Optimal Dispatch in Emergency Service System via Reinforcement Learning

Cheng Hua and Tauhid Zaman

1 Introduction

In the United States, medical responses by fire departments over the last four decades have increased by 367%, as reported by Evarts (2019). The reasons for the dramatic increase in medical calls include the aging population and health insurance that covers most of the ambulance costs (Boston Globe, Nov 29, 2015). Cities with tight budgets are short of response units to respond to the growing amount of medical calls in time. NBC10 in Philadelphia, on Feb 28, 2019, reported that two-thirds of the emergency medical calls had an ambulance response time of more than nine minutes. Thus, how to efficiently use the existing resources becomes an essential topic to decision-makers in emergency response departments.

In current practice, most cities dispatch ambulance units according to a fixed dispatch rule, which always dispatches the closest available unit. The dispatch policy is deemed a myopic policy as it only considers the current call and ignores the impact of dispatching a unit to future calls. In this paper, we model the ambulance dispatch problem as an average-cost Markov decision process (MDP). We propose an alternative MDP formulation for the problem using post-decision states that we show is mathematically equivalent to the original MDP formulation, but with a much smaller state space. Due to the curse of dimensionality, the applications of the formulations are restricted to small problems. To solve larger problems, we use temporal difference learning (TD-learning) with the post-decision states. We show

C. Hua (✉)
Antai College of Economics and Management, Shanghai Jiaotong University, Shanghai, China
e-mail: cheng.hua@sjtu.edu.cn

T. Zaman
School of Management, Yale University, New Haven, CT, USA
e-mail: tauhid.zaman@yale.edu

H. Yang et al. (eds.), *AI and Analytics for Public Health*, Springer Proceedings in Business and Economics, https://doi.org/10.1007/978-3-030-75166-1_3

that the TD-learning algorithm converges quickly, and the policies obtained from our method outperform the myopic policy.

The remainder of this paper is organized as follows. In Sect. 2, we provide a review of the relevant literature. In Sect. 3, we present the Markov decision process formulation. Section 4 presents the formulation using post-decision states which reduces the state space of the original formulation. In Sect. 5, we present the temporal difference learning algorithm that is based on the post-decision states and its theoretical properties, while in Sect. 6, we show the performance of the algorithm in numerical experiments. We conclude the paper in Sect. 7.

2 Literature Review

The optimal dispatch rule was first studied in Carter et al. (1972). The authors studied a simple system of two units. They provided a closed-form solution that determines the response areas to be served by each unit to minimize average response time. However, such a closed-form solution no longer exists in a system with more than two units and finding the optimal dispatch rule has been an important topic.

Jagtenberg et al. (2017a) studied whether dispatching the closest available unit is optimal in a dynamic ambulance dispatching problem based on a Markov decision process. The problem was discretized by time using one minute as the time interval. Two objectives were considered: mean response time and the fraction of calls with response time beyond a certain time threshold. The value iteration method was used to find the optimal solution. Jagtenberg et al. (2017b) provide an empirical bound for the gap between the existing solutions and the optimal solution.

Schmid (2012) followed the same formulation as introduced in Powell (2010) that uses approximate dynamic programming (ADP) with aggregation function to an ambulance relocation and dispatching problem to reduce the mean response time. A seminal paper by Maxwell et al. (2010) applied ADP to the ambulance redeployment problem, where they used Monte Carlo simulation with one-step bootstrap to estimate complex expectations and applied least squares regression with linear function approximation to learn the approximate value functions. Nasrollahzadeh et al. (2018) studied the problem of real-time ambulance dispatching and relocation, which is formulated as a stochastic dynamic program and solved using ADP. Their formulation and method are the same as proposed in Maxwell et al. (2010). The issues of this approach are that while most of the time they beat the benchmark policy, they usually never output the optimal policy. Also, it is not guaranteed that the learning method always converges, and finding useful basis functions for approximation is more of an art than science, which requires domain knowledge and testing.

Our paper is the first to model the emergency dispatch problem as an average-cost MDP, whose objective is more appropriate than discounted sum. We also show that the proposed TD-learning algorithm based on post-decision states is guaranteed to converge to the optimal solution.

