations, increasing response times if an incident occurs in the future in the area where responder should be positioned.

Data-driven approaches have been shown to produce more informed solutions to such problems [29] – examples include predicting crime and traffic accidents in urban areas [20, 22], and building architectures for smart city ecosystems [2]. In this paper, we leverage the potential of data-driven approaches and utilize real-world incident data for making informed decisions about effective stationing and dispatch of Emergency Medical Services (EMS) resources in a large urban community (Nashville, TN).

Contributions: We break down the problem of responder dispatch into three atomic sub-components: incident prediction, environment simulation, and the dispatching approach. Our contributions are as follows:

- Incident Prediction: Online Survival Analysis - We define a novel online approach to incident prediction that predicts incidents in time and space. Previous work in this domain has treated this as a batch learning problem [20, 22, 33], in which incident prediction models are learned once, and are subsequently used to aid response decisions. This fails to capture the changing dynamics of urban systems in which emergency responders operate, and we bridge this gap by creating an online incident prediction algorithm.
- Dispatch Algorithm - We formulate the problem of dispatching responders to incidents as a Semi Markov Decision Process (SMDP). Such an approach has recently been shown to work exceptionally well in this domain [21]. However, such systems have enormous computational load that limit their deployment in practice. We highlight this issue through the course of the paper and design an efficient solution that is fast, scalable and can work in a dynamic environment.
- Decision Theoretic Framework - We compose the Incident Prediction, Environment Simulation, and Dispatching components into a framework that makes real time dispatching decisions based on traffic congestion and predicted incident distributions. Each component is modular, so improvements are easy to integrate into the framework.

Outline: We present and evaluate each of the components separately, as well as the entire system that combines them into an online pipeline and show that it results in better performance, and a remarkable decrease in computational run-time. We begin by presenting a high-level system model and problem description in Section 2, and present our solution in Section 3. We show our empirical evaluation in Section 4, go over a summary of prior work in the field in Section 5 and summarize the paper in Section 6. Table 1 describes the symbols used.

2 PROBLEM DESCRIPTION

The problem deals with an urban area, in which incidents like traffic accidents, fires, distress calls and crimes happen in space and time. Such incidents are reported to a central emergency response system, which then dispatches responders like police vehicles and ambulances. This system governs the entire pipeline of incident response, including detecting and reporting incidents, monitoring and placing a fleet of response vehicles, and finally dispatching responders when incidents occur. Such responders are equipped with devices that facilitate communication to and from central control stations. They are then dispatched by a human (guided by some algorithmic approach), a process which typically takes seconds, but can be longer if dispatchers are busy [8].

Table 1: Notation Table

Symbol	Meaning
G	Set of equally sized grids
R	Set of Responders
t	Arrival-time between incidents
w	Features that affect incident arrival
f	A distribution over t, conditional on w
M_s	Responder Dispatch SMDP
M_d	Responder Dispatch Discrete-Time MDP (DTMDP)
h	Horizon of the Monte-Carlo Search Tree
D	Historical Dataset of incidents
$D^{'}$	Stream Dataset of incidents
L	Log-Likelihood of Incidents
Θ	A Generative Model of the Urban Area

For simplicity we discuss our approach with a single responder type and a single type of incident, but such *homogeneity* is not required for this approach.

Formally, we consider that the entire urban area is divided into as set of grids G. Incidents happen in these grids with an inter-arrival temporal distribution f, conditional on a set of features w. Such incidents need to be responded to by a set of responders R. Each responder is allocated to a specific *depot*, which are immobile stations located in a particular grid. Once a responder has finished servicing an incident, it is directed back to its depot and becomes available to be re-dispatched while in route. We also assume that if there are any free responders when an incident is reported, then some responder must be dispatched to attend to the incident. This is a direct consequence of the legal bounds within which emergency responders operate, as well as of the critical nature of the incidents. If an incident happens and there are no free responders available, then the incident enters a non-priority waiting queue and is attended to when responders become free.

Dispatch Systems in use today by major metropolitan areas such as Nashville work as follows: when an incident is reported, the system dispatches the closest available responder using the euclidean distance (i.e. "as the crow flies") between the incident and the responder's current position [8]. This method has several disadvantages: 1) The euclidean distance between two locations is not necessarily representative of the actual time to travel between them since travel time depends on the road network and its current congestion. 2) By using the responder's current location, it ignores where the responder should be stationed. Nashville's incident response data shows that this can lead to responders being pulled away from their depots, causing future incidents around those depots to have longer response times. 3) This method ignores the likely future incident distribution. Although dispatching the closest responder is greedily optimal, it may not be the best choice given the distribution of future incidents.

This motivates us to model responder dispatch formally. We begin by introducing the problem formulation of the online dispatch system in this section. We denote by $\tau \sim f$, a random variable that represents time between incidents in the urban area.

Given such a model of incident arrival, we now look at the model for responder dispatch. We formally model the problem of dynamic incident response as a semi-Markov decision process (SMDP) [13, 21], and refer to this process as M_s. An SMDP is described by the following tuple,

$$\{S, A, p_{ij}(a), t(i, j, a), \rho(i, a), \alpha\} \tag{1}$$

where S is a finite state space, A is the set of actions, $p_{ij}(a)$ is the probability with which the process transitions from state i to state j when action a is taken, $t(i, j, a)$ is a distribution over the time spent during the transition from state i to state j under action a, $\rho(i, a)$ is the reward received when action a is taken in state s_i, and α is the discount factor for future rewards.

States: At any point in time t, the state of the problem consists of the a tuple $\{I^t, R^t, E^t\}$, where I^t is a collection of grid indices that are waiting to be serviced, ordered according their times of occurrence. R^t represents a collection of vectors, where $r_i^t \in R^t$ captures all relevant information about the i^{th} responder such as it's current position and status. The state variable E^t captures relevant environmental factors that affect dynamic dispatch of responders at time t. Such factors are problem specific, so we leave the choice of such features to the designer of the responder system, but describe the specific features used in our system later.

Actions: Actions in our world correspond to directing responders either to incidents or back to their depots, when the process encounters a decision-making state. We denote the set of all actions by A and refer to a generic action by a. We use $A(s^i)$ to denote the set of actions that are available in state $s^i \in S$, and impose a constraint that whenever at least one responder is available and an incident occurs, we immediately dispatch *some responder*. Since the entire process evolves in continuous time, one can consider the existence of a single decision-making state at any instant, which leads to a single action being needed.

Transitions: The SMDP model evolves as a result of incidents that happen in space and time, and actions that are taken. The state transition probabilities are represented by the random variable $p_{ij}(a)$, which for any state s_i, represents the probability over the system transitioning to state s_j when action a is taken. Also, the time taken for the transition is represented by the random variable $t(i, j, a)$. We collectively refer to the state transition probabilities and time transition probabilities as *transition probabilities* throughout the rest of the paper.

Rewards: Since the broader goal of this problem is to minimize response times for emergency responders, we choose to look at *costs* instead of *rewards*. For each action a that is taken in state s_i, the system incurs a cost $\rho(s_i, a)$, and we seek to find actions for each state that minimizes the expected sum of costs.

