A Deep Reinforcement Learning-Enabled Dynamic Redeployment System for Mobile Ambulances

SHENGGONG JI, Southwest Jiaotong University, China
YU ZHENG[*†], JD Urban Computing Business Unit, China
ZHAOYUAN WANG, Southwest Jiaotong University, China
TIANRUI LI[†], Southwest Jiaotong University, China

Protecting citizens' lives from emergent accidents (e.g. traffic accidents) and diseases (e.g. heart attack) is of vital importance in urban computing. Every day many people are caught in emergent accidents or diseases and thus need ambulances to transport them to hospitals. In this paper, we propose a dynamic ambulance redeployment system to reduce the time needed for ambulances to pick up patients and to increase the probability of patients being saved in time. For patients in danger, every second counts. Specifically, whenever there is an ambulance becoming available (e.g. finishing transporting a patient to a hospital), our dynamic ambulance redeployment system will redeploy it to a proper ambulance station such that it can better pick up future patients. However, the dynamic ambulance redeployment is challenging, as when we redeploy an available ambulance we need to simultaneously consider each station's multiple dynamic factors. To trade off these multiple factors using handcrafted rules are almost impossible. To deal with this issue, we propose using a deep neural network, called deep score network, to balance each station's dynamic factors into one score, leveraging the excellent representation ability of deep neural networks. And then we propose a deep reinforcement learning framework to learn the deep score network. Finally, based on the learned deep score network, we provide an effective dynamic ambulance redeployment algorithm. Experiment results using data collected in real world show clear advantages of our method over baselines, e.g. comparing with baselines, our method can save ∼100 seconds (∼20%) of average pickup time of patients and improve the ratio of patients being picked up within 10 minutes from 0.786 to 0.838. With our method, people in danger can be better saved.

CCS Concepts: • **Information systems** → **Spatial-temporal systems**; *Geographic information systems.*

Additional Key Words and Phrases: Mobile computing, urban computing, dynamic redeployment system for mobile ambulances, deep reinforcement learning, deep score network

ACM Reference Format:
Shenggong Ji, Yu Zheng, Zhaoyuan Wang, and Tianrui Li. 2019. A Deep Reinforcement Learning-Enabled Dynamic Redeployment System for Mobile Ambulances. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 15 (March 2019), 20 pages. https://doi.org/10.1145/3314402

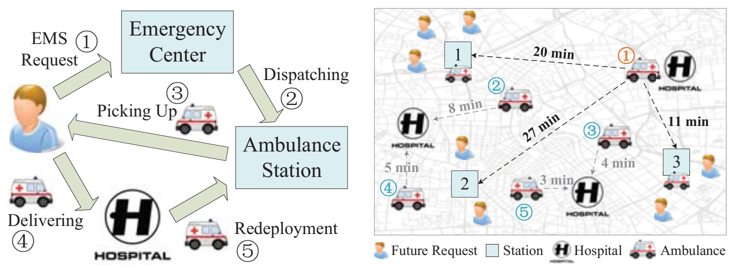
[*]Yu Zheng is also with JD Intelligent City Research, Beijing, China, and School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China.
[†]Yu Zheng and Tianrui Li are the correspondence authors of this paper.

Authors' addresses: Shenggong Ji, School of Information Science and Technology, Southwest Jiaotong University, Chengdu, Sichuan, China, shenggongji@163.com; Yu Zheng, JD Urban Computing Business Unit, Beijing, Beijing, China, msyuzheng@outlook.com; Zhaoyuan Wang, School of Information Science and Technology, Southwest Jiaotong University, Chengdu, Sichuan, China, wang_zhaoyuan@foxmail.com; Tianrui Li, School of Information Science and Technology, Southwest Jiaotong University, Chengdu, Sichuan, China, trli@swjtu.edu.cn.

2474-9567/2019/3-ART15 $15.00
https://doi.org/10.1145/3314402

1 INTRODUCTION

Emergent accidents (e.g. traffic accidents) and emergent diseases (e.g. cerebral hemorrhage, heart attack) take away massive people's lives in cities each year [2, 7, 11]. In city of Tianjin, China, for example, over one hundred thousand citizens are caught in emergent accidents or diseases every year, in need of ambulances to transport them to hospitals timely [4]. Hence Emergency Medical Services (EMS) in a city are of great importance to saving people's lives from emergent accidents and diseases, through dispatching ambulances to efficiently pick up patients and transport them to hospitals (as depicted in Figure 1(a)). For patients, every second counts, and the earlier they are picked up, the more likely their lives can be saved. Consequently, for an EMS system in a city, developing ubiquitous technologies to reduce the pickup time of patients is significantly important.



(a) Ambulance redeployment (b) Multiple dynamic factors

Fig. 1. Ambulance redeployment and the multiple dynamic factors that should be considered.

The pickup times of patients, to a large extent, depends on the dynamic redeployment strategy of mobile ambulances (step 5 in Figure 1(a)). That is, which station should an ambulance be redeployed to, after it becomes available? For example, in Figure 1(b), the ambulance 1 has sent a patient to a hospital and becomes available to be redeployed to one of the three ambulance stations in the city. The pickup times of future patients are affected by the redeployment of current ambulances. For example, three patients will come near station 1 in future, thus redeploying current available ambulance 1 to station 1 can make these patients being picked up quickly, through dispatching ambulances from station 1. Although the dispatching of ambulance (step 2 in Figure 1(a)) also affects the pickup time of patients, in real world, usually it is the nearest ambulance that is dispatched to pick up a patient [16, 26, 35]. This is due to that the greedy dispatching method has already achieved so competitive performance that it is quite hard to develop a better dispatching method [17, 27]. Thus, we study the redeployment method.

However, the dynamic ambulance redeployment is challenging, since there are multiple dynamic factors that should be considered and balanced simultaneously. Specifically, for each station in Figure 1(b), dynamic factors affecting whether the current available ambulance 1 should be redeployed to it are as follows.

(1) The expected number of patients nearby this station in the future. If more patients will come nearby this station in the future, redeploying the current available ambulance to this station will be more helpful.
(2) The number of ambulances this station currently contains. The less ambulances this station contains, the more needed for the emergency center to redeploy the current available ambulance to this station.
(3) The travel time needed for the current available ambulance to reach this station. If the ambulance is far away from this station, it is not proper to be redeployed to this station, spending much time travelling on road, in empty.
(4) The status of other occupied ambulances. For example, in Figure 1(b), there are occupied ambulances 2 and 4, which will be available soon at a hospital closer to station 1 than ambulance 1. Then, it becomes less necessary to redeploy the current available ambulance 1 to station 1, since we can redeploy ambulances 2 and 4 to station 1 in the near future.

Selecting a station under a good trade-off between these four factors is by no means trivial, since different factors have different preferences to stations. For example, in Figure 1(b), factor 1 prefers to redeploy the ambulance 1 to to station 1 as it has the most nearby requests in the future. However, if only factor 2 is considered, the ambulance 1 should be redeployed to station 2 since it doesn't contain any ambulance. Likewise, for factor 3, station 3 is the most proper station as it is the closest to the ambulance 1. Besides these three factors, the status of the occupied ambulances 2-5 (factor 4) should also be considered. Thus, quantitatively balancing all factors is not easy. In real-world EMS systems, the number of stations and the number of occupied ambulances can be dozens, making the selection of a proper station more difficult.

Facing these complex factors, previous ambulance redeployment methods usually manually design indicators to combine these factors as one. However, these indicators consider only one or some of these factors. For instance, coverage of a station is the most widely used indicator [6, 9, 16], which considers only factor 1 and factor 2. Then according to each station's current coverage, an available ambulance is redeployed to the station with the least coverage [16]. Or, some greedy methods may directly redeploy an available ambulance to the nearest station (factor 3), or the station with the least ambulance (factor 2). Besides, some static redeployment methods [9, 32, 35] directly redeploy an ambulance to its base station, without considering any dynamic factor of each station. Clearly, these manually-designed indicators are hard to optimize the pickup time of patients, since these complex factors are almost impossible to be balanced by handcrafted rules.

To deal with this problem, in this paper, we propose a simple, but novel and effective, dynamic ambulance redeployment method, which is able to trade off all complex factors aforementioned. Specifically, our contributions are as follows.