3 Markov Decision Process Formulation

Consider a geographical region $R \subset \mathbb{R}^2$ that is served by a total of N emergency units, all of the same type, e.g., ambulance units. Calls arrive in region R according to a Poisson point process with an arrival intensity $\{\Lambda(x, y) : (x, y) \in R\}$ at location with coordinate (x, y). We partition the region R into J sub-regions and associate a center of mass with each sub-region $R_j \subset R$, which is also referred to as a demand *node*. Note that J can be as large as needed. Denote the total call rate in node j as $\lambda_j = \int_{R_j} \Lambda(x, y) dxdy$. The overall call rate in region R is denoted by $\lambda = \sum_j \lambda_j = \int_R \Lambda(x, y) dxdy$. We assume the mean service time follows a negative exponential distribution with rate μ_i for unit i. We assume that the service time includes turnout time, travel time, and time at the scene. The justification for this assumption is that travel time is usually a small portion of the total service time. With longer travel times, Jarvis (1975) mentioned a *mean service time calibration* to calibrate the service time to maintain the Markov property. Define t_{ij} as the average response time from the base of unit i to node j.

Let b_i represent the state of unit i, where $b_i = 0$ if the unit is available and $b_i = 1$ if the unit is busy. We denote the state space of all units as an N-dimensional vector $B = \{b_N \cdots b_1\} \in \mathscr{B}$, which is in a backward order similar to the representation of binary numbers. We define $B = b_i$ as the status of unit i. Note that $|\mathscr{B}| = 2^N$. If all units are busy when a call is received, we assume that it is handled by a unit outside of the system, such as a private ambulance company, or a unit from a neighboring jurisdiction, which is common mutual aid policy. This paper aims to find the optimal dispatch policy that minimizes the average response time of all served calls.

3.1 State Space

Define S as the state space S and $s \in S$ as the state of the system. We have $s = (j, B)$, which is a tuple of j and B that consists of the location of the current call and the state of all units (available or busy) at the time of the arrival. We denote $s(0) = j$ and $s(1) = B$ in state s. The entire state space has size $|S| = J \times 2^N$.

3.2 Action Space

When a call is received in the system, we decide on which unit to be dispatched to this call. An action in this problem is to dispatch a particular unit upon receiving a call, so the action space is given as $A = \{1, 2, \cdots, N\}$. Note that only an available unit may be dispatched. We define $A_s \subset A$ as the set of feasible actions at state s, where $A_s = \{i : B(i) = 0, i = \{1, 2, \cdots, N\}\}$. We define $a \in A_s$ as an action from the feasible action space.

3.3 Policy Space

We define the policy space as Π, the set of all feasible policies. A policy $\pi \in \Pi$ is a mapping from the state space to the action space, $\pi : S \rightarrow A$. Specifically, $\pi(s) = a, a \in A_s$. The optimal policy $\pi^* \in \Pi$ is the policy that minimizes the average cost over all time steps. Our goal is to find this optimal policy. We use a benchmark policy which sends the closest available unit, denoted by $\pi^m \in \Pi$. We have $\pi^m(s) = \arg\min_i t_{ij}, \forall i \in A_s, \forall s \in S$. Sending the closest available unit is myopic as it does not consider potential future calls. Saving the closest unit to the current call might greatly reduce the expected response time of a future call.

3.4 Costs

Define $c^\pi(s)$ as the cost of being in state s following policy π, which equals to the response time $c^\pi(s) = t_{ij}$ when the call location in state s is j, i.e. $s(0) = j$, and the policy dispatches unit i in state s, i.e. $\pi(s) = i$.