Policy: A policy for a decision-making problem specifies an action for each state of the system. The goal of solving the SMDP is to find a policy that maximizes the sum of expected rewards that the decision process generates as a result of following the said policy. Our goal is to approximate the optimal policy π^* which, starting from for an arbitrary state s_i, minimizes the sum of expected discounted costs.

3 OUR SOLUTION

Before introducing the technical details of our solution approach, we provide a broad overview of the algorithmic approach we take and the associated technical challenges. We point out that two characteristics are fundamental to the operation of an emergency response system - first, it must be equipped with the ability to perform real-time computing in order to process the continuous stream of data received from responders and calls, and secondly, it must be equipped with principled algorithmic approaches to dispatch responders as and when incidents happen. We clarify that real-time computation essentially refers to a soft real-time problem - once incidents happen, the entire pipeline can afford a *short* latency to update existing models and calculate dispatch decisions. With these characteristics in mind, we start by looking at incident prediction algorithms. The canonical way to predict incidents in space and time is using historical data to learn a predictive model and then simulate incidents [5, 23]. However, since accidents often cascade, it is imperative that the model is updated as and when incidents happen. The primary technical challenge here is that re-training the entire model each time an incident happens (or periodically after some pre-defined number of incidents) is computationally slow and puts a heavy toll on the responder-dispatch framework that can only afford a low latency. This calls for the need to design an online mechanism to predict incidents that can be updated as incidents happen. We introduce such a model and explain the algorithmic details involved in section 3.1.

Having looked at the problem of incident prediction, we now look at dispatching responders given an incident prediction model. The SMDP formulation introduced in section 2 is difficult to solve since the state transition probabilities are unknown and cannot be computed in closed-form. One way to tackle this problem is to access a generative model to learn the state-transition probabilities while learning a policy. This has recently been shown to work well on the responder-dispatch problem [21]. However, we point out that such an approach has two major limitations. First, for any urban system, the state space is practically intractable even without environment variables. Even on fairly powerful computing systems, it would take weeks to train the policy [21]. The inclusion of environment variables would be computationally infeasible. Secondly, and partly as a consequence of the first issue, prior approaches are simply not suited for dynamic environments: if a single responder breaks down, traffic conditions change, or incident models evolve, existing approaches [14, 21] prescribe re-learning from scratch, which takes time that is incompatible with the latency constraints on the system. In order to alleviate this concern, we take the SMDP formulation and design an algorithmic approach that bypasses the need to learn the transition probabilities. This saves vital computation time and lets us design an online algorithm that is updated in real-time as the environment evolves (see Section 3.2).

In order to consider the effect of environmental factors on responder dispatch, it is essential to understand how such factors evolve. This motivates us to create predictive models for the environment. One factor of interest in the context of emergency responder-dispatch is traffic conditions in urban areas, since they directly affect travel times of responders. We take this into account by describing a model that enables us to learn the evolution of traffic in

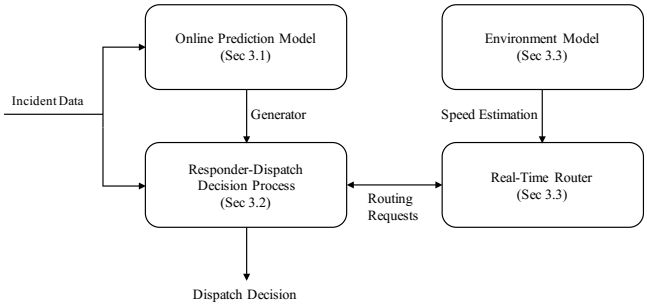

Figure 1: System Overview

an urban area from prior data, and predict traffic conditions on the fly while attempting to solve the MDP (see section 3.3).

The high-level process flow that ties the three problems into one complete pipeline of responder dispatch is shown in figure 1. The online prediction model consumes actual incident data and at any point in time, provides a simulator that captures the latest trends in incident arrival. Also, the model to learn the evolution of environmental variables is used to find optimal vehicular routes between any two points in the urban area. These two atomic pieces are fed into the decision process for responder-dispatch, that given any state of the SMDP, outputs a decision that governs which responder should be sent to respond to an incident.

3.1 Real Time Incident Prediction

We now present the technical details and formalize our methodology, and begin by looking at a principled incident prediction algorithm. Formally, we want to learn a probability distribution over incident arrival in space and time. In order to do so, we leverage our prior work in which we have shown how survival models prove to be extremely effective in predicting incidents like crimes and traffic accidents [20, 21, 23]. Survival Analysis is a class of methods used to analyze data comprising of time between incidents of interest [7]. Survival models can be parametric or non-parametric in nature, with parametric models assuming that *survival time* follows a known distribution. Based on our prior work, we choose a parametric model over incident arrival, and represent the survival model as $f(\tau|\gamma(w))$, where f is the probability distribution for a continuous random variable τ representing the inter-arrival time, which typically depends on covariates w via the function γ. The model parameters can be estimated by the principled procedure of Maximum Likelihood Estimation (MLE). The spatial granularity at which such models are learned can be specified exogenously - a system designer can choose to learn a separate f for each discretized spatial entity (grids in our case), learn one single model for all the grids or learn the spatial granularity from data itself. This choice is orthogonal to the approach described in this paper and we refer interested readers to our prior work [20] for a discussion about such models.

We shift our focus directly to survival models that are used to learn $f(\tau|\gamma(w))$. Intuitively, given an incident and a set of features, we want to predict when the next incident might happen. Before proceeding, we introduce an added piece of notation - we assume the availability of a dataset $D = \{(x_1, w_1), (x_2, w_2), .., (x_n, w_n)\}$, where x_i represents the time of occurrence of the i^{th} incident and w_i represents a vector of arbitrary features associated with the said incident. A realization of the random variable τ, used to measure the inter-arrival time between incidents, can be represented as $\tau_i = x_{i+1} - x_i$. The function γ is usually logarithmic and the relationship of the random variable τ with the covariates can be captured by a log-linear model. Formally, for a time-interval τ_i and associated feature vector w_i, this relationship is represented as

$$log(\tau_i) = \beta_1 w_{i1} + \beta_2 w_{i2} + ... + \beta_m w_{im} + y \tag{2}$$

where, $\beta \in \mathbb{R}^m$ represents the regression coefficients and y is the error term, distributed according to the distribution h. The specific distribution of f is decided by how the error y is modeled. We choose to model τ by an exponential distribution (for the sake of brevity, we refer interested readers to prior work [21] for more details on why an exponential distribution is particularly useful in such models). It turns out that when y follows the extreme value distribution, then τ is distributed exponentially. Thus, in our incident prediction model, we assume that h takes the following form

$$h_Y(y) = e^{y - e^y}$$

Using equation 2, for a given set of incidents, the log-likelihood of the observed data under the specific model can be expressed as

$$L = \sum_{i=1}^{n} \log\ h(\tau_i - w_i^T \beta) \tag{3}$$

The standard way to estimate the parameters of the model is to use a gradient-based iterative approach like the Newton-Raphson algorithm, yielding a set of coefficients β^* that maximize the likelihood expression. Now, the model over inter-arrival times is generative, it can be used to simulate chains of incidents, which is particularly helpful in building a simulator, the purpose of which we explain in the next section.