- We propose to utilize a deep neural network, called deep score network, to automatically integrate (trade off) all dynamic factors of a station into one score, leveraging the excellent representation ability of deep neural networks. Based on each station's integrated score, we can dynamically redeploy an available ambulance to the station with the highest score (Sections 2.1 and 2.3).
- We apply a deep reinforcement learning framework, based on a policy gradient algorithm, to learn the deep score network. Note that the score network cannot be learned in a supervised manner, due to that we don't have the labelled score data given a station's factors. With the deep reinforcement learning framework, we learn the score network meanwhile minimizing the pickup time of patients (Section 2.2).
- We evaluate our dynamic ambulance redeployment method using datasets collected in real world. Experimental results demonstrate the clear advantages of our ambulance redeployment method over state-of-the-art baseline methods. Specifically, comparing with baseline methods, our method can save ~20% (~100 seconds) of average pickup time and can improve the ratio of patients being picked up within 10 minutes from 0.786 to 0.838. Every saved second is important to increasing the probability of patients being saved in time (Section 3).
- Besides, we also conduct extensive experiments to test the robustness of our method to some real-world scenarios, e.g. dynamic traffic conditions, different settings in EMS systems, and human factors. Based on experiment results, our mobile ambulance redeployment method is highly robust to real-world scenarios, indicating probably broad applications in real-world EMS systems (Section 3.12).

2 DEEP SCORE NETWORK AND DYNAMIC REDEPLOYMENT ALGORITHM

In this section, we first study the deep score network to integrate a station's dynamic factors into one score (Section 2.1). And then we propose a reinforcement learning framework to learn the score network (Section 2.2). Finally, we detail how to do dynamic ambulance redeployment based on the deep score network (Section 2.3).

2.1 Deep Score Network

Our deep score network is presented in Figure 2. As shown in the figure, our score network contains inputs, two hidden layers, and finally an output. The inputs are each station i's current factors, as mentioned in the Introduction, denoted by x_i. Each station i's dynamic factors x_i decides the redeployment of the current available ambulance. The number of neurons in each layer is tunable, and in this work, we set them both as 20. After each hidden layer, a *tanh* activation function is used, which adds a nonlinear relation between the output score and the input factors. *tanh* activation function is widely used in deep learning [13]. The output is the (scalar) score of the station i, denoted by y_i. For simplicity, we denote all weights in the score network using θ. Then, the score network can be mathematically expressed as

$$y_i = f(x_i; \theta). \tag{1}$$

The weights θ in the score network are learned in Section 2.2 using a deep reinforcement learning framework.

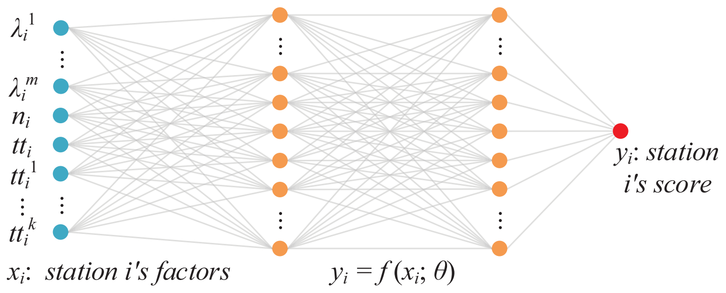


Fig. 2. Our deep score network. Input: a station's current factors x_i; Output: the score of the station y_i.

As shown in Figure 2, for each station i, its factors x_i is a $(m + 1 + 1 + k)$-dimension vector:

$$x_i = (\lambda_i^1, \cdots, \lambda_i^m, n_i, tt_i, tt_i^1, \cdots, tt_i^k), \tag{2}$$

where $(\lambda_i^1, \cdots, \lambda_i^m)$, n_i, tt_i, and $(tt_i^1, \cdots, tt_i^k)$ correspond to factors 1, 2, 3, and 4, respectively, as discussed in the Introduction. Below, we introduce the detail of each factor in x_i.

Factor 1. $\lambda_i^1, \cdots, \lambda_i^m$ is the expected number of future EMS requests nearby station i, e.g. EMS requests in the next hour and in the hour after next. m denotes the number of future time periods under consideration. A time period can be an hour or half an hour. We define that an EMS request is nearby an ambulance station i, if the station i is the nearest station to the request, in terms of the travel time of ambulances on road networks.

Clearly, $\lambda_i^1, \cdots, \lambda_i^m$ can be predicted based on the number of EMS requests nearby station i at corresponding time periods in history. Actually, many time series prediction algorithms have already been proposed in previous literature, e.g. [5, 44]. Since the main focus of this paper is on the design of the dynamic ambulance redeployment methods, without loss of generality, we use the historical average at corresponding time periods as the prediction values for $\lambda_i^1, \cdots, \lambda_i^m$ for each station i. A more accurate prediction can make our method perform better, i.e. the historical average gives a lower bound of our method's performance.

Factor 2. n_i is the number of ambulances that the station i currently contains. From the perspective of the emergency center in an EMS system, it is easy to obtain each station i's dynamic n_i.

Factor 3. tt_i refers to the expected travel time needed for the current available ambulance to reach station i, if it is redeployed to station i. tt_i is an estimated (i.e. predicted) value. Travel time will be affected by traffic conditions. Fortunately, many travel time estimation methods have been proposed, e.g. [36–38, 40]. Thus, when our method is deployed in real world, these travel time estimation methods can be used directly. In experiments, we carefully evaluate the influence of travel time estimation errors to our dynamic ambulance redeployment method, i.e. the robustness of our method to traffic conditions. We find that by using the off-the-shelf travel time estimation method [36], the influence of estimation errors to our method can be ignored.

Factor 4. $tt_i^1, \cdots, tt_i^k$ denote the travel time for each occupied ambulance to reach station i. Like tt_i, $tt_i^1, \cdots, tt_i^k$ are also estimated values. We only consider occupied ambulances which are transporting patients to hospitals. The reason is for each ambulance j of these ambulances, we can estimate its travel time to the hospital from its current location, the time at the hospital, and travel time from the hospital to station i, forming tt_i^j. For ambulances that are going to the scenes of patients, we don't know which hospitals they will go to, thus we cannot estimate their time to station i.

Although the number of occupied ambulances is time-varying, we need to fix k to make the length of the input in the score network fix. To this end, we denote the number of current occupied ambulances by k'. If $k \leq k'$, $tt_i^1, \cdots, tt_i^k$ are the smallest k values of the $tt_i^1, \cdots, tt_i^{k'}$. Note that the closer the occupied ambulances to station i, the less necessary the current available ambulance is redeployed to station i. That is the reason we select the smallest k values. If $k > k'$, $tt_i^1, \cdots, tt_i^k$ are $tt_i^1, \cdots, tt_i^{k'}, 2, \cdots, 2$, i.e. the remaining $k - k'$ values are filled with 2 hours. That is, 2 hours are used to denote the absence of $k - k'$ occupied ambulances.

2.2 Reinforcement Learning Deep Score Network

In this section, we propose to use a reinforcement learning framework [18, 34], based on policy gradient [33], to learn the weights θ in the score network. This is due to that given a station i's (dynamic) factors x_i, we do not have labelled score y_i. Thus we can't learn the θ using any supervised learning algorithms.

2.2.1 *Reinforcement Learning Framework.*

The dynamic ambulance redeployment problem can be modelled as a reinforcement learning task. For a reinforcement learning task, there exist five main concepts, i.e. *state, action, transition, reward,* and *policy*. Below, we introduce these five concepts for the dynamic ambulance redeployment problem.

State. When an ambulance becomes available, we can obtain the current state of the system, denoted by s_t. The current state s_t should include all information related with the redeployment of the current available ambulance. Thus, s_t consists of the factors of all ambulance stations. That is,

$$s_t = (x_1, x_2, \cdots, x_I), \tag{3}$$

where x_i refers to the factors of ambulance station i as defined in Equation 2, and I denotes the total number of ambulance stations in the EMS system.

Action. In the ambulance redeployment problem, the action is a station to which we redeploy the current available ambulance. Hence, the current action, denoted by a_t, is

$$a_t \in \{1, 2, \cdots, I\}, \tag{4}$$

where $a_t = i$ means the current available ambulance is redeployed to station i.

Transition. Given the current state s_t, after an action a_t is taken, we do not need to do anything until there is another ambulance becoming available. That is, the system will transit to the next state s_{t+1}, when there is a new ambulance becoming available. For the example in Figure 3, the current time step is t and current time slot is 10:13 am. If a new ambulance becomes available at 10:27 am, the system will transit to state s_{t+1} at 10:27 am. Time intervals between two states are not fix.