3.5 Transition Probabilities with Augmented Transitions

Define $p^\pi(s, s')$ as the transition probability from state $s = (j, B)$ to state $s' = (j', B')$ under policy π. In determining the transition state probability, we consider an augmented transition where a unit completes a service, and no dispatch action is needed. This is because the number of services completed between two arrivals is a random variable whose probability is complicated to compute. Introducing the augmented transition reduces the number of transition possibilities. Denote I_i as the vector of all 0's except for the ith element, which is 1. The transition rate $p^\pi(s, s')$ with augmented transition is given as

$$p^{\pi}(s, s') = \begin{cases} \frac{\lambda_{j'}}{\lambda + \sum_{k:B(k)=1} \mu_k + \mu_i}, & \text{if } s' = (j', B + I_i), \\ \frac{\mu_l}{\lambda + \sum_{k:B(k)=1} \mu_k + \mu_i}, & \text{if } s' = (\emptyset, B + I_i - I_l). \end{cases} \tag{1}$$

where the expression on the top corresponds to the transition from state s to state s' upon action $\pi(s) = i$ is taken and a new call arrives in node j', and the bottom expression corresponds to the augmented transition where no arrival occurs but a busy unit $l \in A/A_s$ completes its current service. No action is needed since there are no arriving calls in this transition.

3.6 Bellman's Equation

Define $V^{\pi} : S^{+} \mapsto \mathbb{R}$ as the value function for the MDP following policy π and the value of state s is $V^{\pi}(s)$, where S^{+} is the augmented state space that has dimension $|S^{+}| = (J + 1)2^{N}$. Let μ^{π} be the average cost following policy π. The Bellman's equation for the average cost is

$$V^{\pi}(s) = c^{\pi}(s) - \frac{\mu^{\pi}}{2} + \sum_{s' \in S^{+}} p^{\pi}(s, s')V^{\pi}(s'), \ \forall s \in S^{+}. \tag{2}$$

Note that the 1/2 in the above equation is due to the existence of augmented transitions. A transition that is due to a service completion has zero cost and the number of service completions is always equal to the number of calls being served.

Define V^{π} as the vector of all state values, c^{π} as the vector of all state costs, P^{π} as the transition matrix, and e as the vector of all ones. The vector form of Bellman's equation is

$$V^{\pi} = c^{\pi} - \frac{\mu^{\pi} e}{2} + P^{\pi} V^{\pi}. \tag{3}$$

The solution to the above Bellman's equation is not unique. Instead, the set of all value functions takes the form $\{V^{\pi} + me \mid m \in \mathbb{R}\}$. Since shifting the value function by a constant value does not change the relative differences between state values, once we obtain a set of state values V^{π}, the policy can be updated as

$$\pi'(s) = \arg\min_{a \in A_s} t_{aj} + \sum_{s' \in S^{+}} p(s, s'|a)V^{\pi}(s'), \tag{4}$$

where $p^{\pi}(s, s'|a)$ is the one-step transition probability when taking action a instead of following the policy $\pi(s)$. This so-called policy iteration method is summarized in Algorithm 1.

Algorithm 1 Policy iteration method

```
1: Pick a random policy $\pi_0$. Set $k = 0$.
2: while $\pi_k \neq \pi_{k+1}$ do
3:     Compute the cost $c^{\pi_k}$ and transition matrix $P^{\pi_k}$.
4:     Policy Evaluation: Solve the state values $V^{\pi_k}$ from the Bellman's equation (3).
5:     Policy Improvement: For each state $s \in S$, update the actions of each state by
           $$\pi_{k+1}(s) = \arg\min_{a \in A_s} t_{aj} + \sum_{s' \in S^+} p(s, s'|a) V^{\pi_k}(s').$$
6:     $k = k + 1$
7: end while
8: Output: Optimal Policy $\pi^* = \pi_k$
```

4 Post-Decision State Formulation

The policy iteration method guarantees the convergence to the optimal dispatch policy that minimizes the average response time, requiring solving a linear system with $(J + 1)2^N$ states repeatedly. In the section, by realizing the nature of the state transitions of the dispatch problem, we introduce the notion of post-decision states and use them as the new states in our problem. We show that the MDP formulation using post-decision states reduces the state space to 2^N, which also guarantees finding the optimal dispatch policy.

In the original formulation, a state is a tuple of call locations and all units' statuses $s = (j, B)$. A post-decision state s_x is a state that the system is in immediately after observing state s and taking action $a = \pi(s)$, before the next random information arrives into the system, which is the arrival of the next call location. Thus, given state s and unit i being dispatched, i.e., $a = \pi(s) = i$, the post-decision state s_x is $s_x = B + I_i$.

