European Journal of Operational Research 291 (2021) 296–308

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

Multi-agent reinforcement learning algorithm to solve a partially-observable multi-agent problem in disaster response

Hyun-Rok Lee[a], Taesik Lee[a,*]

Department of Industrial & Systems Engineering, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon, South Korea

ARTICLE INFO

Article history:
Received 20 August 2019
Accepted 12 September 2020
Available online 18 September 2020

Keywords:
OR in disaster relief
Artificial intelligence
Multi-agent reinforcement learning
Imitation learning
Selective patient admission

ABSTRACT

Disaster response operations typically involve multiple decision-makers, and each decision-maker needs to make its decisions given only incomplete information on the current situation. To account for these characteristics – decision making by multiple decision-makers with partial observations to achieve a shared objective –, we formulate the decision problem as a decentralized-partially observable Markov decision process (dec-POMDP) model. To tackle a well-known difficulty of optimally solving a dec-POMDP model, multi-agent reinforcement learning (MARL) has been used as a solution technique. However, typical MARL algorithms are not always effective to solve dec-POMDP models. Motivated by evidence in single-agent RL cases, we propose a MARL algorithm augmented by pretraining. Specifically, we use behavioral cloning (BC) as a means to pretrain a neural network. We verify the effectiveness of the proposed method by solving a dec-POMDP model for a decentralized selective patient admission problem. Experimental results of three disaster scenarios show that the proposed method is a viable solution approach to solving dec-POMDP problems and that augmenting MARL with BC for its pretraining seems to offer advantages over plain MARL in terms of solution quality and computation time.

1. Introduction

In the aftermath of a disaster, response operations typically involve a large number of responders of different functions at multiple locations. These responders are required to make various decisions in a highly resource-constrained environment (Timbie et al., 2013). On site, first responders and emergency medical technicians triage casualties to determine priority for providing treatment and transportation. Triage decisions are based on the victim's injury severity and current resource level (Jenkins et al., 2008; Sacco et al., 2005; 2007). At hospitals, doctors and nurses may invoke a contingency plan to secure necessary resources for impending surge demand from disaster sites. A contingency plan typically includes canceling elective admissions, rescheduling surgeries, or discharging relatively less serious patients from an intensive care unit (Hick, Hanfling, & Cantrill, 2012; Peleg & Kellermann, 2009). An emergency department (ED) may selectively admit patients and transfer some of the arriving patients to other hospitals for better use of limited resources available at the ED.

In such environments, information available to each decision-maker is often incomplete, and decision-makers need to make various decisions based on partial, fragmented pieces of information (der Heide, 2006). For example, ambulance dispatchers may send ambulances to a disaster site without knowing the scale of the full incident or other dispatchers' decisions. While advanced communication systems can potentially alleviate this problem (Chan, Killeen, Griswold, & Lenert, 2004; Park, Jeong, Seo, & Kim, 2015; Seo, Jeong, Park, Kim, & Lim, 2015), there remain many communication challenges in disaster response situations (Fischer, Posegga, & Fischbach, 2016; Manoj & Baker, 2007; Sokat, Dolinskaya, Smilowitz, & Bank, 2018); damages in the communication infrastructure, lack of technology interoperability among various organizations and other technical and organizational barriers are commonly experienced during a disaster response phase, all of which can cause communication failures. Given all these challenges, it is unlikely that decision-makers in a disaster response system have access to complete information and knowledge about the current situation and other decision-makers' responses.

In the literature on disaster response decision making problems, most of the previous studies consider a single (or centralized) decision-maker environment and assume that the decision-maker has full access to necessary information. As such, it is often the case in those studies that a decision model focuses on a specific target decision problem while treating other elements as external factors, and given inputs to the model. See, for example, models based on Markov Decision Process (MDP) (e.g., Jacobson, Argon, & Ziya, 2012; Lee & Lee, 2018), integer programming (e.g.,

* Corresponding author.
E-mail addresses: hyunrok@kaist.ac.kr (H.-R. Lee), taesik.lee@kaist.edu (T. Lee).

https://doi.org/10.1016/j.ejor.2020.09.018
0377-2217/

Repoussis, Paraskevopoulos, Vazacopoulos, & Hupert, 2016; Sung & Lee, 2016; Wilson, Hawe, Coates, & Crouch, 2013), and fluid or queuing model (e.g., Cohen, Mandelbaum, & Zychlinski, 2014; Kilic, Dincer, & Gokce, 2014; Mills, Argon, & Ziya, 2013). As mentioned above, however, it is rather ideal to assume a fully observable environment and to isolate a single decision problem from the multitude and complexity of the entire decision making environment of disaster response; in reality, multiple decision-makers strive to maximize the number of survivors by deciding on their individual actions while provided only with limited, partial information.

Modeling wise, these disaster response decision making problems can be cast as a Decentralized-Partially Observable Markov Decision Process (dec-POMDP) model. Agents in a dec-POMDP model partially observe the system state and make decisions at each step to maximize the overall reward for all agents. Optimally solving a dec-POMDP model is well known to be a challenging problem (Bernstein, Givan, Immerman, & Zilberstein, 2002). Because agents in a dec-POMDP model do not have the true state information, agents must use all of previous observations history and actions chosen therein to determine optimal actions. This makes a dec-POMDP model computationally very expensive to solve.

In recent years, deep multi-agent reinforcement learning (deep-MARL) algorithms using deep neural networks (DNN) have been recognized as a promising approximate solution technique for dec-POMDP models. Yet, even with deep-MARL algorithms, multi-agent problems are still much harder to solve than single-agent problems; learning a well-coordinated policy for multi-agents is difficult because the quality of an individual agent's policy depends on other agents' policies and also because multiple agents are simultaneously learning an individual policy in MARL algorithms. Due to this difficulty, MARL algorithms require extensive exploration in learning, making them computationally very expensive (Foerster, Farquhar, Afouras, Nardelli, & Whiteson, 2018; Lowe et al., 2017; Tampuu et al., 2017). This problem is pronounced especially when we model a decision problem in a practical, real-world application environment, such as in this study, because such applications require significant complexity represented in a dec-POMDP model.

We tackle this challenge by, in the learning process, taking advantage of available knowledge or known decision rules relevant to the problem. This is motivated by evidence from single-agent RL literature (Cruz Jr, Du, & Taylor, 2018; Rajeswaran et al., 2018). In a standard MARL approach to obtain a policy solution for a dec-POMDP problem, an MARL algorithm learns a policy from scratch (i.e., initial policy). Our proposed method intends to facilitate MARL algorithm's early exploration by providing demonstrations that it can use as an initial reference. In our context, demonstrations are sample decisions –observations given to the agents and their chosen actions corresponding to those observations. These demonstrations can be obtained based on relevant domain knowledge or from existing decision rules for the dec-POMDP problem at hand. For example, for a patient prioritization problem, it is shown in the literature that when response resources are extremely limited, assigning higher priority to less severe patients is superior to prioritizing more severe patients (Jacobson et al., 2012; Lee & Lee, 2018; Mills et al., 2013; Sung & Lee, 2016).