As pointed out before, such an approach is *offline*. However, it is imperative to capture the latest trends in incident arrival to accurately predict future incidents, which motivates us to design an online approach for learning and predicting incidents. We introduce some added notation before describing the algorithmic approach. First, we reiterate that β^* is used to refer to coefficients already learned from dataset D. Further, we assume that a new set of incidents $D' = \{(x_1', w_1), (x_2', w_2), .., (x_k', w_k)\}$ is available that consists of incidents that have happened after (in time) the original set of incidents. We aim to update the regression coefficients β using D', assuming that the model already has access to β^*.

In order to address this problem, we use stochastic gradient descent to update the distribution f in an online fashion. Formally, we start with the known coefficients β^* and, at any iteration p of the process, we use the following update rule

$$\beta^{p+1} = \beta^p + \alpha \nabla L(\beta^p, D')$$

where $\nabla(L(\beta^*, D')$ is the gradient of the log-likelihood function calculated using D' at β^p and α is the standard step-size parameter for gradient based algorithms. Using equation 3, likelihood of the incidents in the dataset D' can be represented by

$$L = \sum_{i=1}^{k} \log\ e^{(\log \tau_i - \beta^* w) - e^{(\log \tau_i - \beta^* w)}}$$

Algorithm 1: Streaming Survival Analysis

INPUT: Regression Coefficients β^*, Dataset $D^{'}$, Tolerance α, Likelihood Function L, Maximum Iterations MAX_ITER ;
for $p = 1..MAX\_ITER$ **do**
  $\beta^{p+1} = \beta^p + \alpha \nabla L(\beta^p, D^{'})$;
  if $L(\beta^{p+1}, D^{'}) < L(\beta^p, D^{'})$ **then**
    Return β^p;
Return β^p

and subsequently,

$$\frac{\partial L}{\partial \beta_j} = \sum_{i=1}^{k} -w_{ij} + w_{ij}\{e^{(\log \tau_i - \beta^* w_i)}\}$$

The update step is repeated until improvements in the likelihood of the available data. Having already summarized the important steps in the algorithm in this section, we present it formally in Algorithm 1.

This mechanism enables us to update the incident prediction model in an online manner, saving vital computation time for the responder dispatch system. Also, this implicitly betters the dispatch algorithm by generating incident chains that capture the latest trend in incident occurrence.

3.2 Dispatch Algorithm

We begin the discussion on our dispatch algorithm by first explaining how the SMDP problem in formulation 1 can be solved by canonical policy iteration. A principled algorithmic approach [21] to solve the responder-dispatch SMDP is to first convert the SMDP to a discrete-time MDP M_d, which can be represented as

$$\{S, A, \bar{p}_{ij}, \rho, V_\beta, \beta_\alpha\}$$

where $\bar{p}_{ij}(a) = \beta_\alpha^{-1}\beta_\alpha(i,a,j)p_{ij}(a)$ is the scaled probability state transition function and β_α is the updated discount factor. The transformed MDP is equivalent to the original MDP according to the total rewards criterion [13, 21], and hence it suffices to learn a policy for M_d. Given such a conversion, the approach to solving the MDP involves accessing a simulator to learn the state transition probabilities for M_d [21]. The algorithm, *SimTrans*, an acronym for *Simulate and Transform*, uses canonical Policy Iteration on the transformed MDP M_d, with an added computation. It tracks the states and actions encountered by the simulator and gradually builds statistically confident estimates of the transition probabilities.

This process, however, is extremely slow and fails to work in dynamic environments since any change in the problem definition (the number of responders, or the position of a depot) renders the learned policy stale. In order to tackle this problem, we first highlight an important observation - one need not find an optimal action for each state as part of the solution approach since at any point in time, only one decision-making state might arise that requires an optimal action. This difference is crucial, as it lets us bypass the need to learn an optimal policy for the entire MDP. Instead, we describe a principled approach that evaluates different actions at a given state, and selects one that is sufficiently close to the optimal action. We do this using sparse sampling, which creates a sub-MDP around the neighborhood of the given state and then searches that neighborhood for an action. In order to actualize this, we use Monte-Carlo Tree Search (MCTS).

Algorithm 2: Real-Time SMDP Approximation Main Procedure

INPUT: State s, Current Environment E, Horizon h, Stochastic Horizon h^s, Simulation Budget b, Generative Model Θ ;
Set current depth $d \leftarrow 0$;
$C \leftarrow b$ incident chains generated by $\Theta(E)$;
Set Scores $U \leftarrow \varnothing$;
$\bar{A}$ = *SelectCandidateActions* (s, d, h^s) ;
foreach *incident chain* $c \in C$ **do**
  $u \leftarrow$ *ChainEvaluation* $(c, s, d, \bar{A}, h^s, h)$;
  foreach *candidate action* $a \in \bar{A}$ **do**
    $U[a] \leftarrow U[a] + u[a]$;
Return $\text{argmin}_{a \in \bar{A}}(U[a])$;

Algorithm 3: Select Candidate Actions for Given State

Function *SelectCandidateActions* *(State s, Depth d, Stochastic Horizon h^s)*
  $A^s \leftarrow$ set of available actions in state s ;
  $a^* = \text{argmin}_{a \in A^s}(\rho(s,a))$;
  if *depth* $d \geq h^s$ **then**
    Return a^* ;
  else
    $\bar{A}^s = \{a | a \in A^s \text{ and } \rho(s,a) \leq \epsilon * \rho(s,a^*)\}$;
    Return $\bar{A}^s$;

Another important observation is that the incident prediction model discussed in section 3.1 is generative and independent of dispatch decisions, which lets us simulate incidents independently. Note that since models of travel (discussed in section 3.3) as well as service times for responders can also be learned from data, the entire urban area can therefore be simulated. We denote such a simulator by Θ, which can *generate* samples of how the urban area evolves, even though the exact state-transition probabilities are unknown. This observation lets us simulate future states from a given state, leading to the creation of a state-action tree as shown in Fig. 2. We use this to design an algorithmic approach called *Real-Time SMDP Approximation*, and explain it next. Through the course of this discussion, we assume that the simulator can access a modular (and possibly exogenously specified) model to predict the environment at any point in time.