Note that by defining the post-decision state this way, we only need information about the statuses of all units. Define S_x as the post-decision state space; we have $|S_x| = 2^N$. Indeed, $S_x = \mathscr{B}$.

Lemma 1 *Let $p_x^\pi(s_x, s_x')$ be the corresponding transition probability from s_x to s_x'. We have*

$$p_x^\pi(s_x, s_x') = \begin{cases} \frac{\sum_{j \in \mathscr{R}_{l|s_x}^\pi} \lambda_j}{\lambda + \sum_{k:B(k)=1} \mu_k + \mu_i}, & \text{if } s_x' = s_x + I_l, \\ \frac{\mu_l}{\lambda + \sum_{k:B(k)=1} \mu_k + \mu_i}, & \text{if } s_x' = s_x - I_l, \end{cases} \tag{5}$$

where $\mathscr{R}_{l|s_x}^\pi$ is the set of demand nodes where policy π dispatches unit l, i.e.,

$$\mathscr{R}_{l|s_x}^\pi = \{j : \pi(s') = l, s' = (j, s_x)\}. \tag{6}$$

Proof For $s'_x = B + I_i - I_l$, where a transition happens when unit l completes its current service, the post-decision state transition is the same as the transition from s to the augmented state $s' = (\emptyset, B + I_i - I_l)$ as no call arrives and no action is needed for this state; thus $s' = s'_x$.

For $s'_x = B + I_i + I_l$, where a call arrives in post-decision state s_x, we need to capture the randomness of exogenous information, which is the location of call that arrives in s_x. We thus have

$$p^\pi_x(s_x, s'_x) = \sum_j p^\pi(s, s' = (j, s_x))\mathbb{1}_{\{\pi(s')=l\}} = \sum_j p^\pi(s, s')\mathbb{1}_{\{s'=(j,s_x)\}}\mathbb{1}_{\{\pi(s')=l\}}$$

$$= \sum_{j\in\mathscr{R}^\pi_{l|s_x}} p^\pi(s, s' = (j, s_x)) = \frac{\sum_{j\in\mathscr{R}^\pi_{l|s_x}} \lambda_j}{\lambda + \sum_{k:B(k)=1} \mu_k + \mu_i}.$$

□

Lemma 2 *The cost of post-decision state is* $c^\pi(s_x) = \frac{\sum_{l\in A_s}\sum_{j\in\mathscr{R}^\pi_{l|s_x}} \lambda_j t_{lj}}{\lambda+\sum_{k:B(k)=1}\mu_k+\mu_i}$.

Proof The cost of s_x is the expected one-step transition cost from s_x to s'_x under policy π. Let $c^\pi_l(s_x)$ be the expected cost of dispatching unit l in s_x. We have $c^\pi_l(s_x) = \frac{\sum_{j\in\mathscr{R}^\pi_{l|s_x}} \lambda_j t_{lj}}{\sum_{j\in\mathscr{R}^\pi_{l|s_x}} \lambda_j}$ if $l \in A_s$, and 0 otherwise. We thus have

$$c^\pi(s_x) = \sum_{l\in A_s} c^\pi_l(s_x) p^\pi_x(s_x, s_x + I_l) = \sum_{l\in A_s} \frac{\sum_{j\in\mathscr{R}^\pi_{l|s_x}} \lambda_j t_{lj}}{\sum_{j\in\mathscr{R}^\pi_{l|s_x}} \lambda_j} \frac{\sum_{j\in\mathscr{R}^\pi_{l|s_x}} \lambda_j}{\lambda + \sum_{k:B(k)=1} \mu_k + \mu_i}$$

$$= \frac{\sum_{l\in A_s}\sum_{j\in\mathscr{R}^\pi_{l|s_x}} \lambda_j t_{lj}}{\lambda + \sum_{k:B(k)=1} \mu_k + \mu_i}.$$

□

Note that all components defining $c^\pi(s_x)$ and $p^\pi_x(s_x, s'_x)$ are known, which are computed beforehand. Define $J^\pi : S_x \mapsto \mathbb{R}$ as the value function for the post-decision state space S_x. Let μ^π_x be the average cost following policy π in the post-decision state space. The Bellman's equation is

$$J^\pi(s_x) = c^\pi(s_x) - \frac{\mu^\pi_x}{2} + \sum_{s'_x\in S_x} p^\pi_x(s_x, s'_x)J^\pi(s'_x),\ \forall s_x \in S_x. \tag{7}$$