As a means to incorporate reference demonstrations into an MARL algorithm, we use a behavioral cloning (BC) method. BC uses supervised learning to construct a policy from demonstration samples. Hence, BC is referred to as an imitation learning technique in that the resulting policy imitates the decision principles behind the demonstrations (i.e., reference policy). In our proposed method, we use BC with demonstration samples and construct an imitation policy to feed to an MARL algorithm as its initial policy. Note that the need to construct an imitation policy is relevant because a reference policy may not be explicitly known (e.g., human expert's knowledge) or its form is not implementable in a multi-agent partially observable environment (e.g., Multi-agent Markov Decision Process solution).

To demonstrate the proposed method, we consider a decentralized selective patient admission problem under partial-observation. Selective patient admission refers to a strategy where an ED diverts some of the arriving patients to other EDs to preserve its medical resources for future arrivals of severe patients (Einav et al., 2006; der Heide, 2006; Lee & Lee, 2018). In a selective patient admission problem, multiple emergency departments (ED's) respond to a multiple casualty incident (MCI), and these EDs share a common goal of maximizing the overall number of survivors from the MCI. Each ED makes an admission decision for sequentially arriving patients. In its decision making, EDs need various pieces of information including the severity of a currently arriving patient, number of available beds, elapsed time from the incident, and so on. Motivated by the potential difficulties in communication under a disaster situation, we assume a partially observable decision making environment.

Our contribution is twofold. First, formulating a decision problem as a dec-POMDP model is new to the disaster response operations research literature. Although a vast array of real-world emergency medical service (EMS) operations decisions are made in a multi-agent, partially-observable environment, a dec-POMDP model has not received much attention as a major modeling method. Thus, our work offers promising evidence for solving disaster operations decision problems with a dec-POMDP model. Second, to resolve the computational difficulty of a dec-POMDP model, we propose a novel solution method. Our proposed solution method uses demonstration samples from a reference policy to boost the early exploration of a standard MARL algorithm. The proposed solution method pretrains neural networks through BC and then updates an initial dec-POMDP policy with an MARL algorithm. Combining imitation learning and RL algorithm has been shown to be effective in single-agent environments, but its effectiveness has not been demonstrated in multi-agent environments. Our work serves as an example of the effectiveness of the integrated method in a multi-agent case. A practical benefit of this method is that we can take advantage of domain knowledge in an MARL algorithm by incorporating relevant heuristics or other decent policies, e.g., human experts decisions or triage rules in a disaster response protocols.

This paper is structured as follows. Related literature on disaster response and solution algorithms for dec-POMDP models is summarized in Section 2. In Section 3, we provide a brief background on dec-POMDP and describe our proposed method in detail. The proposed solution method is applied to a decentralized selective patient admission problem at multiple EDs under surge demand. A dec-POMDP model and the method to obtain demonstration samples for this problem are described in Section 4, followed by computational experiments in Section 5. Then we conclude our work in Section 6.

2. Related literature

MDP model expresses a decision making environment where an agent has a full observation of a system's state and chooses an action at each state. MDP has been widely used to model dynamic decision-making problems in diverse problem domains. For example, in healthcare operations management, there exist many examples of MDP applications to obtain a policy solution for better use of medical resources: resource allocation in a hospital (Gerchak, Gupta, & Henig, 1996; Green, Savin, & Wang, 2006; Huh, Liu, & Truong, 2013; Thompson, Nunez, Garfinkel, & Dean, 2009), patient admission management at a hospital (Chan, Farias, Bambos, & Escobar, 2012; Helm, AhmadBeygi, & Van Oyen, 2011), ambulance

diversion (Ramirez-Nafarrate, Hafizoglu, Gel, & Fowler, 2014), and many more.

In the operations research literature on emergency medical services (EMS) system's disaster response, MDP is also one of the popular modeling techniques. A patient prioritization problem is a prime example. In a patient prioritization problem, the goal is to determine the priority of patient transport and treatment when there are a greater number of patients than medical resources can provide care at once. Patient prioritization problems have often been modeled as a stochastic scheduling problem by considering patients as impatient jobs (Argon, Ziya, & Righter, 2008; Jacobson et al., 2012; Li & Glazebrook, 2010; 2011). Other examples of MDP applications in this area include a selective admission decision at an ED under surge demand (Lee & Lee, 2018) and ambulance dispatching and hospital selection problem for patient evacuation from MCI sites (Mills, Argon, & Ziya, 2018).

While MDP has been demonstrated to be an effective modeling tool for the EMS system operations research, we need a more sophisticated model to describe the full complexity of disaster response operations. In particular, we need a model to be able to represent multi-agents and incomplete information environment. To that end, a dec-POMDP model is a suitable model for modeling decision problems in a multi-agent, partially observable environment, and, to our best knowledge, there is no prior study in disaster response applications that formulates a decision problem as a dec-POMDP model. In this work, we study a multi-agent, partial observation problem in disaster response, formulate a dec-POMDP model and investigate a solution technique.

A dec-POMDP model is well known to be very difficult to solve. For a finite-horizon dec-POMDP model, various solution approaches have been proposed in the literature. In its early development, a dec-POMDP solution is represented as a policy tree, and dynamic programming and heuristic search techniques are used to identify and prune suboptimal trees in the policy tree (Amato, Dibangoye, & Zilberstein, 2009; Hansen, Bernstein, & Zilberstein, 2004; Oliehoek, Spaan, & Vlassis, 2008; Szer, Charpillet, & Zilberstein, 2005). More recently, sufficient statistics of past joint policies are used to compactly express a value function of a dec-POMDP model (Dibangoye, Amato, Buffet, & Charpillet, 2016; Oliehoek, 2013). Perhaps, the most recent, state-of-the-art dec-POMDP algorithm is a feature-based heuristic search value iteration (FB-HSVI) method (Dibangoye et al., 2016). It transforms a dec-POMDP model into a continuous state MDP using occupancy state as sufficient statistics. While these methods are exact, optimal algorithms, these approaches face a scalability problem. For an infinite-horizon or indefinite-horizon dec-POMDP, an optimal solution is unexpressible and hence its complexity is undecidable (Bernstein et al., 2002). As such, solution techniques to solve these problems are based on approximate, heuristic approaches. With recent advances in reinforcement learning algorithms, MARL[1] has drawn large attention as a viable approach to solving dec-POMDP problems.

Using MARL algorithms to compute an approximate policy solution is a relatively recent topic in the dec-POMDP literature, and various MARL algorithms have been reported to produce decent policies on many challenging dec-POMDP problems (Foerster et al., 2018; Gupta, Egorov, & Kochenderfer, 2017; Omidshafiei, Pazis, Amato, How, & Vian, 2017). For example, Omidshafiei et al. (2017) and Tampuu et al. (2017) use an independent learning approach where a policy solution for each agent is updated solely based on their individual experiences. Foerster, Assael, de Freitas, and Whiteson (2016) and Gupta et al. (2017) focus on developing a shared policy network, as opposed to individual policy networks for each agent, by sharing network parameters for all agents. Another stream of work is multi-agent actor-critic (AC) algorithms (Foerster et al., 2018; Lowe et al., 2017). In multi-agent AC, full state information (i.e., full observation or centralized environment) is used during the training phase, which is a standard approach in classical dec-POMDP algorithms (Oliehoek et al., 2008). A value network, referred to as a centralized critic, is trained with full state information to estimate state values under the current joint policy, and each agent learns its individual policy guided by the centralized critic. Recent studies show that a centralized critic that uses global information, improves the performance of RL in a decentralized environment (Foerster et al., 2018; Lowe et al., 2017). As multi-agent AC with a centralized critic is the latest development, we also use it as an MARL algorithm in our solution method.