Algorithms 2, 3, 4, and 5 describe the various functions of our MCTS approach. We start our discussion with Algorithm 2, which is the highest level procedure that is invoked when presented with a decision-making state. First, b incident chains are sampled using the generative model Θ (refer to step 3 in Algorithm 2), where each chain is a time ordered list of sampled incidents. We create multiple chains in order to limit the impact of variance in the generative model. Next, the algorithm starts building the MCTS tree. We use the function $node(s, \eta, d, t)$ to refer to the creation of a node in the

Algorithm 4: Evaluate a Chain of Incidents

Function ***ChainEvaluation*** *(Incident Chain c, State s, Depth d, Candidate Actions $\bar{A}$, Stochastic Horizon h^s, Horizon h)*
  Set scores $u \leftarrow \varnothing$;
  $d \leftarrow d + 1$;
  foreach *action* $a \in \bar{A}$ **do**
    Next state $s' \leftarrow$ *UpdateState*(s, a, c) ;
    Utility util = s'.responseTime ;
    Root ← new Node(State = s', util = util) ;
    Update $u[a] \leftarrow$ *CreateStateTree*$(Root, c, d, h^s, h)$;
  Return u

Algorithm 5: Generate State Tree

Function ***CreateStateTree*** *(Parent Node n, Incident Chain c, Depth d, Stochastic Horizon h^s, Horizon h)*
  if *d > horizon h* **then**
    Return n.util ;
  else
    A = *SelectCandidateActions*$(n.state, d, h^s)$;
    $d \leftarrow d + 1$;
    Let ChildUtils $\leftarrow \varnothing$;
    foreach *candidate action* $a_i \in A$ **do**
      Next state $s' \leftarrow$ *UpdateState*$(n.state, a, c)$;
      Let $\text{cost}_i \leftarrow$ *UtilityUpdate*$(s', n.cost, d)$;
      Let $x \leftarrow \text{Node}(s', \text{cost}_i, d, t)$;
      ChildUtils ← ChildUtils ∪ *CreateStateTree*(x, c, d, h^s, h)
    Return min (ChildUtils)
Function ***UtilityUpdate*** *(State s, Parent Utility u_p, Depth d, time t)*
  Return $u_p + (\gamma^t)(t - u_p)/(d + 1))$;

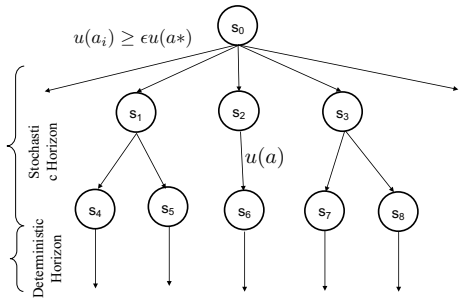

Figure 2: State-Action Tree

tree, which tracks the current state of the system (s), the cost of the path from the root to the node (η), the depth of the tree (d) and the total time elapsed (t). Also, we use *UpdateState(s,a,c)* to retrieve the next state of the system, given the current state s, action a and chain c. For any state, we start by finding a set of candidate actions for the given incident (refer to step 5 in Algorithm 2), which takes the algorithmic flow to Algorithm 3. The candidate actions are chosen according to the current depth of the MCTS tree - if the tree is within the stochastic horizon h^s, the candidate actions include all actions with a cost that is at most ϵ times the cost of the myopically optimal action a^*. The parameter ϵ can be varied to control the trade off between the computational load of the algorithm and performance. Once the tree is deeper than h^s, the algorithm picks the best myopic action as a heuristic to construct the tree's nodes until depth h, since rewards are sufficiently discounted. After candidate actions are found for the sampled incidents of the chain, Algorithm 4 is used to evaluate possible decision-making courses - each available action is tried and the MDP is simulated to generate future decision-making states, from which the entire process is repeated. This gradually builds a tree, where each edge is an action and each node is a decision-making state. We explain this procedure in Algorithm 5.

The key steps of the procedure are as follows. First, costs are tracked for every branch as the tree is built (refer to steps 10 and 14 in Algorithm 5), which is based on the response time in seconds for the assigned responder to the current incident. A lower cost is better, as it corresponds to lower response times. For any given node that was generated by action a from parent node p, the cost is

$$cost = u_p + (\gamma^t)((t - u_p)/(d + 1)) \tag{4}$$

where u_p is the parent node's cost, γ is the discount factor for future decisions, and t is the time elapsed between taking action a at the parent node and the occurrence of the current node. This is essentially an updated weighted average of the response times to incidents given the dispatch actions.

Once the tree is completed, the cost for each candidate action for the dispatch incident is determined by the cost of the best leaf node in it's sub-tree, as this represents the result of the best sequence of future actions that could be taken given the dispatch action. Finally, the algorithm averages the costs for each dispatch action across the b generated incident chains, and selects the candidate action with the minimum overall cost as the best action in the current state (refer to step 10).

If all the responders are busy when an incident occurs, the incident is placed in a waiting queue. As soon as a responder becomes available, it is assigned to the incident at the front of the queue. This continues until the queue is emptied, after which the algorithm returns to using the heuristic policy above.

3.3 Predicting Environmental Factors

We now look at the final component of the proposed pipeline - in order to capture the effect of environment, we must learn how the environment evolves. We specifically focus our attention to traffic conditions, that directly affect the movement of responders. While information about current traffic conditions can be collected while making decisions, it does not suffice for long-term planning. As the dispatch algorithm builds the state-action tree into the future, estimates of environmental variables are needed ahead of time, thereby making it imperative to learn predictive models for such variables. We therefore, design an algorithmic approach to predict future traffic conditions, and highlight how it can be used with an appropriate route-finding algorithm to predict travel times for emergency responders.

Traffic Prediction Model: We model the urban area as a set of road segments. For each segment, we assume that the dataset contains an associated set of features, which include data about the number of lanes, length of a segment, vehicular speed at different

Table 2: Final Hyper-Parameter Choices

Number of Stations (Fraction of Nashville Count)	26 (full)	13 (1/2)	6 (1/4)	3 (1/8)
Simulation Budget b	10	10	10	10
Candidate Action Factor ϵ	1.5	1.5	2.5	1.5
Stochastic Horizon h^s	1	1	2	1
Discount Factor γ	0.9	0.9	0.99999	0.99999

times and so on. Using this data set and features we learn a function over vehicular speed on a segment, conditional on the set of features using a Long Short-Term Memory Neural Network (LSTM) [12] model. The primary capability of such a framework is to model long-term dependencies and determine the optimal time lag for time series problems, which is especially desirable for traffic prediction in the transportation domain.

Route Finding Algorithm: Armed with a model that can predict vehicular speed on road segments, we now look for an approach to find the optimal route between two given points in the urban area. Specifically, given a source, destination and departure time, we seek to find the route with minimum *expected* travel time. To this end, we design a router based on A* search with landmarks (*ALT*) [10]. *ALT* improves upon euclidean-based A* search [11] by introducing landmarks to compute feasible potential functions using the triangle inequality, thereby improving the computational cost involved with such a procedure.

4 PERFORMANCE

4.1 Data and Methodology

Our evaluation uses traffic accident data obtained from the fire and police departments of Nashville, TN, which has a population of approximately 700,000. We trained the generative survival model on 9345 incidents occurring between 1-1-2016 and 2-1-2017, and evaluated the algorithm on 1386 incidents occurring between 2-1-2017 and 4-1-2017. We gathered information about road segments and their geographical locations using real-time traffic data collected from HERE Traffic API [1] for Nashville area. The granularity of this dataset lets us access real-time vehicular speed for all segments in Nashville, which is sampled every minute throughout the day.

Caching the Router Results: While we recommend using a *router* in real-time using the exact locations of responders and incidents to make decisions, it is not feasible for our experiments. Our preliminary analysis showed that each router request takes approximately 0.2 seconds on average. In order to reduce the query time needed to find vehicular speed between arbitrary locations, we cached travel times between locations for different times of the day, with time discretized every 30 minutes. Our experiments showed that travel times in Nashville do not change significantly at this interval (ranging from 2-7 mph). In order to actualize caching, we used the same grid system described in section 2 for locations, with any location in the city discretized to the centroid of it's grid.