Let J^π, c^π_x and P^π_x be the corresponding vector representations. The vector form Bellman's equation around post-decision states is

$$J^\pi = c^\pi_x - \frac{\mu^\pi_x e}{2} + P^\pi_x J^\pi. \tag{8}$$

Algorithm 2 Policy iteration with post-decision states

```
1: Pick a random policy $\pi_0$. Set $k = 0$.
2: while $\pi_k \neq \pi_{k+1}$ do
3:     Compute the cost $c_x^{\pi_k}$ and transition matrix $P_x^{\pi_k}$.
4:     Policy Evaluation: Solve the state values $J^{\pi_k}$ from the Bellman's equation (8).
5:     Policy Improvement: For each state $s \in S$, update the actions of each state by
           $\pi_{k+1}(s) = \arg\min_{a \in A_s} t_{aj} + J^{\pi_k}(s_x = B + I_a)$.
6:     $k = k + 1$
7: end while
8: Output: Optimal Policy $\pi^* = \pi_k$
```

Theorem 1 *The MDP formulation around post-decision states is equivalent to the original formulation. In particular*

(i) $\mu_x^\pi = \mu^\pi$;
(ii) *For* $s_x = B$, *let* $\Gamma = \lambda + \sum_{k:B(k)=1} \mu_k$, $s^{(j)} = (j, B)$ *and* $s^{[k]} = (\emptyset, B - I_k)$. *We have*

$$J^\pi(s_x) = \sum_j \frac{\lambda_j}{\Gamma} V^\pi\left(s^{(j)}\right) + \sum_{k:B(k)=1} \frac{\mu_k}{\Gamma} V^\pi\left(s^{[k]}\right). \tag{9}$$

The proof of the above theorem is obtained from expanding the value functions V^π in (9) by (2) and collecting terms. The details of the proof is shown in the full version of the paper online. Under this formulation, the new policy π' for state $s = (j, B)$ is updated as $\pi'(s) = \arg\min_{a \in A_s} t_{aj} + J^\pi(s_x = B + I_a)$. The new policy iteration around post-decision states is summarized in Algorithm 2.

5 Temporal Difference Learning with Post-Decision States

Let $\phi_{[p]} : S_x \mapsto \mathbb{R}$, $p = 1, 2 \cdots, P$, be the basis functions of post-decision states, and let $r = \{r_{[p]} : p = 1, 2 \cdots, P\}$ be the tunable parameters. The value function approximation is given by $\tilde{J}(s_x, r) = \sum_{p=1}^{P} r_{[p]} \phi_{[p]}(s_x)$. Let $\tilde{J}(r)$ be the vector of approximate state values of all states given parameter vector r and let Φ be an $2^N \times P$ matrix whose pth column is equal to the vector $\phi_{[p]}$ of all states in S^x. The vector form of the above equation is $\tilde{J}(r) = \Phi r$. Define $\{x_t \mid t = 0, 1, \ldots\}$ as the Markov chain on the post-decision state space S_x with transition matrix P_x^π.

Lemma 3 *The Markov chain corresponding to the state space* S_x *and transition matrix* P_x^π *is irreducible and has a unique stationary distribution.*

The proof of the above lemma is straightforward by noting that the Markov chain with post-decision state space forms a hypercube loss model whose property can be found in Larson (1974).

We define the temporal difference by $d_t = c(x_t) - \frac{\mu_t}{2} + \tilde{J}(x_{t+1}, r_t) - \tilde{J}(x_t, r_t)$, where $c(x_t) - \frac{\mu_t}{2} + \tilde{J}(x_{t+1}, r_t)$ is the differential cost function at state x_t based on the one-step bootstrap and $\tilde{J}(x_t, r_t)$ is the old approximate differential cost function at state x_t. Define r_t as the parameter vector at time t. We update the parameter vector r by $r_{t+1} = r_t + \gamma_t d_t \phi(x_t)$ and $\mu_{t+1} = (1-\gamma_t)\mu_t + 2\gamma_t c(x_t)$. We let $\gamma_t = \frac{a}{a+t}$ where $a \geq 1$ is a hyper-parameter that controls the learning speed.