To improve the performance of the multi-agent AC algorithm, we propose to integrate imitation learning and the multi-agent AC algorithm. Imitation learning is a method to obtain a decent policy solution by imitating a target, reference policy, or its demonstrations. Recent literature on single-agent RL suggests that the performance of RL algorithms can be improved by integrating imitation learning and RL, and its advantages of such an approach have been well demonstrated in single-agent problems. Expert demonstrations introduced by imitation learning can enhance the exploration in RL, thereby improving learning speed (Gao et al., 2018; Hester et al., 2018; Kang, Jie, & Feng, 2018; Lakshminarayanan, Ozair, & Bengio, 2016; Nair, McGrew, Andrychowicz, Zaremba, & Abbeel, 2018; Rajeswaran et al., 2018; Subramanian, Isbell Jr, & Thomaz, 2016; Vecerík et al., 2017), and augmenting an imitation learning by reinforcement learning helps overcome a limitation of imitation learning – i.e., quality of the policy solution from imitation learning is bounded by the quality of expert demonstrations used – (Cheng, Yan, Wagener, & Boots, 2018; Sun, Bagnell, & Boots, 2018; Sun, Venkatraman, Gordon, Boots, & Bagnell, 2017). Integrating imitation learning and RL can be implemented in a few ways. For example, one can use imitation learning to pretrain a network for a subsequent RL, or expert demonstrations can be mixed into training samples in an RL algorithm. While all of the above studies demonstrate the effectiveness of the integrated approach, it has not been sufficiently investigated in MARL applications (Lee & Lee, 2019).

3. Proposed solution method for a dec-POMDP problem

3.1. Background: Dec-POMDP model

Dec-POMDP is defined as a tuple $< I, S, \{A_i\}, \mathcal{P}, \mathcal{R}, \{\Omega_i\}, \mathcal{O}, K >$ (Oliehoek & Amato, 2016). I is a finite set of agents, S is a finite set of states, A_i is a finite set of actions for each agent $i \in I$, $\mathcal{P}$ is a state transition function, $\mathcal{R}$ is a reward function, Ω_i is a finite set of observations by each agent i, $\mathcal{O}$ is an observation function, and K is a decision horizon of the problem. We denote the number of agents as N; $|I| = N$.

At every decision epoch $k \in \{1, \ldots, K\}$, each agent i obtains individual observation $o_{i,k} \in \Omega_i$, and chooses its action $a_{i,k} \in A_i$ to maximize a shared reward. A joint action from all agents is denoted by $a_k = (a_{1,k}, a_{2,k}, \ldots, a_{N,k})$, and a system state transitions from $s_k \in S$ to $s_{k+1} \in S$ following state transition function $\mathcal{P}(s_{k+1}|s_k, a_k)$. Observation function $\mathcal{O}(o_{k+1}|a_k, s_{k+1})$ defines the probability of observing joint observation $o_{k+1} = (o_{1,k+1}, o_{2,k+1}, \ldots, o_{N,k+1})$ when the state is s_{k+1} by joint action a_k. Because agents share a common goal in a dec-POMDP setting, they receive the same reward; reward function $\mathcal{R}(s_k, a_k)$ defines the reward earned by the agents when joint action a_k is taken at s_k.

A policy for each agent, π_i, is a function that probabilistically defines the agent's individual action, given the information available to the agent. In dec-POMDP, an agent neither knows the

[1] To be precise, MARL algorithms discussed in this paper are deep-MARL, which adopts DNN to approximate policy function and value function.

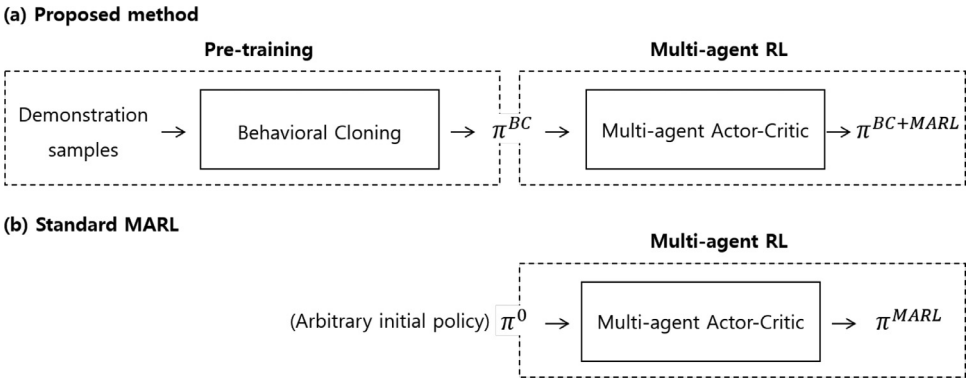

Fig. 1. Proposed solution method.

full state of the system nor other agents' actions at each decision epoch. Thus, each agent chooses its action based on its own previous actions and observations, i.e., action-observation history (AOH). Specifically, $\pi_i(a_{i,k}|\bar{\theta}_{i,k})$ is a probability of choosing an individual action $a_{i,k}$, given individual AOH. We denote individual AOH by $\bar{\theta}_{i,k} = (a_{i,1}, o_{i,2}, \ldots, a_{i,k-1}, o_{i,k})$ and joint AOH by $\bar{\theta}_k = (\bar{\theta}_{1,k}, ..., \bar{\theta}_{N,k})$. A jointly separable policy is the product of the individual policies:

$$\pi(a_k|\bar{\theta}_k) = \prod_{i=1}^{N} \pi_i(a_{i,k}|\bar{\theta}_{i,k}). \tag{1}$$

Then, the objective of a dec-POMDP model is to find a jointly separable policy π that maximizes its expected reward[2]:

$$\pi^* = argmax_\pi E_\pi[\sum_{k=1}^{K} \mathcal{R}(s_k, a_k)] \tag{2}$$

Since it is often impractical to exactly solve Eq. (2), MARL based on deep neural network is used as a practical alternative to obtain an approximate solution. In such MARL algorithms, a dec-POMDP policy is represented as a neural network; specifically, Recurrent Neural Network (RNN) is commonly used (Foerster et al., 2016; Foerster et al., 2018; Lowe et al., 2017; Omidshafiei et al., 2017). RNN is an effective architecture to consider temporal dependency between inputs and outputs, and, as such, is appropriate for a POMDP environment where an observation sequence, i.e., AOH, is used as an input. This policy network is specifically denoted as π_w, where subscript w represents the network parameters: $\pi_w(a_k|\bar{\theta}_k)$. As in Eq. (3), this can be written as

$$\pi_w(a_k|\bar{\theta}_k) = \prod_{i=1}^{N} \pi_{i;w}(a_{i,k}|\bar{\theta}_{i,k}). \tag{3}$$

More details on the RNN architecture and its implementation are described in the supplementary material.