4.2 Experimental Setup

We begin by evaluating the streaming survival model separately by comparing it to a batch-learning approach [21]. Then, we evaluate the performance of the optimizers that are used for the router, and finally evaluate the dispatch algorithm. There are two considerations that need to be made during the evaluation -

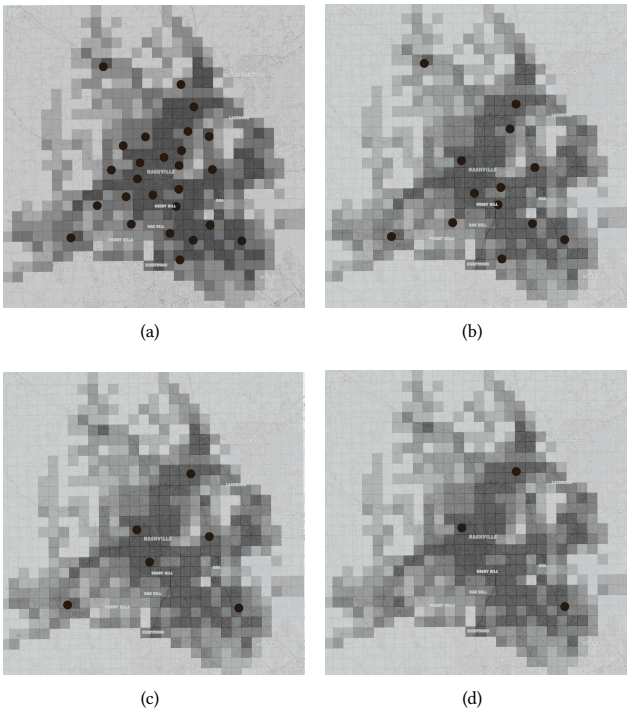

Figure 3: Distribution of the stations in each experiment overlaid on an incident occurrence heatmap (the background shows the map of Nashville, TN): (a) Actual (26) stations; (b) 13 stations; (c) 6 stations; and, (d) 3 stations.

(1) **Decreased Responder Availability**: It is reasonable to assume that the base policy of dispatching the closest responder is correct most of the time and it is only rarely that non-greedy actions are needed. We hypothesize that such situations occur more frequently in practice as the strain on the system is greater: i.e. the incident to responder ratio is higher. This happens since responders attend not only to traffic accidents, but to a variety of other incidents (crimes for example). To take this into account, we ran several experiments with different number of responders: The full Nashville responder count of 26, and then cutting it by a factor of half three times to simulate test-beds with 13, 6, and 3 responders. The locations of the stations in each of these test-beds compared to Nashville's incident density heatmap is shown in figure 3.
(2) **Hyper-Parameters**: We performed a hyper-parameter search (refer to the appendix for a concise summary of the hyper-parameters) for each of the test-beds based on the number of stations. The parameters that gave the best response time savings were chosen for each set, shown in table 2. We note that each hyper-parameter is important and strongly prescribe that each should be tested and tuned carefully for a new environment and hardware the system is deployed on, as these values may not be optimal for more constrained hardware, different responder distributions, or different cities with other incident arrival models.

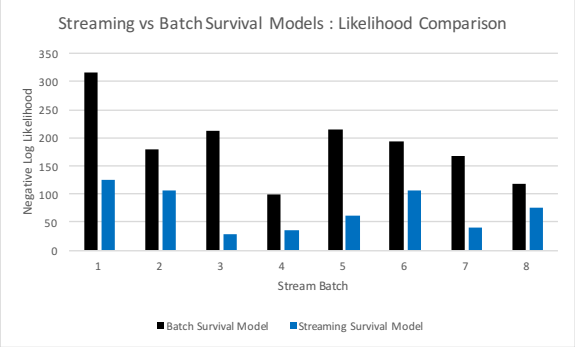

Figure 4: Negative Log Likelihood comparison (lower is better) between Batch and Streaming Survival Models. The Streaming model outperforms the batch model by a significant margin

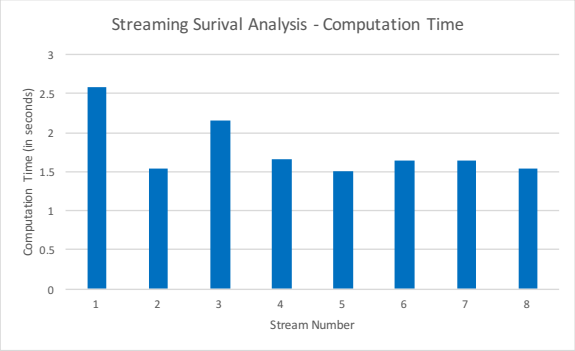

Figure 5: Computational Time for Streaming Survival Analysis.

4.3 Results and Discussion

4.3.1 Streaming Survival Analysis. We learned the batch model using the entire training data set and then, in the evaluation set, we considered each week as a *stream*, and further split 80% of the stream into a training set and 20% as the test set. We evaluated the batch model as well as the streaming model on the test set of each of the streams. Note that the batch model has access to all the data in the streams in the form of features; the stream model, on the other hand, gets updated after each data stream is received according to Algorithm 1. We use the *de-facto* standard of comparing likelihoods for evaluation, and present the results in in Figure 4. The streaming model results in a significant increase in likelihood (we plot the negative log likelihoods, hence lower is better) and convincingly outperforms the batch model. We point out a minor caveat - the updates can be used in practice only if the time taken to update the model is small as compared to the latency that emergency responders can afford. To illustrate this, we present the computational run-times of the stream model (for each stream) in Figure 5, and observe that it usually takes less than 2 seconds for an update to run, which justifies the usage of such models in practice.

In order to visually illustrate the benefit of a streaming model, we look at a fabricated example, where we feed the incident prediction models with data that is deviant from standard accident patterns. We show these results as heatmaps in Figure 6. Note that a brighter color corresponds to higher density of incidents. We see that the batch model *weakly* learns the pattern since it has access to the

Figure 6: Batch Model (a) vs Streaming Model (b): These predicted heatmaps demonstrate that the streaming model adjusts more quickly to new incident distributions. Starting with a survival model learned from the training set, we fed the models synthetic incident data with incidents only occurring in the yellow boxed area, which means that the model should learn that there is now a higher incident likelihood in this area. The batch model picks up the new pattern weakly, whereas the streaming model shows higher likelihood in marked box.

updated dataset only in the form of features; the streaming model on the other hand, identifies the current trend and predicts higher density of incidents in the concerned region, thereby highlighting the importance of such models in dynamic environments.

4.3.2 Predicting Travel Times. We briefly show results of our traffic router before moving to the dispatch algorithm. We compared the LSTM using three different optimizers (Adam [17], SGD [26], and Adagrad [9]), and model performance was evaluated using five-fold shuffled cross validation. Adam, SGD and Adagrad showed Mean Absolute Errors of 5.47, 4.27 and 6.16 miles/hours respectively. Therefore, we chose SGD for our router described in Section 3.3. While evaluating the router on unseen data, the model with SGD optimizer showed MAE of only **6.419 miles/hour**.