In this paper, we present a simple way of defining the basis function. We give an index i_x to each post-decision state $s_x = B$, where $i_x = \sum_i^N 2^i \mathbb{1}_{B(i)=1}$. We let the basis function $\phi_{[p]}(s_x)$ be

$$\phi_{[p]}(s_x) = \begin{cases} 1, & \text{if } p = i_x, \\ 0, & \text{if } p \neq i_x. \end{cases} \tag{10}$$

Algorithm 3 Policy iteration with TD-learning

```
1:  Pick a random policy $\pi_0$. Set $k = 0$. Specify $T$ and $K$.
2:  while $k \leq K$ do
3:      Set $t = 0$. Initialize $r_0$ and $\mu_0$.
4:      Starting from a random state $x_0$, generate a state trajectory $\{x_t \mid t = 0, 1, \ldots T\}$ corresponding to the Markov chain with state transition probability $P_x^{\pi_k}$ that is defined by the policy $\pi_k$.
5:      for $t = 0$ to $T$ do
6:          Calculate the temporal difference $d_t$ by
                $d_t = c(x_t) - \frac{\mu_t}{2} + \tilde{J}(x_{t+1}, r_t) - \tilde{J}(x_t, r_t)$.
7:          Update the parameters by
                $r_{t+1} = r_t + \gamma_t d_t \phi(x_t)$,
                $\mu_{t+1} = (1-\gamma_t)\mu_t + 2\gamma_t c(x_t)$.
8:      end for
9:      For each state $s \in S$, update the policy
            $\pi_{k+1}(s) = \arg\min_{a \in A_s} t_{aj} + \tilde{J}(s_x = B + I_a, r_T)$.
10:     $k = k + 1$
11: end while
12: Output: Policy $\tilde{\pi}^* = \pi_K$
```

Theorem 2 *Algorithm 3 has the following three properties:*

(i) *Converges with probability 1.*

(ii) *The limit of the sequence $\frac{\mu_t}{2}$ at the kth iteration of the algorithm is the average cost $\frac{\mu_x^{\pi_k}}{2}$, i.e., $\lim_{t\to\infty} \mu_t = \mu_x^{\pi_k}$.*

(iii) *The limit of the sequence r_t at the kth iteration of the algorithm, denoted by r^{k*}, is the unique solution of the equation $T\left(\Phi r^{k*}\right) = \Phi r^{k*}$, where $T : \mathbb{R}^{2^N} \mapsto \mathbb{R}^{2^N}$ is an operator defined by $TJ = c_x^{\pi_k} - \frac{\mu_x^{\pi_k} e}{2} + P_x^{\pi_k} J$.*

The proof of the above theorem follows from Tsitsiklis and Van Roy (1999) and Lemma 3, and the basis functions $\phi(s_x)$ being linearly independent for all states. It is also necessary that γ_t is positive, deterministic, and satisfies $\sum_{t=0}^{\infty} \gamma_t = \infty$ and $\sum_{t=0}^{\infty} \gamma_t^2 < \infty$. The details of the proof is shown in the full version of the paper. Theorem 2 guarantees that Algorithm 3 always returns the optimal policy if T is large enough. When T is moderately large, it is enough to obtain a policy close to optimal, as shown in the next section. Our algorithm avoids solving the complex Bellman's equation, which has exponential complexity. Once we calculate the cost vector c_x^{π} and transition matrix P_x^{π}, we easily obtain the temporal differences d_t by Monte Carlo simulation and evaluate the value functions that are needed for policy improvement in the policy iteration algorithm.

6 Numerical Results

In this section, we show the numerical results comparing the policy obtained from the TD-Learning method to the myopic policy that always dispatches the closest available unit for systems with $N = 5$, 10, and 15 units. We created an imaginary region which is partitioned into $J = 30$ demand nodes. We randomly locate units in the region and obtain the corresponding response times from each unit to each demand point. The policy from the proposed TD-Learning method with post-decision states is obtained by running the algorithm in 25 iterations. We perform a roll-out with 200,000 state transitions in each iteration and update the parameter vector r using the temporal differences d. We record the sample average response time in each iteration and the results are shown in Figs. 1a, b, and c, respectively.