Reward. Given the current state s_t and the action a_t taken, the EMS system will output an instant scalar reward $r(s_t, a_t)$ when it transits to next state s_{t+1}. In this problem, the reward $r(s_t, a_t)$ is defined as the number of patients being picked up using pickup time less than a given time threshold (e.g. 10 minutes), between state s_t and s_{t+1}. For the same example in Figure 3, there are two patients being picked up within 10 minutes, and one patient

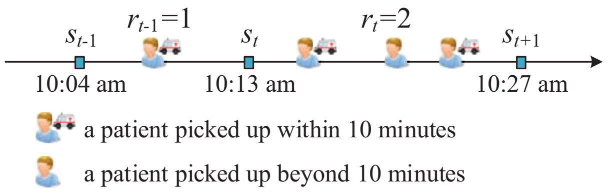

s_{t-1}
r_{t-1}=1
s_t
r_t=2
s_{t+1}
10:04 am
10:13 am
10:27 am
a patient picked up within 10 minutes
a patient picked up beyond 10 minutes

Fig. 3. Transitions between states.

being picked up beyond 10 minutes, between state s_t and s_{t+1}. Thus, in this situation, we have reward $r(s_t, a_t) = 2$. While between s_{t-1} and s_t, $r(s_{t-1}, a_{t-1}) = 1$. For simplicity of notations, we use r_t to stand for $r(s_t, a_t)$.

Policy. Given the current state s_t, the policy $\pi_\theta(s_t, a_t)$ is used to select an action. Specifically, $\pi_\theta(s_t, a_t)$ is a probability function, describing the probability of taking an action a_t given the current state s_t. In this work, we use a policy network for the policy $\pi_\theta(s_t, a_t)$, as shown in Figure 4. Given the current state $s_t = (x_1, x_2, \cdots, x_I)$, the policy network first calculates the score of each station i, i.e. $y_i = f(x_i; \theta)$. To this end, the score network in Figure 2 is embedded in the policy network. The policy network contains only one set of weights θ, i.e. different stations share one set of parameters θ. This is because that we are going to learn one score network for all stations, not one score network for each station. After the calculation of each station i's score y_i, a softmax function is used to get the probability $\pi_\theta(s_t, a_t = i)$ for each possible action i (station i). The softmax function is widely used in reinforcement learning [33, 34]. Specifically, $\pi_\theta(s_t, a_t = i)$ is

$$\pi_\theta(s_t, a_t = i) = \frac{\exp(f(x_i; \theta))}{\sum_{i'=1}^{I} \exp(f(x_{i'}; \theta))}. \tag{5}$$

The larger a station's score, the higher the probability that the current available ambulance is redeployed to it. As will be detailed in Section 2.2.2, the objective of the reinforcement learning is to learn an optimal policy network (i.e. policy $\pi_\theta(s_t, a_t)$) such that by following the policy, the rewards obtained can be maximized. Then, the learned policy network contains the weights of the score network, which are what we really want.

2.2.2 *Learning θ with Policy Gradient.*

Objective. The objective of reinforcement learning is to learn an optimal policy network (i.e. optimal weights θ) such that given any state s, by following the policy π_θ, the agent is able to obtain the *maximum expected long-term discounted reward*, i.e. picking up the most number of patients within a given time threshold [34]. Formally, the objective function is formulated as

$$\max_\theta \ J(\theta) = \mathbb{E}_{s \sim \pi_\theta}[v(s)], \tag{6}$$

where $v(s)$ refers to the *expected long-term discounted reward* starting from state s and following policy π_θ. $s \sim \pi_\theta$ means state s is sampled by following policy π_θ, starting from any random state. Mathematically, $v(s)$ is

$$v(s) = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s = s_t], \tag{7}$$

where $\gamma \in [0, 1]$ is the discounted ratio (e.g. $\gamma = 0.99$) of future rewards. $v(s)$ is also called state value [34]. Besides, for any state s and taken action a, we can define the expected long-term discounted rewards by following policy π_θ, denoted by $q(s, a)$, as

$$q(s, a) = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s = s_t, a = a_t]. \tag{8}$$

$q(s, a)$ is called state-action value [34]. According to the definitions [34], the relation between state value $v(s)$ and state-action value $q(s, a)$ is

$$v(s) = \sum_{a \in A} \pi_\theta(s, a) q(s, a). \tag{9}$$

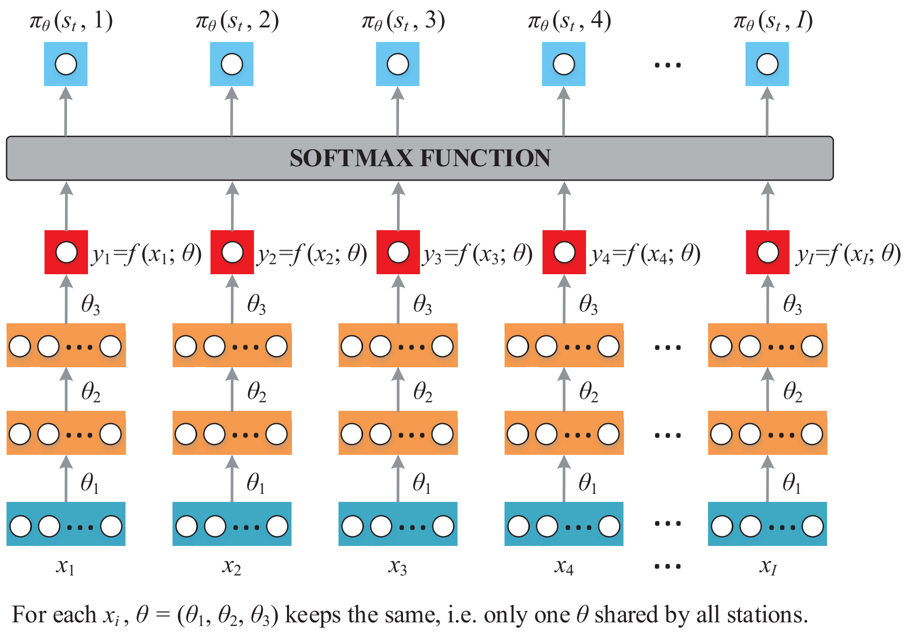
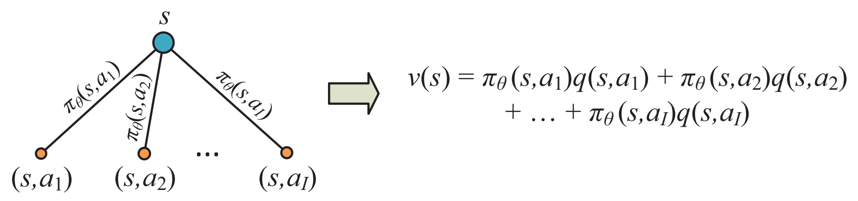


Fig. 4. Policy network.

The relation between $v(s)$ and $q(s, a)$ can be understood using Figure 5. Given state s, by following policy π_θ, each action is taken with probability $\pi_\theta(s, a)$, associated with state-action value $q(s, a)$. Thus, state value $v(s)$ is the probability expectation of state-action value $q(s, a)$, i.e. the Equation 9.



s
$\pi_\theta(s,a_1)$
$\pi_\theta(s,a_2)$
$\pi_\theta(s,a_I)$
(s,a_1)
(s,a_2)
(s,a_I)
$v(s) = \pi_\theta(s,a_1)q(s,a_1) + \pi_\theta(s,a_2)q(s,a_2) + \ldots + \pi_\theta(s,a_I)q(s,a_I)$

Fig. 5. The relation between $v(s)$ and $q(s, a)$.

Gradient. To maximize the objective function, policy gradient algorithms [18, 34] can be used. Specifically, combining Equations 6 and 9 and according to [33, 34], we can derive the gradient of $J(\theta)$ to θ as

$$\nabla_\theta J(\theta) = \mathbb{E}_{(s,a)\sim\pi_\theta}\left[(\nabla_\theta \log \pi_\theta(s, a)) \cdot q(s, a)\right], \tag{10}$$

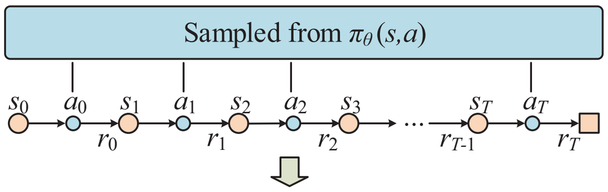
where $(s, a) \sim \pi_\theta$ denotes state-action pair (s, a) which is sampled by following policy π_θ, starting from any random state. The derivation is in Appendix A. Based on the gradient in Equation 10, we can update θ as

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta), \tag{11}$$

where α is the learning rate (e.g. $\alpha = 0.005$). It should be noted that in this problem, we are going to maximize the $J(\theta)$. Thus we update θ using $\theta + \alpha \cdot \nabla_\theta J(\theta)$, instead of $\theta - \alpha \cdot \nabla_\theta J(\theta)$.

Challenge. However, the expectation in Equation 10 is hard to obtain due to the complexity of the environment (the EMS system), in which $(s, a) \sim \pi_\theta$ and $q(s, a)$ cannot be modelled mathematically.