3.2. Framework of proposed solution method

3.2.1. Overall concept of the proposed solution method

We propose a solution method for a dec-POMDP problem. Our solution method combines supervised learning with an MARL algorithm. In particular, we augment a widely-used MARL algorithm – multi-agent Actor-Critic (AC)– by behavioral cloning (BC). The motivation of this solution method is to leverage domain knowledge or expert's decision rules when computing a policy solution by an MARL algorithm. We construct a dec-POMDP policy by learning from reference demonstrations (i.e., behavioral cloning), and provide it for the MARL algorithm to use as its initial policy. Later, we show with our numerical examples that the use of the initial policy from BC enhances a standard MARL algorithm in terms of its solution quality and computational cost.

Fig. 1 shows the overall concept of the proposed solution method. In a standard MARL, shown in Fig. 1(b), we start an MARL algorithm, such as multi-agent AC, with an arbitrary initial policy, π^0. Then, the MARL algorithm improves the initial policy to obtain a final policy solution, for which we use π^{MARL} to denote.[3] On the other hand, in the proposed solution method, rather than using an arbitrarily generated initial policy, we construct an initial policy through pretraining it with demonstration samples as a reference. For the pretraining step, we use behavioral cloning which imitates decisions demonstrated in the samples to construct a cloned policy, π^{BC}. Then, a multi-agent AC algorithm takes π^{BC} as its initial policy and improves it through iterations. The final policy solution after the MARL step with multi-agent AC is denoted as $\pi^{BC+MARL}$. Note that we can think of demonstration samples as specific instances of decision making by an underlying policy, such as human expert knowledge or heuristic decision rules. We refer to this underlying policy as a reference policy, π^{ref}. With that notion, our proposed solution method can be summarized as follows: construct π^{BC} from π^{ref} by BC and then uses π^{BC} as an input for the multi-agent AC algorithm to obtain $\pi^{BC+MARL}$.

Using numerical experiments, we will show that pretraining by using BC before the MARL step provides advantages over the standard MARL (i.e., without pretraining). Firstly, the use of π^{BC} reduces the computation time to reach the convergence of a policy solution. We conjecture this reduction is attributed to a more efficient exploration in the multi-agent AC algorithm, especially during the early phase of learning. By starting with π^{BC} rather than an arbitrary initial policy π^0, the amount of improvement that needs to be achieved through MARL is smaller; this is because the performance of π^{BC} if properly constructed, is closer to the optimal dec-POMDP policy than π^0 is. In addition, π^{BC} also seems to allow the AC algorithm to focus its exploration in a more relevant policy solution space. In a multi-agent environment, it is important for agents to learn and develop a policy solution in which their actions are coordinated to achieve the system objective. When starting from an arbitrary policy π^0, it can take many iterations to reach a reasonably-coordinated policy. On the other hand, π^{BC} is a result of imitating coordinated behaviors, demonstrated in the reference samples, and thus we expect that the exploration would be more effective than starting from π^0.

[2] To be precise, the expectation should be written as $E_{s_k,a_k\sim\pi}[\cdot]$. For the sake of simplicity, we simply write it as E_π and omit the random variables.

[3] Throughout this paper, we use π^{MARL} as a simplified notation for $\pi_{w^{MARL}} = \prod_{i=1}^{N} \pi_{i;w^{MARL}}$, where i is an index for an agent and w^{MARL} is the network parameter obtained after training by the MARL algorithm. We use the same notation convention for π^{BC} and $\pi^{BC+MARL}$.

The proposed solution method is explained in detail in the following sections. In Section 3.2.2, we describe how to obtain π^{BC} when we have demonstrations to imitate. Then, Section 3.2.3 provides details on the design and implementation of a multi-agent AC algorithm.

3.2.2. Pretraining: using BC to obtain an initial dec-POMDP policy

Behavioral cloning (BC) refers to a supervised learning method, where agents learn their policy from demonstration samples. Its objective is to construct a policy π^{BC} that best imitates the decision making by the reference policy π^{ref}. Note that π^{ref} is only notional, and it exists as demonstration samples. BC generates π^{BC}, a feasible dec-POMDP policy, from demonstration samples which encode π^{ref}.

The main input to BC is a set of sample demonstrations for all agents. Let $\bar{\theta}_{i,k}$ denote a single sample of AOH for agent i at time k: $\bar{\theta}_{i,k} = (a_{i,1}, o_{i,2}, \ldots, a_{i,k-1}, o_{i,k})$. In the demonstration samples, for each $\bar{\theta}_{i,k}$, we know $a_{i,k}$, the action chosen by agent i. In the supervised learning (BC) concept, $a_{i,k}$ is a true label for each $\bar{\theta}_{i,k}$. Then, we use $\bar{\theta}_K$ to denote a collection of $\bar{\theta}_{i,k}$ for all i and k. Likewise, we let $\bar{a}_K$ denote the history of observed actions for all agents throughout K: $\bar{a}_K = ((a_{1,1}, a_{1,2}, \ldots, a_{1,K}), \ldots, (a_{N,1}, \ldots, a_{N,K}))$. We define a sample demonstration τ to consist of state sequence, AOH, and reward sequence: $\tau = (\bar{s}_K, \bar{\theta}_K, \bar{a}_K, \bar{r}_K)$. s_k denotes a sampled state, and $\bar{s}_K = (s_1, s_2, \ldots, s_K)$. $\bar{r}_K$ is a sequence of sampled reward r_k, and $\bar{r}_K = (r_1, r_2, \ldots, r_K)$.

BC aims to reduce deviations between the chosen action $a_{i,k}$ and the predicted action from the current individual policy $\pi_{i;w}$. It updates network parameter w such that $\pi_{i;w}(a_{i,k}|\bar{\theta}_{i,k})$ is 1 for as many input $\bar{\theta}_{i,k}$ in demonstrations as possible. A loss function $L(w)$ to update w is

$$L(w) = -\sum_{\tau \in B^D} \sum_{k=1}^{K} u_k^{\tau} \sum_{i=1}^{N} \log \pi_{i;w}(a_{i,k}|\bar{\theta}_{i,k}) \tag{4}$$

where B^D denotes a demonstration buffer, which is a set of all demonstration samples. Recall that $\pi_{i;w}(a_{i,k}|\bar{\theta}_{i,k})$ is the probability of choosing $a_{i,k}$ given $\bar{\theta}_{i,k}$ under the current policy represented by a policy network with parameter w. u_k^{τ} is a positive weight for a sample trajectory τ at time k. u_k^{τ} is introduced to assign a higher weight to the predictions at earlier time steps. $L(w)$ quantifies the degree to which the actions predicted by a current policy match the actions in demonstration samples. $L(w)$ defines the negative cross-entropy loss with weights on each prediction. Cross-entropy measures the difference between two probability distributions. Negative cross-entropy loss uses cross-entropy to evaluate how well the prediction function is correctly matching the true label. $L(w)$ is 0 if the classifier predicts a true label with a probability 1; that is, if $\pi_{i;w}(a_{i,k}|\bar{\theta}_{i,k}) = 1\ \forall i$ and k, then $L(w) = 0$. Conversely, if the classifier does not match the true label at all, $L(w)$ diverges to positive infinity.

A gradient descent method is used to find w that minimizes loss function $L(w)$. It finds a local minimum of an objective function through iterations using gradient information. As specified in Eq. (4), we use a full batch of demonstration samples (B^D) at every iteration. In this work, we use Adam (Adaptive moment estimation) method (Kingma & Ba, 2015), which updates parameters by using the gradient values in previous iterations.