We observe that the TD-Learning algorithm converges quickly in all cases as expected. We show the updates of the post-decision state values $\tilde{J}$ in one TD-Learning iteration for the case of $N = 5$ in Fig. 1d. The resulted policies in all three cases outperform the myopic policy that always dispatches the closest available units. We also observe that our algorithm obtains a superior policy reasonably quickly in about three iterations. In the case where $N = 15$, solve the Bellman's equation requires solving a system with $31 \times 2^{15} = 1{,}015{,}808$ states. In contrast, our method obtains a good policy in less than two minutes, and it applies virtually to systems of any sizes as guaranteed by its theoretical properties.

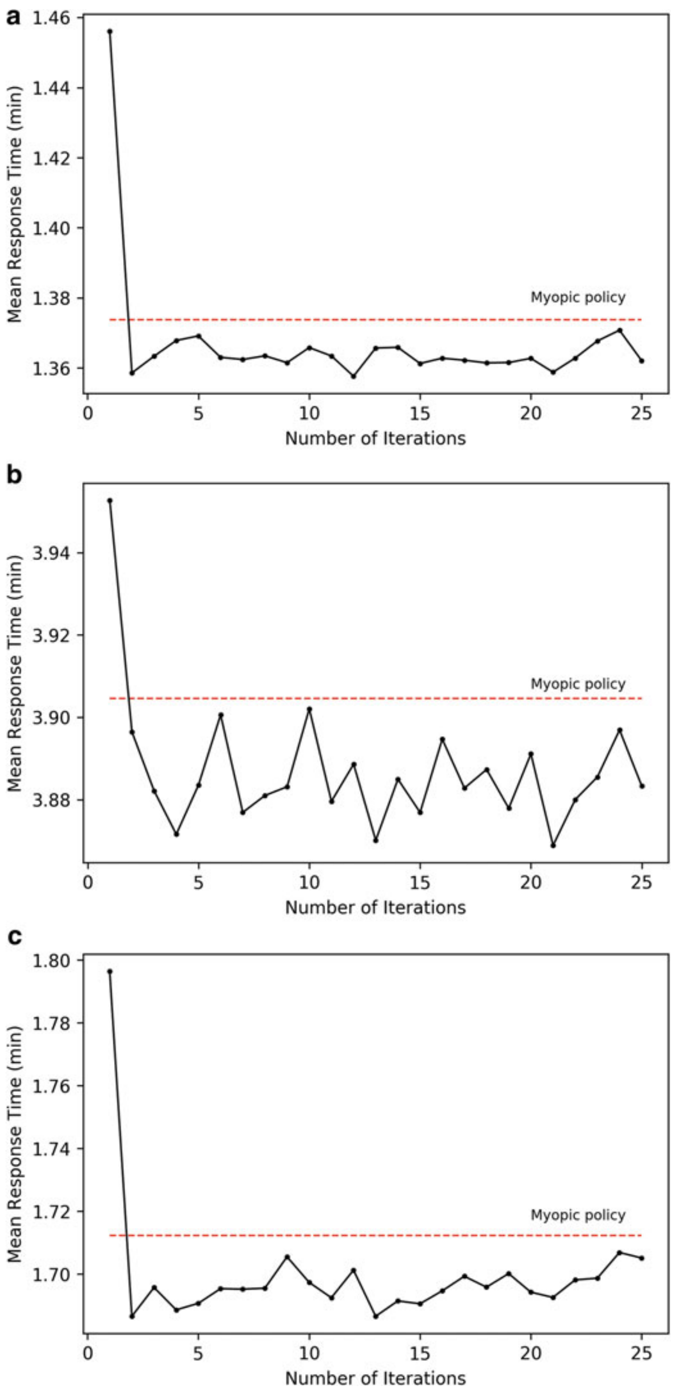

Fig. 1 (continued)