Solution. Monte-Carlo Sampling. To address this issue, in this paper, we apply a policy gradient algorithm, called REINFORCE with baseline [33, 34], as demonstrated in Algorithm 1. Specifically, to calculate the gradient in Equation 10, Monte-Carlo sampling (Figure 6) is used to sample state-action pairs (i.e. $(s, a) \sim \pi_\theta$) and state-action values (i.e. $q(s, a)$), based on which we can obtain the gradient. Given any random initial state s_0, we sample an action a_0 using current policy network $\pi_\theta(s, a)$. The state will then transit to the next state s_1, and then a_1 will be sampled similarly. The process continues until the maximum number of steps is reached, i.e. T steps. And we obtain the state-action pairs and values, as shown in Figure 6.



$$(s_0, a_0) — q^{sample}(s_0, a_0) = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots + \gamma^T r_T$$

$$(s_1, a_1) — q^{sample}(s_1, a_1) = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots + \gamma^{T\text{-}1} r_T$$

$$(s_2, a_2) — q^{sample}(s_2, a_2) = r_2 + \gamma r_3 + \gamma^2 r_4 + \ldots + \gamma^{T\text{-}2} r_T$$

$$\vdots$$

$$(s_T, a_T) — q^{sample}(s_T, a_T) = r_T$$

Fig. 6. Monte-Carlo sampling for obtaining state-action pair and value samples.

Solution. Approximate Gradient. Thus, the gradient in Equation 10 can be rewritten as

$$\nabla_\theta J(\theta) \approx \frac{1}{T+1} \sum_{t=0}^{T} \left[(\nabla_\theta \log \pi_\theta(s_t, a_t)) \cdot q^{sample}(s_t, a_t) \right]. \tag{12}$$

If using Equation 12, the algorithm is called REINFORCE. However, according to literature [33, 34], REINFORCE with baseline works better, which adds a baseline. Thus, in this work, we use

$$\nabla_\theta J(\theta) \approx \frac{1}{T+1} \sum_{t=0}^{T} \left[(\nabla_\theta \log \pi_\theta(s_t, a_t)) \cdot \left(q^{sample}(s_t, a_t) - \overline{q^{sample}} \right) \right], \tag{13}$$

where $\overline{q^{sample}}$ is the average of $q^{sample}(s_t, a_t)$, i.e. $\overline{q^{sample}} = \frac{1}{T+1} \sum_{t=0}^{T} q^{sample}(s_t, a_t)$. $\overline{q^{sample}}$ works as the baseline [33, 34].

Solution. Algorithm. Finally, we can use Equation 11 and Equation 13 to update the weights θ. Given the random initial weights θ, the update process is repeated until θ converge, as demonstrated in Algorithm 1. Lines 4-6 form a learning episode of the reinforcement learning. With more episodes, the θ will converge.

2.3 Dynamic Redeployment Algorithm

The learned policy network is actually an algorithm for ambulance redeployment. Thus, once learned, the policy network can be used to redeploy available ambulances. However, the policy network use softmax function to

Algorithm 1: Learning θ using REINFORCE with baseline

```
1: procedure LearningScoreNet
2:     $\theta \longleftarrow$ random initialization
3:     repeat            %% An episode
4:         sample:    state-action pairs and values          ▷ Figure 6
5:         calculate: gradient $\nabla_\theta J(\theta)$     ▷ Equation 13
6:         update:    $\theta \longleftarrow \theta + \alpha \cdot \nabla_\theta J(\theta)$     ▷ Equation 11
7:     until $\theta$ converge
8:     return $\theta$
```

select an action (station), which may introduce some randomness. During the learning of the score network, the randomness explores more state-actions, which is helpful for the learning and thus is preferred. However, once learned, the randomness will confuse the staffs in an EMS system. Thus, we derive a redeployment algorithm based on the learned score network, as depicted in Algorithm 2. Our redeployment algorithm redeploys an ambulance to the station with the highest score. Clearly, our redeployment algorithm is more staff-friendly. According to our experiment results, the difference between the performance of our redeployment algorithm (Algorithm 2) and the performance of directly using the policy network is neglectable. Our redeployment algorithm is triggered whenever there is an ambulance becoming available and ready to be redeployed.

Algorithm 2: Dynamic Ambulance Redeployment Algorithm

```
1: procedure Redeploy
2:     obtain each station $i$'s current factors $x_i$
3:     calculate each station $i$'s current score $y_i = f(x_i; \theta)$
4:     get the selected station $i^* = \arg\max_i \{y_i = f(x_i; \theta)\}$
5:     return $i^*$
```

As shown in Algorithm 2, our redeployment algorithm consists of three steps to redeploy an available ambulance. First, we obtain each station i's current factors x_i as Equation 2 (i.e. the current state of the EMS system). Second, we calculate the score y_i of each station i, using the score network, i.e. $y_i = f(x_i; \theta)$ as Equation 1. Third, we select the station i^* with the maximum score, to which the current available ambulance is redeployed. That is,

$$i^* = \arg\max_i \{y_i = f(x_i; \theta)\}. \tag{14}$$

3 EVALUATION

3.1 Dataset

Real-world data is leveraged to evaluate our dynamic ambulance redeployment method. Specifically, we collect data from the EMS system in city of Tianjin, China, which is comprised of EMS request records, road networks, ambulance stations, and hospitals, as detailed below.

EMS request records. EME request records contain the arrival of 120 calls (like 911 calls in USA) from patients. Each record consists of a time stamp, a latitude, and a longitude, recording the time slot and location of each patient. From October 1 to November 21, 2014, we collect 51 days of EMS request record data, containing 23,549 records. On average, every day ~462 EMS requests arrive and every hour has ~20 EMS requests. The spatial and temporal distribution of EMS requests are as depicted in Figure 7. Clearly, EMS requests are spatially and temporally unbalancing.

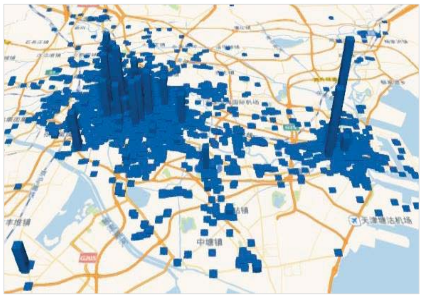
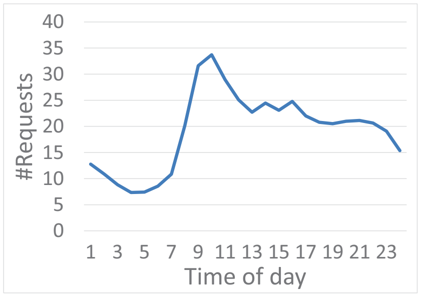



Fig. 7. The spatial (left) and temporal (right) distribution of EMS requests in Tianjin.

Ambulance stations. In Tianjin, there exist 34 ambulance stations with geographical locations, consisting of latitudes and longitudes.

Hospitals. There are 41 hospitals in Tianjin, to which ambulances will transport patients. Similarly, we have their geographical locations.

Road networks. The road network contains road information in Tianjin, including road vertices with latitudes and longitudes, road segments with road lengths and speed limits, etc. In total, there are 99,007 road vertices and 133,726 road segments. The road networks are used in our simulation below (Section 3.2) to define the real travel time between any two locations.

Fig. 8. Road networks in Tianjin (partial).

3.2 Simulation

We perform simulations to evaluate our ambulance redeployment method. Simulation is an effective way to evaluate ambulance redeployment methods, which has been widely used in a large number of previous works [16, 17, 22, 27, 35, 41, 43]. Besides ambulance redeployment problem, simulation is also widely used in other real-world decision making problems, like taxi sharing [21, 25], express services [42] and so on. Specifically, our simulation is built on top of previous simulations [23, 35, 43]. However, our simulation uses the real-world data aforementioned. In our simulation, each EMS request comes according to the real-world EMS request records, including time stamps and geographical locations. We use the previous 31 days of EMS requests as training data in our simulation to learn the deep score network, using the deep reinforcement learning algorithm (Algorithm 1). The latter 20 days is used as the test data to evaluate the redeployment algorithm (Algorithm 2) based on the learned deep score network, as shown in Figure 9. For any two locations l_1 and l_2, the travel time of ambulances from l_1 to l_2 is set as the minimum travel time aggregation among all road segments that concatenate l_1 and l_2 [8, 14, 37]. The travel time of ambulances on a road segment is set based on the length and the speed limit of the road segment.

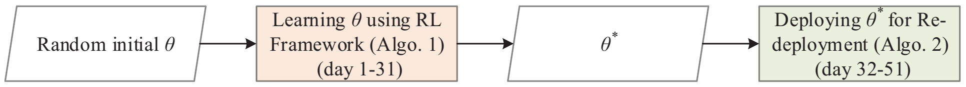

Random initial θ
Learning θ using RL Framework (Algo. 1) (day 1-31)
θ*
Deploying θ* for Re-deployment (Algo. 2) (day 32-51)

Fig. 9. The relation between the reinforcement learning algorithm 1 and the ambulance redeployment algorithm 2.