4.3.3 Responder Dispatch. In table 3 we present the results of comparing the tuned algorithms for each stations configuration. We compare our solution against the base policy (sending the nearest responder) using the average response time reduction for incidents impacted by the algorithm (i.e. incidents with different response times than the base policy), the number of incidents impacted, and the average computation time. The first observation is that the computation times are all well within acceptable limits, as they are near the human decision maker's visual reaction times [30]. This demonstrates that the system overcomes the technical challenge of running and updating in real-time, and can integrate into emergency response systems described in section 1.

We observe that when there is high responder coverage, demonstrated by the experiment with 26 stations, the baseline policy is nearly always used, with only 5 of the 1386 incidents serviced being impacted by the policy. But as the number of responders decreases, the baseline policy is sub-optimal for an increasing number of incidents, capping at 150 with 3 stations. This shows that the system can respond to changing responder availability, and that it is most useful when the system is strained by high incident demand.

Last, the average time saved for impacted incidents is significant, particularly for the experiments with 26 and 3 stations, as 30

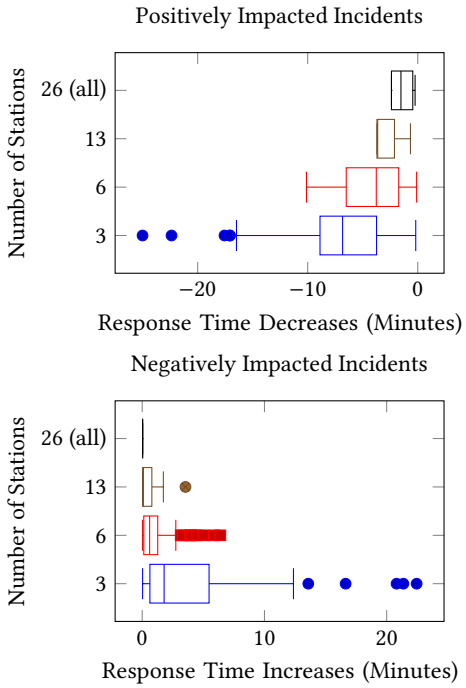

Figure 7: Incident response time difference between the base policy and our solution. The left chart shows the distribution of response time decreases for positively impacted incidents, while the right chart shows increases for negatively impacted incidents. The distribution of time difference in minutes (x axis) is compared across each experiment involving the various station counts (y axis).

seconds can be the difference between mortality and survival in response situations [4]. However, these represent average savings, and to dissect the performance of our approach, we plot the distributions of the response time savings for incidents that benefited from our solution, and response time increases for negatively impacted incidents in figure 7.

Comparing the box plots, the negatively impacted response times are more dense near zero compared to the savings. This shows that in general, the algorithm is not making large sacrifices for individual incidents in comparison to the savings generated, which is reinforced by the overall distribution of response times shown in figure 8. The response times for the positively impacted incidents are generally much improved; the median improvement is over 200 seconds for the experiment with 13 stations, for example. Unfortunately, however, there are some outliers with unacceptably large sacrifices. For example, there is an incident in the experiment with 13 stations that took over 200 additional seconds to respond to compared to the base policy, which significantly increases the potential mortality of that incident if it is severe. This raises the question of integrating severity of incidents into the SMDP model, and we plan to consider the integration of prioritization of incidents in future work.

5 RELATED WORK

A Traffic Incident Management Decision Support System is an information system that supports the process of preparing for, responding to, and managing the effects of traffic incidents. It must

Table 3: Performance of System Compared to Base Policy

Number of Stations (Fraction of Nashville Count)	26 (full)	13 (1/2)	6 (1/4)	3 (1/8)
Average Response Time Savings for Incidents Impacted by Policy (seconds)	38.705	2.231	15.917	34.871
Number of Incidents Impacted by Policy	5	14	99	150
Average Computation Time per Incident (seconds)	0.384	0.198	0.350	0.0343

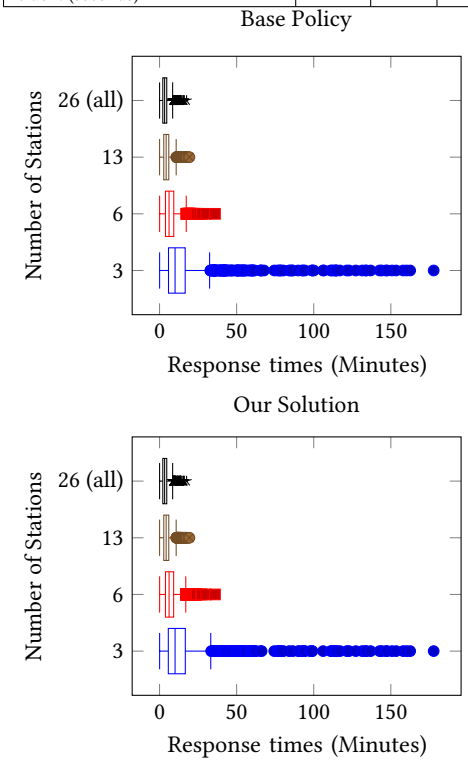

Figure 8: Response time distributions of the base policy compared to our solution for each station experiment. The two are very similar on average since most incidents have the same responder, demonstrating that our solution is not worse than the base policy. The benefits of our solution are clear when looking at the response times for incidents with different dispatching decisions, as shown in figure 7.

support functions such as stationing emergency response resources, dispatching to incidents in real time, and routing resources [34]. Most of these sub-problems have been studied in an orthogonal manner. We look at prior work for each of the sub-problems. First, we look at the problem of dispatching responders given a model of incident arrival. Traditionally, problem has been looked at as a part of the responder allocation problem [18, 20], in which an allocation of responders to depots naturally creates an algorithm for dispatch. The problem has also been studied as part of a joint optimization problem that balances distribution of resources and response times [31]. Finally, principled decision-theoretic models have also been used to study the problem [14, 21], that look at learning a policy of actions for all states the urban area can be in.

The second sub-problem is that of predicting incidents like traffic accidents, crimes, fires and others, that need emergency response.

The availability of such a mechanism is crucial to the first subproblem as decision-theoretic approaches can be aided by mechanisms that can simulate the world in which such responders operate. This specific problem has been widely studied in the past. One of the most widely studied types of incidents is crime, and a variety of approaches [15, 27] have been taken to tackle this problem. The problem of predicting traffic accidents has also received significant attention [3, 28]. Recently, freeway accidents have been predicted using panel data analysis approach that predicts incidents based on both time-varying and site-specific factors [25]. A survey of the literature on crash prediction models is presented in [16], which highlights the prevalence of Poisson distribution based models, and multiple linear regression approaches. There are also approaches that use clustering techniques to differentiate between incident types [24]. Finally, there are generic approaches that can work with multiple incident types [20, 22].

6 CONCLUSION

We designed a complete pipeline for the responder dispatch problem. We created an online incident prediction model that can consume streaming data and efficiently update existing incident arrival models. Then, we designed a framework for finding near-optimal decisions of an SMDP by using Monte-Carlo Tree Search, that bridges an important gap in literature by making such models computationally tractable. To aid the decision-making algorithm, we designed a Recurrent Neural Network architecture to learn and predict traffic conditions in urban areas. Our experiments showed significant improvements over prior work and existing strategies in both incident prediction and responder dispatch. We would like to highlight that while we treated incidents with equal severity, an interesting direction of future work involves designing the SMDP reward structure based on priorities, and directing responders based on incident prediction models that take severity into effect.