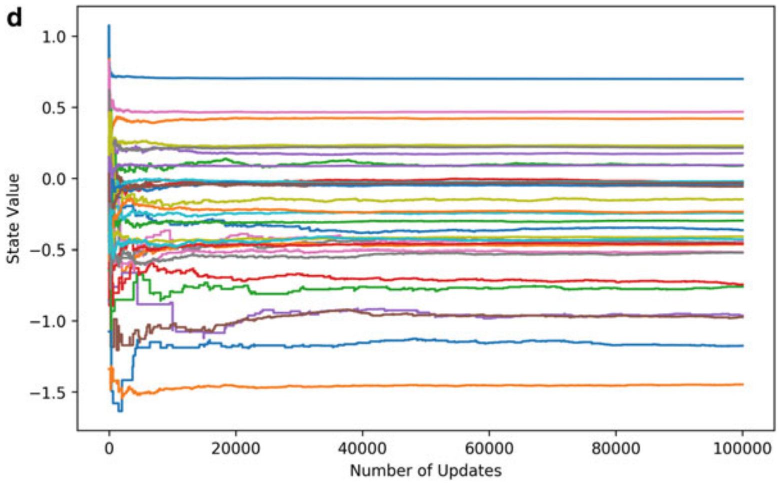

Fig. 1 Numerical results. (**a**) Mean response time comparison for $N = 5$ units. (**b**) Mean response time comparison for $N = 10$ units. (**c**) Mean response time comparison for $N = 15$ units. (**d**) State value updates in one TD-Learning iteration

The policies obtained from our algorithm result in an average of three seconds reduction in terms of response time with no additional resources. Our findings suggest that emergency response departments can improve their performance with minimal to no cost.

7 Conclusion

In this paper, we model the ambulance dispatch problem as an average-cost Markov decision process and aim to find the optimal dispatch policy that minimizes the mean response time. The regular MDP formulation has a state space of $(J + 1) \cdot 2^N$. We propose an alternative MDP formulation that uses the post-decision states and reduces the state space to 2^N. We show that this formulation is mathematically equivalent to the original MDP formulation.

The two formulations are restricted to only small problems due to the curse of dimensionality. We next present a TD-Learning algorithm based on the post-decision states that is guaranteed to converge to the optimal solution. In our numerical experiments, we show that the TD-Learning algorithm with post-decision states converges quickly. The policies obtained from our method outperform the myopic policy that always dispatches the closest available unit in all cases.

References

Carter, G. M., Chaiken, J. M., & Ignall, E. (1972). Response areas for two emergency units. *Operations Research, 20*(3), 571–594.

Evarts, B. (2019). Fire loss in the united states during 2018. *NFPA National Fire Protection Association, Quincy*.

Jagtenberg, C. J., Bhulai, S., & van der Mei, R. D. (2017a). Dynamic ambulance dispatching: is the closest-idle policy always optimal? *Healthcare Management Science, 20*(4), 517–531.

Jagtenberg, C. J., van den Berg, P. L., & van der Mei, R. D. (2017b). Benchmarking online dispatch algorithms for emergency medical services. *European Journal of Operational Research, 258*(2), 715–725.

Jarvis, J. P. (1975). *Optimization in stochastic service systems with distinguishable servers*. Ph.D. thesis, Massachusetts Institute of Technology.

Larson, R. C. (1974). A hypercube queuing model for facility location and redistricting in urban emergency services. *Computers & Operations Research, 1*(1), 67–95.

Maxwell, M.S., Restrepo, M., Henderson, S. G., & Topaloglu, H. (2010). Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing, 22*(2), 266–281.

Nasrollahzadeh, A. A., Khademi, A., & Mayorga, M. E. (2018). Real-time ambulance dispatching and relocation. *Manufacturing & Service Operations Management, 20*(3), 467–480.

Powell, W. B. (2010). Merging AI and OR to solve high-dimensional stochastic optimization problems using approximate dynamic programming. *INFORMS Journal on Computing, 22*(1), 2–17.

Schmid, V. (2012). Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European Journal of Operational Research, 219*(3), 611–621.

Tsitsiklis, J. N., & Van Roy, B. (1999). Average cost temporal-difference learning. *Automatica, 35*(11), 1799–1808.