3.3 Baseline

To better evaluate the performance of our dynamic ambulance redeployment method, we compare our method (DRLSN: Deep Reinforcement Learning Score Network) with the following baseline redeployment methods.

(1) RS: The most naïve redeployment method is selecting a Random Station for an available ambulance each time.
(2) NS: If considering only factor 3, we will redeploy the available ambulance to the Nearest Station.
(3) LS: If considering only factor 2, the available ambulance will be redeployed to the Station with the Least ambulances.
(4) ERTM [32]: ERTM is an effective static ambulance redeployment method. Static redeployment methods first decide a base station for each ambulance. Then, when an ambulance becomes available, it is directly redeployed to its base station, regardless of the current factors of each station. Specifically, ERTM method assigns each ambulance a base station such that the expected travel time from stations to EMS requests is minimized. Clearly, it is a mathematical optimizaiton problem, and its assignment solution can be obtained using some optimization methods [32].
(5) MEXCLP [9]: MEXCLP is also a static redeployment method. According to literature [35], ERTM and MEXCLP perform the best among all static redeployment methods. Thus, we compare our method with these two methods. MEXCLP method assigns each ambulance a base station such that the expected coverage of stations is maximized. Similarly, this problem is also an optimization problem, whose solution can also be obtained by mathematical optimization [9].
(6) DMEXCLP [16]: DMEXCLP is an updated version of MEXCLP, which redeploys an available ambulance through considering the dynamic coverage of each station. DMEXCLP is the most state-of-the-art ambulance redeployment method.

3.4 Metric

According to previous literature, two metrics are frequently utilized to evaluate the performance of ambulance redeployment methods. The first one is the average pickup time of all patients [32, 35], which is denoted by AvePT. Formally, AvePT is formulated as

$$\text{AvePT} = \frac{1}{Q}\sum_{q=1}^{Q} pt_q, \tag{15}$$

where Q refers to the total number of requests for evaluation and pt_q denotes the pickup time of request q. The smaller the AvePT, the better a redeployment method.

Table 1. Comparisons with baselines.

	#ambu = 50		#ambu = 60		#ambu = 70		#ambu = 80		#ambu = 90	
	AvePT	RelaPT	AvePT	RelaPT	AvePT	RelaPT	AvePT	RelaPT	AvePT	RelaPT
RS	856.5	0.531	778.7	0.569	759.6	0.583	747.9	0.591	741.4	0.596
NS	773.1	0.585	767.3	0.589	753.2	0.602	747.6	0.607	736.2	0.616
LS	603.8	0.745	531.2	0.785	480.1	0.808	444.8	0.823	423.9	0.831
ERTM	505.2	0.786	432.9	0.830	398.8	0.844	389.5	0.848	384.1	0.850
MEXCLP	502.4	0.774	461.8	0.822	409.1	0.852	392.9	0.862	376.9	0.872
DMEXCLP	516.9	0.773	447.3	0.826	408.7	0.852	387.1	0.865	375.2	0.872
DRLSN (ours)	*402.8*	*0.838*	*367.0*	*0.864*	*351.2*	*0.874*	*342.0*	*0.879*	*338.0*	*0.880*

The second one is the ratio of patients being picked up within a time threshold [9, 16, 35], which we denote using RelaPT (relative pickup time). Formally, RelaPT is

$$\text{RelaPT} = \frac{1}{Q}\sum_{q=1}^{Q} ||pt_q \leq pt_*||. \tag{16}$$

$||pt_q \leq pt_*|| = 1$ if the pickup time pt_q of request q is less or equal to the threshold pt_* (e.g. 10 minutes); otherwise $||pt_q > pt_*|| = 0$. For a redeployment method, we expect a bigger RelaPT.

3.5 Effectiveness of Our Redeployment Method

This subsection studies the effectiveness of our dynamic ambulance redeployment method, comparing with many baseline methods. For each method, we conduct extensive simulations to obtain its performance, i.e. AvePT and RelaPT. The number of ambulances (denoted by #ambu) in an EMS system is set as $50, 60, \cdots, 90$. Comparison results are summarized in Table 1. We can find that our method is the best, in terms of both two metrics, defeating baseline methods remarkably. More in detail, when #ambu=50, the average pickup time of patients (AvePT) can be reduced to 402.8 seconds from 502.4 seconds (the best result of baseline methods). That is, our method can reduce 99.6 seconds of average pickup time, i.e. 1.6 minutes, and 19.8% of 502.4 seconds. The ratio of patients picked up within 10 minutes (i.e. RelaPT) has also been increased from 0.786 to 0.838. From another perspective, the performance of our method with #ambu=50 has already surpassed the performance of the best baseline with #ambu=60. That is, using our method, ambulances can be saved to obtain some certain performance levels. The improvement is such significant that we strongly believe that using our redeployment method, EMS systems in cities can significantly improve their abilities to protect people's lives from emergent accidents and diseases. Noted that for patients in danger every second counts and the quicker they are picked up, the more probable they can be saved.

3.6 Time Efficiency

Our ambulance redeployment method is realized in Python and are performed on a computer with 3.31 GHz Intel(R) Xeon(R) CPU and 16 GB. To train a score function takes just less than 10 hours using only CPUs, much more efficient than other deep reinforcement learning tasks using a huge number of GPUs [15, 24, 31]. After the score function has been learned, to real time redeploy an available ambulance takes just around a few milliseconds, which is obviously efficient enough to meet the real-world time efficiency requirement.

3.7 Convergency of Training

Figure 10 demonstrates the performance of our score function during the training process. It shows that the training is convgergent and that the score function can quickly converge to a near-optimal solution.

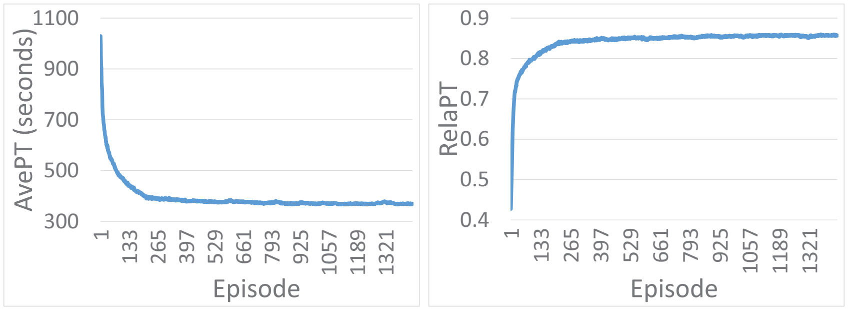


Fig. 10. Convergency of the learning. #ambu=60, $m = 1$, $k = 1$.

3.8 Necessity of Using Deep Score Networks

We study the necessity of using a deep score network as our score network in this subsection. Some traditional machine learning models are used for comparison, including ridge regression, lasso regression, and logistic regression. These models are shallow models. The learning of the parameters in these models is still under the reinforcement learning framework. As the results shown in Figure 11, our deep score network achieves clear advantages over these shallow models, indicating the necessity of using deep neural networks. This is due to that the complex factors considered in this work are hard to be well balanced using shallow models, since shallow models would easily result in under-fitting [3]. In contrast, deep neural networks with more layers have better representation ability [13, 19] and thus are able to learn the complex relation underlying the multiple factors.

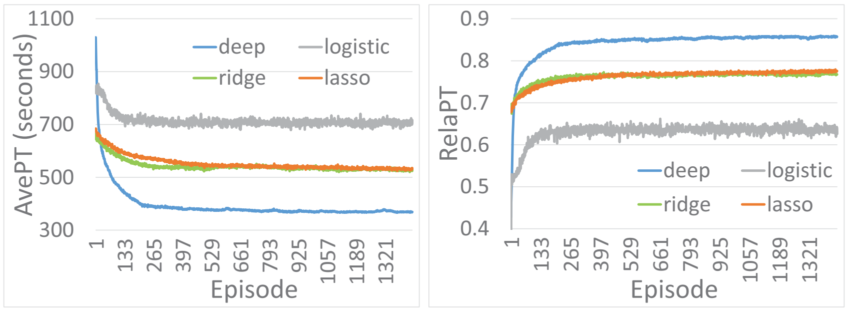


Fig. 11. The comparison between our deep score network with other shallow models. #ambu=60, $m = 1$, $k = 1$.