REFERENCES

[1] 2018. HERE API. https://developer.here.com/. Accessed: 2018-11-17.

[2] Mohammad Abu-Matar and John Davies. 2017. Data driven reference architecture for smart city ecosystems. In *2017 IEEE SmartWorld*. IEEE, IEEE, San Francisco, CA, USA, 1–7.

[3] Williams Ackaah and Mohammed Salifu. 2011. Crash prediction model for two-lane rural highways in the Ashanti region of Ghana. *IATSS research* 35, 1 (2011), 34–40.

[4] Thomas H Blackwell and Jay S Kaufman. 2002. Response time effectiveness: comparison of response time and survival in an urban emergency medical services system. *Academic Emergency Medicine* 9, 4 (2002), 288–295.

[5] Ciro Caliendo, Maurizio Guida, and Alessandra Parisi. 2007. A crash-prediction model for multilane roads. *Accident Analysis & Prevention* 39, 4 (2007), 657 – 670. https://doi.org/10.1016/j.aap.2006.10.012

[6] Christos G Cassandras. 2016. Smart cities as cyber-physical social systems. *Engineering* 2, 2 (2016), 156–158.

[7] David Roxbee Cox and David Oakes. 1984. *Analysis of survival data*. Vol. 21. CRC Press, New York, NY, USA.

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

[9] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[10] Andrew V Goldberg and Chris Harrelson. 2005. Computing the shortest path: A search meets graph theory. In *16th ACM-SIAM symposium on Discrete algorithms*. SIAM, ACM, Vancouver, BC, Canada, 156–165.

[11] Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Qiying Hu and Wuyi Yue. 2007. *Markov decision processes with their applications*. Vol. 14. Springer Science & Business Media, New York, NY, USA.

[14] 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.

[15] Leslie W Kennedy, Joel M Caplan, and Eric Piza. 2011. Risk clusters, hotspots, and spatial intelligence: risk terrain modeling as an algorithm for police resource allocation strategies. *Journal of Quantitative Criminology* 27, 3 (2011), 339–362.

[16] Vasin Kiattikomol. 2005. *Freeway crash prediction models for long-range urban transportation planning*. Ph.D. Dissertation. The University of Tennessee, Knoxville.

[17] Diederik P Kingma and Jimmy Lei Ba. 2014. Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*.

[18] Xueping Li, Zhaoxia Zhao, Xiaoyan Zhu, and Tami Wyatt. 2011. Covering models and optimization techniques for emergency response facility location and planning: a review. *Mathematical Methods of Operations Research* 74, 3 (2011), 281–310.

[19] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. (2018).

[20] Ayan Mukhopadhyay, Yevgeniy Vorobeychik, Abhishek Dubey, and Gautam Biswas. 2017. Prioritized Allocation of Emergency Responders based on a Continuous-Time Incident Prediction Model. In *AAMAS*. IFAAMS, 168–177.

[21] Ayan Mukhopadhyay, Zilin Wang, and Yevgeniy Vorobeychik. 2018. A Decision Theoretic Framework for Emergency Responder Dispatch. In *AAMAS 2018 Proceedings*. Stockholm, Sweden, 588–596. http://dl.acm.org/citation.cfm?id=3237471

[22] Ayan Mukhopadhyay, Chao Zhang, Yevgeniy Vorobeychik, Milind Tambe, Kenneth Pence, and Paul Speer. 2016. Optimal Allocation of Police Patrol Resources Using a Continuous-Time Crime Model. In *Conference on Decision and Game Theory for Security*.

[23] Ayan Mukhopadhyay, Chao Zhang, Yevgeniy Vorobeychik, Milind Tambe, Kenneth Pence, and Paul Speer. 2016. Optimal allocation of police patrol resources using a continuous-time crime model. In *International Conference on Decision and Game Theory for Security*. Springer, Springer, New York, NY, USA, 139–158.

[24] Geoffrey Pettet, Saideep Nannapaneni, Benjamin Stadnick, Abhishek Dubey, and Gautam Biswas. 2017. Incident analysis and prediction using clustering and Bayesian network. In *2017 IEEE SmartWorld*. IEEE, IEEE, San Francisco, CA, USA, 1–8.

[25] Yi Qi, Brian L Smith, and Jianhua Guo. 2007. Freeway accident likelihood prediction using a panel data analysis approach. *Journal of transportation engineering* 133, 3 (2007), 149–156.

[26] Herbert Robbins and Sutton Monro. 1985. A stochastic approximation method. In *Herbert Robbins Selected Papers*. Springer, 102–109.

[27] Martin B Short, Maria R D'orsogna, Virginia B Pasour, George E Tita, Paul J Brantingham, Andrea L Bertozzi, and Lincoln B Chayes. 2008. A statistical model of criminal behavior. *Mathematical Models and Methods in Applied Sciences* 18, supp01 (2008), 1249–1267.

[28] Praprut Songchitruksa and Kevin Balke. 2006. Assessing weather, environment, and loop data for real-time freeway incident prediction. *Transportation Research Record: Journal of the Transportation Research Board* 1, 1959 (2006), 105–113.

[29] Megan K. Sutherland and Meghan E. Cook. 2017. Data-Driven Smart Cities: A Closer Look at Organizational, Technical and Data Complexities. In *Proceedings of the 18th Annual International Conference on Digital Government Research (dg.o '17)*. ACM, New York, NY, USA, 471–476. https://doi.org/10.1145/3085228.3085239

[30] George T. Taoka. 1989. Brake Reaction Times of Unalerted Drivers. *ITE Journal* 59, 3 (1989), 19–21.

[31] Hector Toro-DíAz, Maria E Mayorga, Sunarin Chanta, and Laura A Mclay. 2013. Joint location and dispatching decisions for emergency medical services. *Computers & Industrial Engineering* 64, 4 (2013), 917–928.

[32] Haiyang Yu, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma. 2017. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors* 17, 7 (2017), 1501.

[33] Chao Zhang, Victor Bucarey, Ayan Mukhopadhyay, Arunesh Sinha, Yundi Qian, Yevgeniy Vorobeychik, and Milind Tambe. 2016. Using abstractions to solve opportunistic crime security games at scale. In *2016 AAMAS Proceedings*. International Foundation for Autonomous Agents and Multiagent Systems, IFAAMAS, Bologna, Italy, 196–204.

[34] Konstantinos G Zografos, Konstantinos N Androutsopoulos, and George M Vasilakis. 2002. A real-time decision support system for roadway network incident response logistics. *Transportation Research Part C: Emerging Technologies* 10, 1 (2002), 1–18.

A APPENDIX

In order to ensure brevity in the main body of the paper, we move some areas of discussion to the appendix. We extend the discussion on all three components of the responder dispatch pipeline here, and start with our algorithm for predicting incidents.

A.1 Data Sources

Table 4: Data Sources

Type	Source	Format	Frequency	Range
Traffic	HERE	Traffic Message Channel	One Minute	10/16 - Present
Accident	Nashville Fire Department	JSON	Manually	02/14 - 06/17
Weather	Dark Sky	JSON	Five Minutes	03/16 - Present

We collect static and real-time data from multiple data sources in the city of Nashville, TN. Table 4 shows the different data used in this work.