Algorithm 1 summarizes the BC algorithm used in our study. The output of Algorithm 1 is a decentralized policy, which we denote as π^{BC}. This policy is used as in initial policy in the subsequent MARL step.

Finally, we wish to comment on the quality of π^{BC}. Because BC's goal is to imitate a reference policy as much as possible, the maximum quality we can expect from π^{BC} is evidently limited by that of π^{ref}. On the other hand, a lower bound of π^{BC}'s performance is related to the prediction error in BC. We construct a theorem based on Ross and Bagnell (2010) to formally state the lower bound of π^{BC}'s performance:

Algorithm 1 Behavioral cloning (Glorot & Bengio, 2010).

1: Initialize w by Xavier uniform initialization method~(Glorot & Bengio, 2010)
2: Set weight $u_k^{\tau}\ \forall k \in \{1, \ldots, K\}, \tau \in B^D$ and learning rate α
3: **for** supervised learning steps **do**
4: Compute $L(w)$ using Eq. (4)
$L(w) = -\sum_{\tau \in B^D} \sum_{k=1}^{K} u_k^{\tau} \sum_{i=1}^{N} \log \pi_{i;w}(a_{i,k}|\bar{\theta}_{i,k})$
5: Update w
$w \leftarrow w - \alpha \nabla_w L(w)$
6: **end for**

Theorem 1. *Define ϵ_k, prediction error of π^{BC} at decision epoch k, as $\epsilon_k = E_{\bar{\theta}_k, a_k \sim \pi^{ref}}[E_{\hat{a}_k \sim \pi^{BC}}[I(a_k \neq \hat{a}_k)]]$. Let $J(\pi)$ denote the expected total reward of policy π. Then $J(\pi^{BC}) \geq J(\pi^{ref}) - (r_{max} - r_{min})K^2\epsilon$, where $\epsilon = \frac{1}{K}\sum_{k=1}^{K} \epsilon_k$.*

Proof of Theorem 1 is provided in the supplementary material. In the above statement, $(r_{max} - r_{min})$ is the gap between the maximum and minimum of the immediate reward. Theorem 1 states that the lower bound of π^{BC} is positively proportional to the quality of π^{ref} and negatively proportional to the average prediction error ϵ. It implies that it is important to choose π^{ref} that is both high-quality and easy to learn in imitation learning. To this end, domain-specific knowledge and experience can help improve the quality of π^{BC}.

3.2.3. MARL: a multi-agent Actor-Critic algorithm

After obtaining π^{BC}, now we further improve it through MARL. We employ a multi-agent Actor-Critic (AC) as our MARL algorithm. A multi-agent AC algorithm is one of the latest and widely used MARL algorithms. It uses both a policy function and a value function to find a policy solution (Konda & Tsitsiklis, 2000). Recall that a value function $V^{\pi_w}(s_k)$ is the expected return at the current state s_k when an action is chosen according to policy π_w: $V^{\pi_w}(s_k) = E_{\pi_w}[\sum_{l=0}^{K-k} \mathcal{R}(s_{k+l}, a_{k+l})|s_k]$. As with a policy function, a value function is approximated as a neural network with network parameters v: $V^{\pi_w}(s_k) \approx V_v^{\pi_w}(s_k)$.

An AC algorithm uses a value function when computing a policy gradient to update a policy function. When an "actor" chooses an action under a current policy π_w, a "critic" updates parameters v of the value network by incorporating rewards realized by the chosen action. Then, the critic uses the updated value function to guide the actor in updating parameter w for its policy function. For its multi-agent implementation, we use a centralized critic, which uses the information on the full states to update its value network; it follows that we have a single value network that acts as a sole critic for all agents, hence termed as a centralized critic. It is structured as a feed-forward network that takes the current state s as its input and predicts the value $V_v^{\pi_w}(s)$.

Policy parameter w is updated to maximize the expected total return, $J(w) = E_{\pi_w}[r_1 + r_2 + \cdots + r_K]$, by moving in the direction of policy gradient $\nabla_w J(w)$. Specifically, we calculate the policy gradient based on generalized advantage estimator (GAE) (Schulman, Moritz, Levine, Jordan, & Abbeel, 2016):

$$\begin{aligned} \nabla_w J(w) &= E_{\pi_w}[Q^{\pi_w}(s_k, a_k) \nabla_w \log \pi_w(a_k|\bar{\theta}_k)] \\ &\sim E_{\pi_w}[A_k^{GAE} \nabla_w \log \pi_w(a_k|\bar{\theta}_k)]. \end{aligned} \tag{5}$$

Note in Eq. (5) that A_k^{GAE} acts as an estimate of q-function $Q^{\pi_w}(s_k, a_k)$[4]. Specifically, GAE uses a control variate method

[4] $Q^{\pi_w}(s_k, a_k) = E_{\pi_w}[\sum_{l=0}^{K-k} \mathcal{R}(s_{k+l}, a_{k+l})|s_k, a_k]$.

with exponential weights to get an estimator that substitutes $Q^{\pi_w}(s_k, a_k)$:

$$A_k^{GAE} = \sum_{l=0}^{K-k} (\lambda^{GAE})^l \{r_{k+l} + V_v^{\pi_w}(s_{k+l+1}) - V_v^{\pi_w}(s_{k+l})\}. \quad (6)$$

It balances the variance and bias of the estimator by using $\lambda^{GAE} \in [0, 1]$.

Parameter v for a value function is updated to reduce the difference between the current estimate of the value function and what is computed by sampled rewards from simulations. For updating v, TD(λ) method is used. TD(λ) method is a value estimation method that adjusts how much of the sample rewards are used for defining the target, G_k^λ. As in GAE, $\lambda \in [0, 1]$ represents the degree of compromise between the variance and bias of the estimate. In TD(λ) method, the following loss function is used:

$$L(v) = \sum_{k=1}^{K} (G_k^\lambda - V_v^{\pi_w}(s_k))^2 \quad (7)$$

where G_k^λ is a weighted sum of n-step return $G_k^{(n)}$.[5] TD(0) minimally uses the sampled reward by setting $G_k^{(0)}$ as $r_k + V_v^{\pi_w}(s_{k+1})$. For TD(1), it becomes $\sum_{l=0}^{K-k} r_{k+l}$, which is the cumulative return.

Finally, we briefly describe the initialization of a policy network and value network. The proposed solution method initializes a policy network and a value network based on the results from the BC step. For a policy network, the network parameter w^{BC} of π^{BC} is directly used to define the initial policy network for the AC algorithm. For a value network, we need to construct an initial value network because, during the BC step, only a policy network is trained. We estimate a value function from sample rewards by simulating π^{BC}. That is, we obtain parameter v that minimizes Eq. (7) in which π_w is replaced with π^{BC}.

Algorithm 2 summarizes the MARL algorithm used in our study.

Algorithm 2 Multi-agent AC with a centralized critic.