3.9 Necessity of Considering All Factors

In this subsection, we investigate the necessity for our ambulance redeployment method to consider the four complex factors of each station, as detailed in Section 2. Towards this end, we consider different combinations of the four factors, i.e. $\{12, 13, 14, 23, \cdots\}$ as depicted in Figure 12. For example, the x-axis with '12' means only factor 1 and factor 2 are considered for each station, i.e. $x_i = (\lambda_i^1, \cdots, \lambda_i^m, n_i)$. Then a new deep score network will be learned with inputs consisting of only factor 1 and factor 2. Finally we can evaluate the redeployment performance of the score network considering only factor 1 and factor 2. Similarly, the axis with '123' means we consider factors 1, 2, and 3. We conduct extensive experiments for our redeployment method considering different combinations of factors and obtain comparison results in Figure 12. Obviously, considering all four factors returns the best performance for both AvePT and RelaPT.

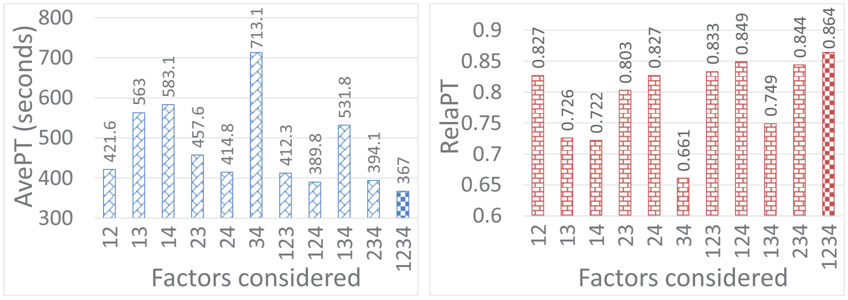


Fig. 12. The performance of our redeployment method considering different factors. #ambu=60, $m = 1$, $k = 1$.

Furthermore, we study the importance of each factor. We first define $\text{AvePT}_j = \frac{\sum_{fa \in FA_j} \text{AvePT}_{fa}}{|FA_j|}$ as the average pickup time of each factor j. FA_j refers to the set of factors containing factor j, for example, $FA_1 = \{12, 13, 14, 123, 124, 134\}$, $FA_2 = \{12, 23, 24, 123, 124, 234\}$. AvePT_{fa} denotes the average pickup time of our redeployment method considering factors fa, e.g. $\text{AvePT}_{12} = 421.6$, $\text{AvePT}_{13} = 563.0$ as depicted in the left part of Figure 12. Similarly, we can define $\text{RelaPT}_j = \frac{\sum_{fa \in FA_j} \text{RelaPT}_{fa}}{|FA_j|}$ for each factor j. Then AvePT_j and RelaPT_j can be used to indicate the importance of each factor j. The smaller (bigger) the AvePT_j (RelaPT_j), the more important the factor j. From Figure 13, we can see that factor 2 is the most important, followed by factor 1. This agrees with our common sense that the current number of ambulances in a station and the number of future requests nearby a station are the most important two factors when making a decision on whether an available ambulance should be redeployed to the station.

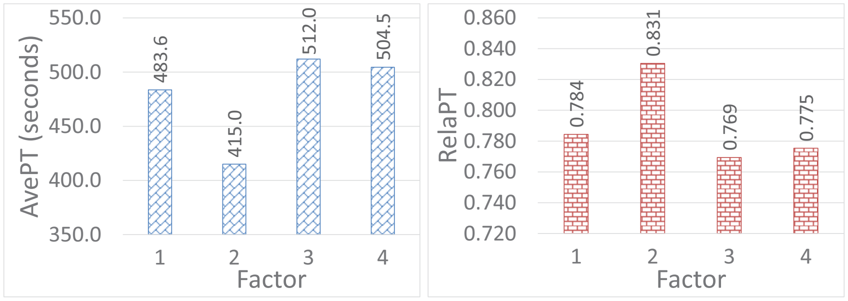


Fig. 13. Importance of each factor. #ambu=60, $m = 1$, $k = 1$.

3.10 Influence of Parameter m and k

For factor 1 and factor 4, there are two parameters m and k needed to be discussed, where we consider m time periods of future requests nearby a station and k occupied ambulances. Here, we study the influence of different

m's and k's to our redeployment method. The experimental results are summarized in Figure 14. Specifically, we consider different pairs of (m, k), i.e. the x-axis in Figure 14, and get the AvePT and RelaPT (y-axes) after the score network becomes convergent and stable. As demonstrated in Figure 14, different m's and k's don't present clear differences on the performance of our redeployment method. It implies that our redeployment method is robust to different settings of m and k.

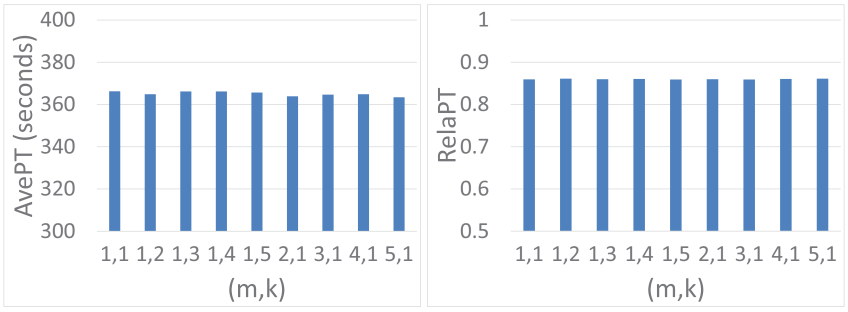


Fig. 14. Influence of parameter m and k to our redeployment method. #ambu=60.

3.11 Influence of Patient Amount

In this subsection, we study the performance of our method to the amount of patients (requests) in different time periods of a day. For each patient, dispatching an ambulance from the nearest station (in terms of travel time) is the optimal, using the least pickup time. However, the nearest station may lack ambulances and then an ambulance from other stations should be dispatched, which will result in extra pickup time. Thus, we define the ratio of the AvePT to the optimal AvePT in a time period (i.e. an hour), denoted by $\text{Ratio}_{\text{AvePT}} = \frac{\text{AvePT}}{\text{AvePT}_{\text{optimal}}}$. The $\text{AvePT}_{\text{optimal}}$ is the average pickup time of patients, under the assumption that all patients are picked up by ambulances in the nearest stations. Similarly, we can define the ratio of the RelaPT to the optimal RelaPT in each hour, i.e. $\text{Ratio}_{\text{RelaPT}} = \frac{\text{RelaPT}}{\text{RelaPT}_{\text{optimal}}}$. The calculation of $\text{RelaPT}_{\text{optimal}}$ is under the same assumption of calculating $\text{AvePT}_{\text{optimal}}$. Clearly, we expect a smaller $\text{Ratio}_{\text{AvePT}}$ and a bigger $\text{Ratio}_{\text{RelaPT}}$. The relation between these two ratios and the number of patients in an hour is presented in Figure 15.

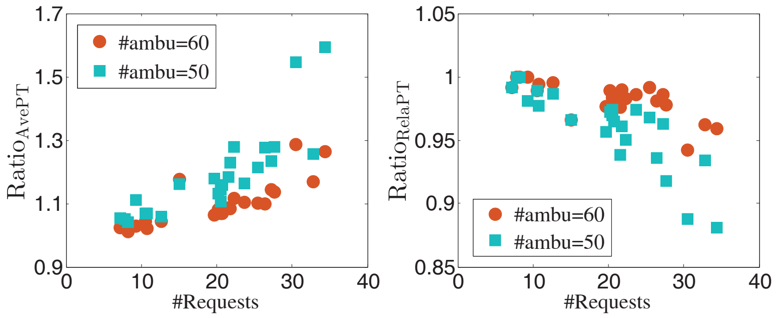


Fig. 15. Influence of patient amount to our redeployment method. $m = 1$, $k = 1$.

As demonstrated in Figure 15, in those time periods with more patients, for example from 9 a.m. to 11 a.m. (see Figure 7), the $\text{Ratio}_{\text{AvePT}}$ is large, indicating that many patients are not picked up by ambulances in the nearest stations, spending more pickup time. Similarly, when a time period has many patients, the $\text{Ratio}_{\text{RelaPT}}$ decreases, i.e. less patients are picked up within 10 minutes. It implies that more ambulances are needed in those time periods with many patients. Indeed, as shown in Figure Figure 15, an EMS system with #ambu=60 achieves much better performance than that with #ambu=50 in time periods with more patients. In addition, we can study how to automatically add or reduce ambulances in different time periods of a day, such that both the pickup time of patients and the operation cost of EMS systems can be reduced. We will study this interesting topic in future.