A.2 Real-Time Incident Prediction

While most of our approach towards learning a probability distribution over inter-arrival time between incidents is described in the main paper, we describe the features used to learn the survival model here. Our primary choice of features is governed by prior work and expert opinions, and we list the features used in our model in Table 5.

Table 5: Features used in the incident prediction model.

Feature	Description
Time of day	Each day was divided into 6 equal time zones with binary features for each.
Weekend	Binary features to consider whether crime took place on a weekend or not.
Season	Binary features for winter, spring, summer and fall seasons.
Mean Temperature	Mean Temperature in a day
Rainfall	Rainfall in a day
Past Incidents	Separate variables considered for each discrete crime grid representing the number of incidents in the last two days, past week and past month. We also looked at same incident measures for neighbors of a grid.

A.3 Dispatch Algorithm

The original problem M_s, as formulated in section 3.2 is a Markov-Decision Process which can be defined as

$$\{S, A, p_{ij}(a), t(i,j,a), \rho(i,a), \alpha\}$$

For any state s_i and policy π, we define expected discounted total reward over an infinite horizon as

$$V^{\pi}(s_i) = \sum_{n=0}^{\infty} \mathbb{E}\{e^{-\alpha T_n} \rho(s^n, \pi(s^n))\} \tag{5}$$

where s^n is the state at n^{th} decision epoch, and T_n its duration. The broad goal of solving a general MDP is to learn a policy that maximizes the sum of expected rewards for any given state. We look at costs instead of rewards, and seek to find actions for a given state that minimizes the sum of expected costs.

The evolution of this system can be described by the following four steps:

(1) Given a decision-making state s^i, an action $a \in A$ is taken.
(2) This action results in the system receiving an instantaneous reward (or incurring a cost) which is defined by the function $\rho(s^i, a)$.
(3) Upon taking this action, the system transitions to state s^j according to the probability distribution $p_{ij}(a)$
(4) The transitions are however, not instantaneous. The system takes time t to make the transition, where $t \sim t_{ij}$.

Table 6: Algorithm Hyper-parameter Description

Candidate Action Factor ϵ:	ϵ controls the number of responders considered for each incident: any responder with response time within ϵ times the greedy action's response time is considered. Therefore ϵ directly controls the branching factor of the MCTS tree, and has the most significant impact on computation time.
Simulation Budget b:	b is the number of incident chains that are generated and evaluated from the incident model Θ. Increasing b decreases the variance inherent when sampling. Each chain of incidents can be processed in parallel, so increasing b does not directly increase computation time, assuming enough computing resources are available.
Stochastic Horizon h^s:	h^s is the number of predicted incidents to explore in the future before defaulting to the greedy action. Increasing h^s also has a pronounced affect on computation time. Each level of the search tree that is expanded increases the number of states to simulate.
Discount Factor γ:	γ is the discount applied to future predicted incident rewards, and is in the range (0,1). High values of γ weight far future incidents more similarly to those about to happen, while low values give much more weight to incidents that are happening soon. Unlike the other hyper-parameters, varying γ has no effect on computation time, a value that minimizes response times for each environment should be chosen.

The first step (and the most general approach) in solving such a problem is to convert this into a Discrete-Time Markovian process. However, even after discretization, it is particularly challenging to solve this problem since - a) the state-space is practically intractable, and b) the state transition probabilities $p_{ij}(a)$ are unknown. In order

Table 8: LSTM Hyperparameter tuning table

Optimizer	MAE
Adam	5.47
SGD	4.27
Adagrad	6.16

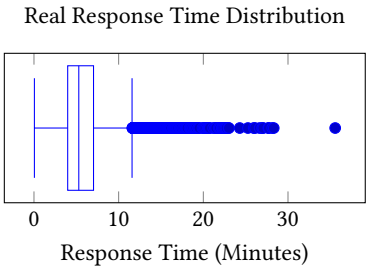

Figure 9: Response time distribution for EMS response for the Nashville Fire Department from 01-02-2016 to 01-02-2017.

Table 7: Summary and dimension of implemented features for traffic prediction model.

Feature	Dim.	Description
Hour of day, Day of week	2	Hour of the day and Day of week used to sample speed data
Length	1	Length of the street segment, collected from OpenStreetMap
Freeflow speed	1	Freeflow speed on the street segment, collected from HERE API. This dataset is private and is collected by our research group
Number of lanes	1	Number of lanes on the street segment, collected from OpenStreetMap
TAZ	741	Binary indication of Traffic Analysis Zone (TAZ) corresponding to this feature vector, collected from US Census Bureau. A TAZ can contain multiple network segments.
Realtime speed	1	The realtime speed value collected from HERE API. This dataset is private and is collected by our research group

to alleviate these issues, *SimTrans* uses canonical policy iteration with an added computational step. At each step of policy iteration, it uses a simulator to estimate values of states it encounters; this provides crucial data about how state transitions occur in the system, which is then used to learn the distribution $p_{ij}(a)$.

While policy iteration is guaranteed to converge to the optimal policy, this approach has major drawbacks. First, due to the size of the state-space, even on fairly powerful computing systems, it takes weeks to learn the optimal policy. Finding such a policy with the inclusion of environment variables would be computationally infeasible. Secondly, and partly as a consequence of the first issue, *SimTrans* is simply not suited for for dynamic environments: if a single responder breaks down, traffic conditions change, or incident models evolve, the policy must be relearned from scratch, which takes time that is incomparable to the latency that such emergency responder systems can afford.

In order to alleviate these concerns, the paper described an algorithm based on Monte-Carlo Tree Search, that looks to learn a near-optimal action for a given state only, instead of focussing on learning a policy over the entire state space. We describe the algorithm in the main paper, but present a summary of the exogenous parameters here in the appendix, for easy reference.

A.4 Traffic and Congestion Prediction

As mentioned in the main paper, we assume that the entire urban area under consideration is divided into a set of road segments V and every $v_i \in V$ has a set of features associated with it.

For building the search tree for the dispatching approach described in the section 3.2, we need to accurately estimate the time it takes for a responder to get to an incident. In order to facilitate this, we developed a predictive model to estimate speed on each road segment for a given time interval. The model needs to be contextualized with features that affect speed, and from our experience and prior work, we chose hour of day, day of week, number of lanes on the road, and the traffic analysis zone as our principal features. These features are described in Table 7.

To find the best optimizer for our LSTM model, we trained it with three different optimizers– Adam [17], SGD [26], Adagrad [9] and model performance was evaluated using five-fold shuffled cross validation. Table 8 shows Mean absolute error (MAE) in miles/hour units for different optimizers. The result shows that SGD performs better with our LSTM model than other optimizers, hence it is chosen for our route finding algorithm. In test dataset, LSTM model with SGD optimizer has MAE of **6.419 miles/hour**. There are other state-of-the-art models for estimating traffic speeds [19, 32], however, we have not explored them in this paper. Our system design is modular and other algorithms can easily fit in.

A.5 Real Incident Response Times

Figure 9 shows the Nashville Fire Department's actual response times to incidents for one year of data from February 2016 to February 2017.