Initialize v and w. Use pretrained v and w if available.
Set λ, λ^{GAE} and learning rate β and γ.
for MARL steps **do**
  Empty training buffer B and set $\Delta v = 0$
  for Batch size **do** //~sample an episode by running π_w
    s_1 = initial state, $k = 1$, $\bar{\theta}_{i,1} = \emptyset$ $\forall i \in \{1, \ldots, N\}$
    while $s_k \neq$ terminal **do**
      Get observation o_k
      $\bar{\theta}_{i,k} \leftarrow (\bar{\theta}_{i,k-1}, a_{i,k-1}, o_{i,k})$
      $a_k \sim \prod_i \pi_{i;w}(a_{i,k}|\bar{\theta}_{i,k})$
      Get reward r_k and next state s_{k+1}
      $k = k + 1$
    end while
    Store episode $\tau = (\bar{s}_K, \bar{\theta}_K, \bar{a}_K, \bar{r}_K)$ to B
    Calculate TD(λ) target G_k^λ using v
    $\Delta v = \Delta v + \sum_{k=1}^{K} \nabla_v (G_k^\lambda - V_v^{\pi_w}(s_k))^2$
  end for
  Update parameter v for value network
  $v \leftarrow v - \beta \Delta v$
  Update parameter w for policy network
  $\Delta w = \sum_{\tau \in B} \sum_{k=1}^{K} [A_k^{GAE} \nabla_w \log \prod_i \pi_{i;w}(a_{i,k}|\bar{\theta}_{i,k})]$
  $w \leftarrow w + \gamma \Delta w$
end for

We use a stochastic descent and ascent method to compute parameter v and w. At each training, it stochastically samples a batch of episodes drawn from simulation under current policy π_w. These sampled episodes are used to compute the loss function Eq. (7) and the policy gradient defined in Eq. (5). A value network is updated to minimize the summation of loss value for each episode in a batch, and a policy network is improved by moving parameter w in the direction of the policy gradient. As with our BC implementation, we use the Adam method for updating both networks to find optimal w and v by gradient-based local optimization.

4. Application of the proposed solution method: Decentralized selective patient admission problem

4.1. Problem description

In the aftermath of a multiple casualty incident (MCI), a large number of patients with different severity arrive at emergency departments (ED's) in a short period of time. When low-severity patients arrive early at EDs and occupy their resources first (der Heide, 2006; Hogan, Waeckerle, Dire, & Lillibridge, 1999; Waeckerle, 1991), care provision to high-severity patients may be delayed due to resource shortage. Selective patient admission refers to a strategy to divert some patients to other ED to preserve medical resources for future arrivals of patients in more critical condition. The goal of a selective patient admission strategy is to maximize the overall number of survivors from an MCI.

Fig. 2 describes our decision problem. There are N^p patients at an MCI site, and they belong to one of the two severity classes: immediate and delayed. Under widely used triage classification systems such as START, both immediate- and delayed-class patients require definitive care (e.g., surgery) for their survival. Immediate-class (red) patients are in a highly critical condition, and they need definitive care to be provided immediately. Delayed-class (yellow) patients are also in a critical condition, but the urgency of definitive care provision is relatively less than immediate-class patients. The survival probability of these patients decreases in time, and the rate of decrease depends on their severity class and the elapsed time from an incident. Patients arrive at each ED according to a time-dependent arrival rate,[6] and upon their arrival, an ED determines whether to admit the currently arriving patient. If the decision is to admit, a bed in the ED (or equivalent resource) is consumed. If it decides to divert, the patient is transferred to another ED.

This problem is cast as a decentralized decision problem under partial-observation. We assume that each ED only recognizes the current state of itself and makes its admission decision without having access to information on other EDs (e.g., available beds) or on the patients (e.g., number of patients being diverted). Lack of such information brings a significant challenge for decision making. For example, an ED would divert a patient only if it knows that at least one ED has a bed to immediately admit the patient, but without knowing it, the ED has to make a diversion decision under the uncertainty of prompt care provision at a destination ED.

4.2. dec-POMDP formulation

We briefly introduce key elements of our dec-POMDP model.

- Decision epoch, k: A decision epoch is the moment of patient arrival at any of the EDs in the system. The decision problem is terminated when there is no remaining bed or all patients

[5] $G_k^\lambda = (1-\lambda)\sum_{n=1}^{K-k-1} \lambda^{n-1} G_k^{(n)} + \lambda^{K-k-1} \sum_{l=0}^{K-k} r_{k+l}$ $G_k^{(n)} = \sum_{l=0}^{n-1} r_{k+l} + V_v^{\pi_w}(s_{k+n})$

[6] We assume patient arrivals – the time of arrival (t), severity class (m), and destination ED (i^*) – follow a multinomial distribution: $(t, m, i^*) \sim Multinomial(N^p; p_{t,m,i^*})$. A multinomial distribution is used to ensure a fixed number of patients arrive from an MCI.

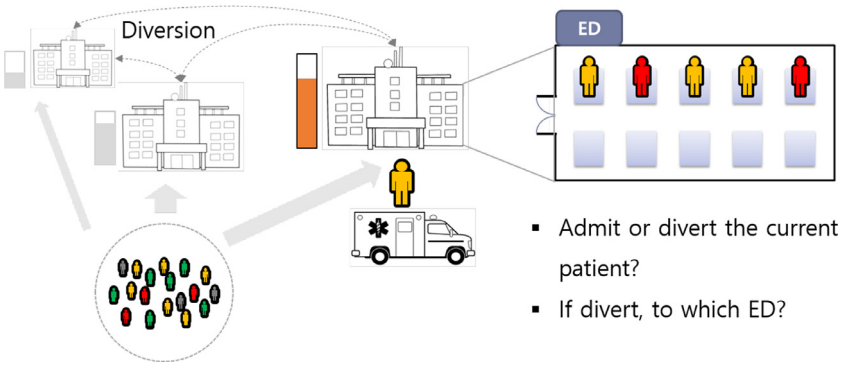

Fig. 2. Illustration of a decentralized selective patient admission problem.

Table 1
Meaning of each state element.

Element	Definition
t	Discretized time since the onset of MCI, $t = \{0, \Delta, 2\Delta, ...\}$ where Δ is a discretized time step length
m	Severity-class of the currently arriving patient, $m \in \{immediate, delayed\}$
i^*	Index of ED at which the current patient arrives, $i^* \in \{1, \ldots, N\}$
y^{sc}	Number of patients who have arrived from MCI site during $[t, t+\Delta)$, $y^{sc} \in \{0, \ldots, N^p\}$
y^{z}	Number of patients who have arrived via transfer during $[t, t+\Delta)$, $y^{z} \in \{0, \ldots, N^p\}$
b_i	Number of beds remaining at ED i, $b_i \in \{0, \ldots, B_i\}$, where B_i is the initial number of beds available at ED i.
$z^Y_{l,i}$	Number of transfer patients of delayed-class to arrive at ED i during $[t + l * \Delta, t + (l+1) * \Delta)$, $z^Y_{l,i} \in \{0, \ldots, N^p\}$
$z^R_{l,i}$	Number of transfer patients of immediate-class to arrive at ED i during $[t + l * \Delta, t + (l+1) * \Delta)$, $z^R_{l,i} \in \{0, \ldots, N^p\}$

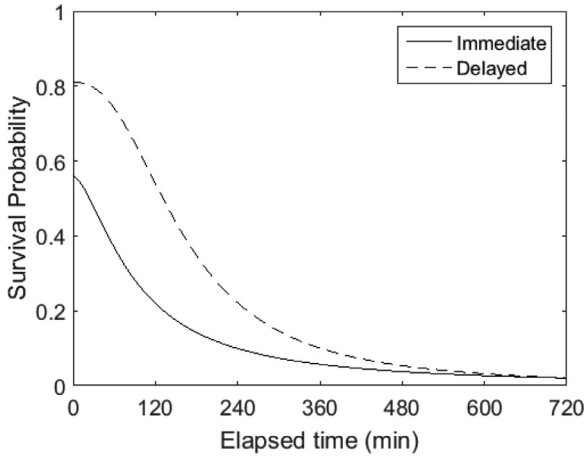

Fig. 3. Survival probability for immediate- and delayed-class patients (Mills et al., 2013).

are admitted. Note that the system may never reach the above terminal states when there is at least one patient who keeps being diverted by the EDs. When that happens, we consider the system has reached a terminal state[7].