3.12 Robustness of Our Redeployment Method

3.12.1 Robust to Traffic Conditions. Time-evolving traffic conditions will make the estimation of travel time not accurate, i.e. the factor 3 and factor 4 (tt_i and $tt_i^1, \cdots, tt_i^k$) will be with noises. Thus, it is necessary to study the robustness of our redeployment method to traffic conditions (travel time estimation errors). We assume that the estimated travel time tt_i follows a Gaussian distribution (similar for $tt_i^1, \cdots, tt_i^k$). That is, $tt_i \sim \mathcal{N}\left(\tilde{tt}_i, (\varepsilon\tilde{tt}_i)^2\right)$, where $\mathcal{N}$ is a Gaussian distribution, $\tilde{tt}_i$ denotes the actual travel time, and $\varepsilon \geq 0$ is the travel time estimation error rate. Obviously, for a travel time estimation method, the smaller the error rate ε, the more accurate its estimation results. According to the state-of-the-art travel time estimation methods [36], the error rate ε has been reduced to less than 0.15. Specifically, due to $tt_i \sim \mathcal{N}\left(\tilde{tt}_i, (\varepsilon\tilde{tt}_i)^2\right)$, $|\frac{tt_i-\tilde{tt}_i}{\tilde{tt}_i}|$ is a half-normal distribution [39], with expectation $\mathbb{E}[|\frac{tt_i-\tilde{tt}_i}{\tilde{tt}_i}|] = \varepsilon\sqrt{2}/\sqrt{\pi}$. According to [36], MAPE $= \mathbb{E}[|\frac{tt_i-\tilde{tt}_i}{\tilde{tt}_i}|] < 0.12$. Thus, we can obtain MAPE $= \varepsilon\sqrt{\pi}/\sqrt{2} < 0.12$, resulting in $\varepsilon < 0.15$.

Figure 16 presents the performance of our redeployment method with travel time estimation errors. As depicted in Figure 16, our method is highly robust to the travel time estimation error (traffic conditions). The AvePT (RelaPT) increases (decreases) slowly with error rate ε. Using the state-of-the-art travel time estimation method [36] with $\varepsilon < 0.15$, the influence of estimation errors can be ignored.

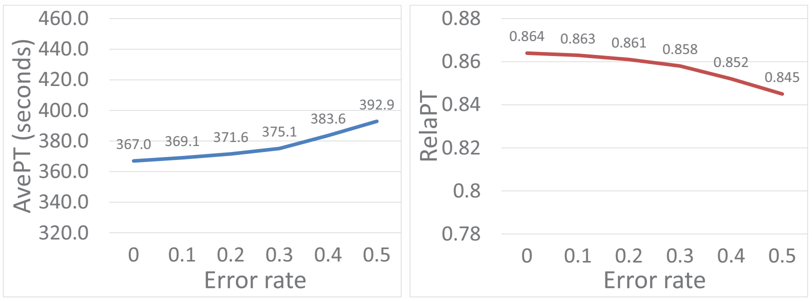


Fig. 16. Robust to traffic conditions (i.e. travel time estimation errors). #ambu=60, $m = 1$, $k = 1$.

3.12.2 Robust to #ambu's. In this subsection, we investigate the robustness of our method to different #ambu's in different EMS systems. More specifically, we study whether the learned score network under one setting can be applied to EMS systems under other settings. We learn a score network in an EMS system with #ambu=60, and then apply the learned score network to EMS systems with different #ambu's, e.g. 50, 60, $\cdots$, 90. Experiment results are demonstrated in Figure 17. Results with label '#ambu=60' denote the performance of the score network learned under #ambu=60. Results with '#ambu=50-90' refer to the performance of score networks learned under corresponding #ambu's, i.e. the result of DRLSN in Table 1. From these results, we can see that the score network learned under #ambu=60 performs equally excellent with score networks learned under corresponding #ambu's. This means we can learn just one score network under one setting for many different EMS systems (even in different cities), demonstrating its excellent transferability and broad applications in real life.

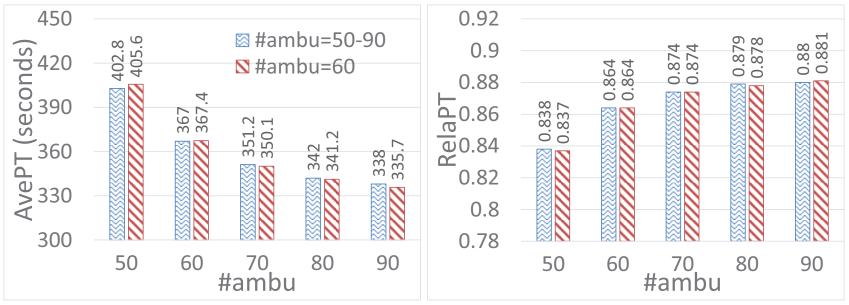


Fig. 17. Robust to #ambu's. '#ambu=60' denote the performance of the score network learned under #ambu=60. '#ambu=50-90' denotes the performance of the score networks learned under corresponding #ambu's. $m = 1$, $k = 1$.

3.12.3 Robust to Human Factors. We investigate the situations in which crews in ambulances disobey or delay the redeployment decisions obtained by our method. We define disobedience probability $prob^{dis}$ that an available ambulance is not redeployed to the station obtained by our method, but to a random station. Similarly, we can define delay probability $prob^{del}$ that an available ambulance is delayed to be redeployed to the station obtained by our method. We assume the duration of a delay is uniformly distributed within 30 minutes. Experiment results are in Figure 18. As crews are well trained in EMS systems, disobedience only happens occasionally, e.g. disobedience probability being less than 1/50. In this situation, as shown in Figure 18, the influence of occasional disobedience of crews to our method is small. Although delay happens slightly more times, the influence of delay grows slowly with the increase of delay probability. As a result, our method is also robust to human factors.

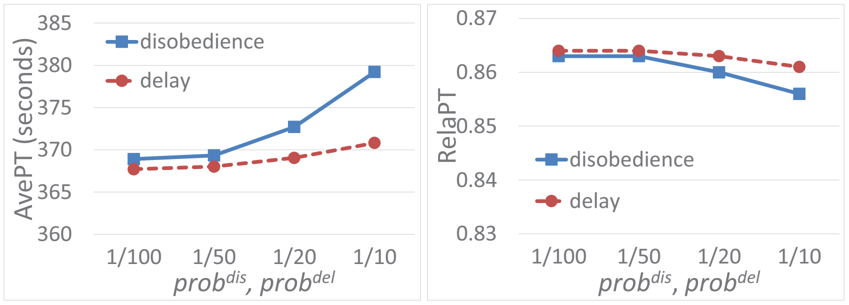


Fig. 18. Robust to human factors. #ambu=60, $m = 1$, $k = 1$.

4 RELATED WORK

4.1 Dynamic Ambulance Redeployment

Previous ambulance redeployment methods can be categorized into three groups. The first group of methods [9, 32, 35] determine a base station for each ambulance. Then when an ambulance becomes available, it is redeployed to its base station, regardless of any dynamic factor of each station. The second group of methods [10, 12, 26, 28, 41] also determine a base station for each ambulance, but the base station of an ambulance can be different at different time slots (e.g. 2 hours). These methods take some time-evolving factors into consideration, like traffic condition and EMS requests. However, the dynamic factors of each station are still not well considered. The last group is the dynamic ambulance redeployment methods [16, 22, 27], which don't determine base stations for ambulances and redeploy ambulances based on dynamic factors. However, previous methods don't well consider the multiple dynamic factors as aforementioned in our work. They only consider partial of the factors. Each factor considered in this work is important. To well balance these factors, we propose a deep score network, leveraging the great power of deep neural networks.

4.2 Deep Reinforcement Learning

Recently, deep reinforcement learning has achieved significant success in many sequential decision making problems [1, 18], e.g. Go [29, 31], Atari 2600 games [24], robotics [20], and so on. Combining deep learning and reinforcement learning, deep reinforcement learning is a revolutionary tool for artificial intelligence. Many deep reinforcement learning algorithms have been proposed [1, 15, 18, 24, 30]. In this work, we leverage the deep reinforcement learning framework to learn the deep score network, based on which we can design an effective dynamic ambulance redeployment algorithm. To the best of our knowledge, this is the first work that uses deep reinforcement learning to do the dynamic ambulance redeployment.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose a dynamic ambulance redeployment method. Specifically, we first propose a deep score network, which is then learned by a deep reinforcement learning algorithm. Based on the learned score network, finally we provide an effective dynamic ambulance redeployment algorithm. Using our method, the pickup times of patients can be significantly reduced. Comparing with baseline methods, our method can save around 20% (100 seconds) of average pickup time of patients, and improve the ratio of patients being picked up within 10 minutes from 0.786 to 0.838. Thus, it can be safely concluded that our method can help EMS systems in cities better protect citizens' lives from emergent accidents or diseases.

In the future, we will try to use the deep score network in other sequential decision making problems, e.g. taxi dispatching, food carrier dispatching, express carrier dispatching, and so on. We expect the idea of learning deep score networks can also perform well in these problems. Besides, currently we only study the ambulance redeployment strategy, making the ambulance dispatching strategy fix. In the future, we plan to study how to jointly dispatch and redeploy ambulances using deep score networks. Actually, the joint of dispatching and redeployment can be modelled as a multi-task reinforcement learning problem. To deal with this problem following the insight of this work (using deep score networks), there exist at least three important points that should be well considered. *First*, two score networks will be needed, one for the dispatching task and one for the redeployment task. *Second*, some multi-task reinforcement learning techniques will be needed for the joint learning of the two score networks. *Third*, the factors for the dispatching task should be carefully designed.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61672399, U1609217, 61773324), the Fundamental Research Funds for the Central Universities (Nos. 220710004005040177), and Doctoral Innovation Fund Program of Southwest Jiaotong University.

A DERIVATION OF THE GRADIENT OF THE OBJECTIVE FUNCTION $J(\theta)$

In this appendix, we show how to derive the gradient $\nabla_\theta J(\theta)$ of the objective function $J(\theta)$, i.e. how to obtain Equation 10. Based on Equation 6 and Equation 9, we have

$$J(\theta) = \mathbb{E}_{s\sim\pi_\theta}[v(s)] = \sum_{s\in S} p(s)v(s) = \sum_{s\in S} p(s) \sum_{a\in A} \pi_\theta(s,a)q(s,a) = \sum_{s\in S}\sum_{a\in A} p(s)\pi_\theta(s,a)q(s,a) = \mathbb{E}_{(s,a)\sim\pi_\theta}[q(s,a)].$$

$s \sim \pi_\theta$ (or $(s,a) \sim \pi_\theta$) means that state s (or state-action pair (s,a)) is sampled by following policy π_θ and starting a random initial state. Thus, $p(s)$ (or $p(s)\pi_\theta(s,a)$) denotes the probability of state s (or state-action pair (s,a)) by following policy π_θ and starting a random initial state. Then, $\nabla_\theta J(\theta)$ is

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{a \in A} p(s) \nabla_\theta \pi_\theta(s, a) q(s, a) \tag{17}$$

$$= \sum_{s \in S} \sum_{a \in A} p(s) \pi_\theta(s, a) \nabla_\theta \log \pi_\theta(s, a) q(s, a) \tag{18}$$

$$= \mathbb{E}_{(s,a)\sim\pi_\theta} \left[\left(\nabla_\theta \log \pi_\theta(s, a) \right) q(s, a) \right]. \tag{19}$$

From Equation 17 to Equation 18, there is a mathematical transform

$$\nabla_\theta \log \pi_\theta(s, a) = \nabla_\theta \pi_\theta(s, a) / \pi_\theta(s, a). \tag{20}$$

Thus, Equation 20 holds. The derivation can also be found in literature [34].

REFERENCES

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine* 34 (2017), 26–38.

[2] ASIRT. 2018. *Road Crash Statistics.* http://asirt.org/Initiatives/Informing-Road-Users/Road-Safety-Facts/Road-Crash-Statistics.

[3] Christopher M. Bishop. 2007. *Pattern recognition and machine learning, 5th Edition.* Springer. http://www.worldcat.org/oclc/71008143.

[4] Tianjin EMS center. 2015. *Emergency medical services center of Tianjin.* http://tianjin.emss.cn/hj.htm.

[5] Albert Y. Chen, Tsung-Yu Lu, Matthew Huei-Ming Ma, and Wei-Zen Sun. 2016. Demand Forecast Using Data Analytics for the Preallocation of Ambulances. *IEEE Journal of Biomedical and Health Informatics* 20, 4 (July 2016), 1178–1187.

[6] Richard Church and Charles R. Velle. 1983. The Maximal Covering Location Problem. *Regional Science Association* 32 (1983), 101–118.

[7] Cleveland Clinic. 2018. *Sudden Cardiac Statistics.* https://my.clevelandclinic.org/health/diseases/17522-sudden-cardiac-death-sudden-cardiac-arrest.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms (3. ed.).* MIT Press.

[9] Mark S. Daskin. 1983. A Maximum Expected Covering Location Model: Formulation, Properties and Heuristic Solution. *Transportation Science* 17, 1 (1983), 48–70.

[10] Dirk Degel, Lara Wiesche, Sebastian Rachuba, and Brigitte Werners. 2015. Time-dependent ambulance allocation considering data-driven empirically required coverage. *Health Care Management Science* 18 (2015), 444–458.

[11] Heart Foundation. 2018. *Sudden Cardiac Statistics 2.* https://www.heartfoundation.org.au/your-heart/sudden-cardiac-death.

[12] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. 2001. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Comput.* 27, 12 (September 2001), 1641–1653.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* MIT Press.

[14] 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 (July 1968), 100–107.

[15] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

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

[17] C. J. Jagtenberg, S. Bhulai, and R. D. van der Mei. 2017. Optimal Ambulance Dispatching. *Operations Research for Health Care* 248 (March 2017), 269–291.

[18] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* 4 (1996), 237–285.

[19] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[20] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research* 17 (2016), 39:1–39:40.

[21] Shuo Ma, Yu Zheng, and Ouri Wolfson. 2015. Real-Time City-Scale Taxi Ridesharing. *IEEE Transactions on Knowledge and Data Engineering* 27 (2015), 1782–1795.

[22] Matthew S. Maxwell, Mateo Restrepo, Shane G. Henderson, and Huseyin Topaloglu. 2010. Approximate Dynamic Programming for Ambulance Redeployment. *INFORMS Journal on Computing* 22, 2 (May 2010), 266–281.

[23] Richard McCormack and Graham Coates. 2015. A simulation model to enable the optimization of ambulance fleet allocation and base station location for increased patient survival. *European Journal of Operational Research* 247 (May 2015), 294–309.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (February 2015), 529–533.

[25] Meng Qu, Hengshu Zhu, Junming Liu, Guannan Liu, and Hui Xiong. 2014. A cost-effective recommender system for taxi drivers. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 45–54.

[26] Sandhya Saisubramanian, Pradeep Varakantham, and Hoong Chuin Lau. 2015. Risk Based Optimization for Improving Emergency Medical Systems. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. 702–708.

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

[28] Verena Schmid and Karl F. Doerner. 2010. Ambulance location and relocation problems with time-dependent travel times. *European Journal of Operational Research* 207, 3 (October 2010), 1293–1303.

[29] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabi. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[30] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. 2014. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31th International Conference on Machine Learning*. 387–395.

[31] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550 (2017), 354–359.

[32] Lawrence V. Snyder and Mark S. Daskin. 2004. Reliability Models for Facility Location: The Expected Failure Cost Case. *Transportation Science* 39, 3 (August 2004), 400–416.

[33] Richard S. Sutton and Andrew G. Barto. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8 (May 1992), 229–256.

[34] Richard S. Sutton and Andrew G. Barto. 2017. *Reinforcement Learning: An Introduction (Second Edition, in progress)*. MIT Press, Cambridge, MA.

[35] Pieter L. van den Berg, J. Theresia van Essen, and Eline J. Harderwijk. 2016. Comparison of static ambulance location models. In *Proceedings of the 3rd International Conference on Logistics Operations Management*. 1040–1051.

[36] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

[37] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 25–34.

[38] Bradford S. Westgate, Dawn B. Woodard, David S. Matteson, and Shane G. Henderson. 2016. Large-network travel time distribution estimation for ambulances. *European Journal of Operational Research* 252 (2016), 322–333.

[39] Wikipedia. 2018. *Half-normal distribution*. https://en.wikipedia.org/wiki/Half-normal_distribution.

[40] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*. 99–108.

[41] Yisong Yue, Lavanya Marla, and Ramayya Krishnan. 2012. An Efficient Simulation-Based Approach to Ambulance Fleet Allocation and Dynamic Redeployment. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

[42] Siyuan Zhang, Lu Qin, Yu Zheng, and Hong Cheng. 2016. Effective and Efficient: Large-Scale Dynamic City Express. *IEEE Transactions on Knowledge and Data Engineering* 28 (2016), 3203–3217.

[43] Lu Zhen, Kai Wang, Hongtao Hu, and Daofang Chang. 2014. A simulation optimization framework for ambulance deployment and relocation problems. *Computers & Industrial Engineering* 72 (March 2014), 12–23.

[44] Zhengyi Zhou and David S. Matteson. 2015. Predicting Ambulance Demand: a Spatio-Temporal Kernel Approach. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2297–2303.

Received August 2018; revised November 2018; accepted January 2019