- System state, s_k: At each decision epoch k, state vector s_k is defined as

$$s_k = (t, m, i^*, y^{sc}, y^z, b_1, \ldots, b_N, z^Y_{0,1}, \ldots, z^Y_{0,N}, \ldots, z^Y_{L,1}, \ldots, z^Y_{L,N}, z^R_{0,1}, \ldots, z^R_{0,N}, \ldots, z^R_{L,1}, \ldots, z^R_{L,N}).$$

  N is the number of EDs in the system, and L is the travel time between the two farthest EDs. Table 1 explains each of the elements in s_k. It contains system state information in four dimensions – attributes of the currently arriving patient (t, m, i^*), the number of patients arriving within the current time step (y^{sc}, y^z), the number of remaining beds ($b_1, \ldots, b_N$), and the status of patients on diversion ($z^Y_{0,1}, \ldots, z^R_{L,N}$).
- Observation, o_k: Each ED makes a partial observation on the system state. When a patient arrives at ED i^*, ED i^* gains information, $o_{i^*,k}$, on the arriving patient and its own resource state. Note all other EDs do not obtain any information at the moment. Specifically, $o_{i^*,k} = (t, m, i^*, b_{i^*})$, $o_{i,k} = \emptyset$ for $i \neq i^*$, $o_k = (o_{1,k}, \ldots, o_{N,k})$
- Action, a_k: An ED decides whether to admit a currently arriving patient and, if diverting, to which ED it diverts the patient. That is, upon an arrival of a patient, an ED's action is to select one of the EDs: $a_{i,k} \in \{1, \ldots, N\}$, $a_k = (a_{1,k}, \ldots, a_{N,k})$.

[7] In our implementation, we set an upper bound for the maximum number of decision epochs to be a sufficiently large number, M.

- Reward, $\mathcal{R}$: A positive reward is earned when a patient is admitted to an ED – $\mathcal{R}(s_k, a_k) = f(m, t)$ if $a_{i^*,k} = i^*$. $f(m, t)$ denotes a survival probability function for a patient with severity-class m. If a patient is diverted to another ED, no immediate reward is given. That is, $\mathcal{R}(s_k, a_k) = 0$ if $a_{i^*,k} \neq i^*$. For $f(m, t)$, we use the survival probability function in Mills et al. (2013), which is shown in Fig. 3.
- State transition: State transition depends on a chosen action. If an ED's chosen action is to admit a patient, the number of remaining beds in the ED is reduced by one. If the action is to divert a patient, patient diversion status changes. State transition also happens when a patient arrives at an ED either from an MCI site or from another ED (diversion). Details on the state transition probabilities are provided as supplementary material.

4.3. Demonstration samples for BC from oracle solutions

As described in Section 3.2.2, BC takes demonstration samples $\tau = (\bar{s}_K, \bar{\theta}_K, \bar{a}_K, \bar{r}_K)$ as an input to construct π^{BC}. τ may be obtained from human experts' demonstrations, decision rules, or any other relevant source of demonstrations. In this problem, we use solutions from a job assignment integer programming (IP) model as a source of demonstration data. Since the solution of this IP model is obtained with complete knowledge of all future arrivals of patients, we call this IP model as P^{oracle} and a solution of P^{oracle} as an oracle solution.

The following IP model, P^{oracle}, defines the job assignment problem:

$$(P^{oracle}) \quad \max \sum_{i=1}^{N} \sum_{j=1}^{N^p} x_{ij} f_{ij} \tag{8}$$

$$s.t. \sum_{i=1}^{N} x_{ij} \leq 1, \quad \forall j \in \{1, \ldots, N^p\} \tag{9}$$

$$\sum_{j=1}^{N^p} x_{ij} \leq B_i, \quad \forall i \in \{1, \ldots, N\} \tag{10}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in \{1, \ldots, N\}, \ j \in \{1, \ldots, N^p\} \tag{11}$$

Decision variables, x_{ij}, indicate an admission decision for patient j to ED i – $x_{ij} = 1$ if admitted and 0 otherwise. N is the number of EDs and N^p is the number of patients. In objective function (8), f_{ij} is the survival probability of patient j when admitted to ED i, and (8) maximizes the total survival probability. Constraints (9) require that a patient can be admitted to at most one ED; constraints (10) are capacity constraints for each ED to limit the number of admissions to B_i for ED i.

We generate a large number of instances of patient arrivals by sampling from the assumed patient arrival distribution. In each problem instance, all future patient arrivals are specified – time and severity class of each arrival and its destination ED. Note that, for a given problem instance, we can compute f_{ij}. Then, we can solve P^{oracle} to obtain a patient-ED assignment solution, x_{ij}. This patient-ED assignment solution encodes admission decisions at individual EDs.

With an admission decision solution from P^{oracle} for each instance of patient arrivals, we run the simulation following the admission decision. In the simulation, when a patient arrives at an ED, we use the admission decision solution to either admit the patient to the current ED or divert to the ED specified in the solution. As the simulation progresses, we record state sequence, reward sequence, and action-observation history for each ED. Thus, for each patient arrival instance, we obtain one demonstration sample, $\tau = (\bar{s}_K, \bar{\theta}_K, \bar{a}_K, \bar{r}_K)$. We obtain τ for a large number of patient arrival instances, and these τ's are used as a demonstration set in BC.

Note that the oracle admission decisions would be different from admission decisions of an optimal dec-POMDP policy even when the system experiences the same patient arrivals. This is because when computing the oracle decision solution, all future patient arrivals are fixed and known a priori. This hypothetical setting for P^{oracle} is obviously non-existent, and its solution exists only in a notional sense. Nevertheless, since oracle admission decisions are obtained with much more information than what can be obtained in a dec-POMDP environment, we use oracle admission decisions as a source of demonstration samples for BC with a belief that they contain valuable knowledge for obtaining a good dec-POMDP policy.

5. Computational experiments

5.1. Experimental setting

We apply the proposed solution method to three MCI scenarios with varying scales. The total number of patients in each scenario is 12, 30, and 65, and the number of responding EDs is 3, 5, and 11, respectively. The first scenario is based on the actual incident data from a bus crash on Incheon Bridge in South Korea (Kang et al., 2013), where three immediate-class patients and nine delayed-class patients were transported to three EDs – ED A, ED B, and ED C. For this scenario, we assume the initial number of available beds at each ED, B_i, to be proportional to the size of the EDs: $B_A = 4, B_B = 6, B_C = 2$. We estimate an arrival density, $Multinomial(N^p; p_{t,m,i^*})$, from the actual patient arrival record at each hospital, as shown in Table 2. Travel times between the three EDs are obtained from their actual locations: 0.5 hr for all ED pairs.

Table 2
p_{t,m,i^*} for Scenario 1.

Time [hr]	Delayed-class			Immediate-class		
	ED A	ED B	ED C	ED A	ED B	ED C
[0.0, 0.5)	0	0	0	0	0	0
[0.5, 1.0)	0.250	0	0	0	0	0
[1.0, 1.5)	0.333	0.083	0.083	0.083	0	0
[1.5, 2.0)	0	0	0	0.083	0.083	0

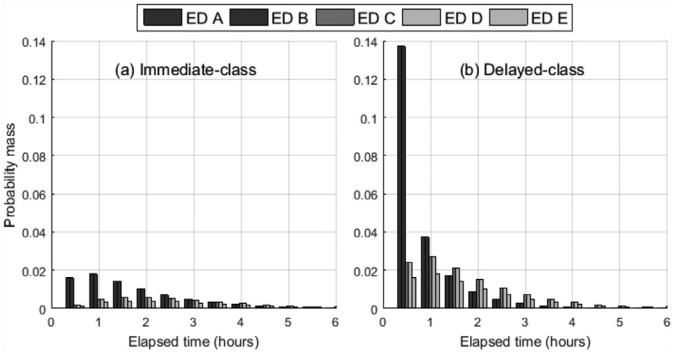

Fig. 4. p_{t,m,i^*} for Scenario 2.

In the second scenario, we consider a resource-constrained situation after a large-scale MCI. There are 30 patients with a 1:3 ratio between immediate-class and delayed-class patients. These patients are transported to five nearby EDs with the following initial capacity: $B_A = 3, B_B = 3, B_C = 12, B_D = 12, B_E = 15$. Patient arrival density is constructed such that arrivals of patients reach a peak at the beginning and gradually decreases.[8] Fig. 4 shows p_{t,m,i^*} for Scenario 2. This scenario depicts a situation where first responders send a large number of patients to ED A and B which have only a few available beds. Transfer time between ED B and other EDs is 0.5 hrs while between all other EDs, it is 1.0 h.

The third scenario is based on the Mauna Ocean Resort gymnasium collapse (Cha et al., 2017), one of largest MCIs in recent years in South Korea. In this scenario, we consider 65 severely-injured patients with a 1:4 ratio between immediate-class and delayed-class patients. Patients are transported to eleven EDs, distributed in three nearby cities – four EDs, six EDs, and one ED in each city. Initial capacity of each EDs is $B_A = 5, B_B = 6, B_C = 14, B_D = 6, B_E = 15, B_F = 10, B_G = 5, B_H = 5, B_I = 6, B_J = 15, B_K = 5$. We set patient arrival density using the same method in constructing scenario 2 based on peak arrival time and arrival volume data. Distances between EDs and the MCI site are determined based on their actual locations.

In behavioral cloning, we use 1,000 demonstration samples obtained from solving P^{oracle}, and a policy network is trained for 512 iteration steps. We use a sampled return, $r_k + r_{k+1} + \cdots + r_K$, as positive weights u_k^τ. A value network is pretrained for 64 steps using 128 episodes per each step generated by π^{BC}. After the pretraining, the AC algorithm learns a dec-POMDP policy for 5000 steps. At each MARL training step, we also use 128 episodes. In the MARL phase, we set an immediate reward as a reduction in

[8] Details on setting p_{t,m,i^*} are provided in the supplementary material.

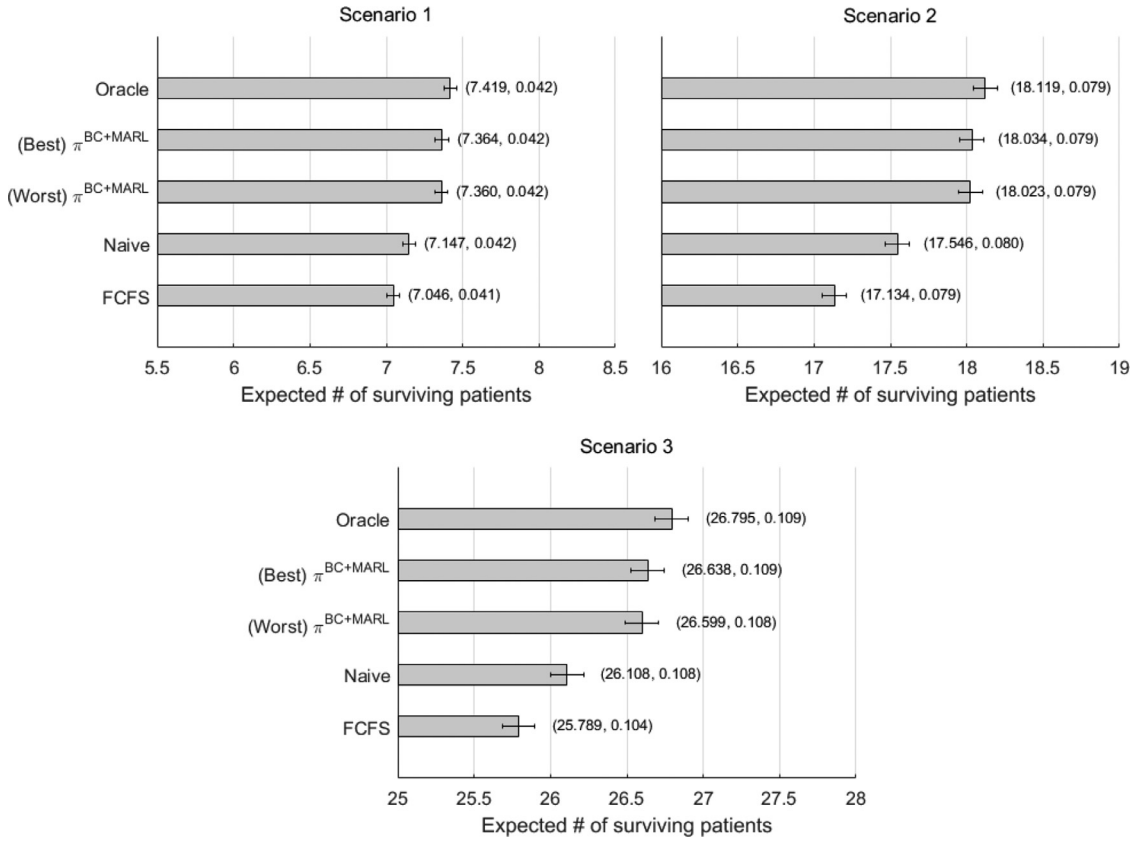

Fig. 5. Performance of a dec-POMDP policy from the proposed method and alternative policy solutions. Bars indicate 95% confidence interval.

survival probability when a patient is diverted. Under these settings, we obtain five policy solutions with interim policy networks saved at every 100 training steps for each policy solution. Other hyperparameters related to the neural network structure and training scheme are summarized in the supplementary material.

To assess the performance of our policy solutions, we compare our policy solutions with other heuristic policies. For comparison, we consider two alternative heuristic policies for selective patient admission: First-come first-serve (FCFS) policy and "naïve" policy. Under the FCFS policy, an ED always admits a patient as long as there is an available bed; if no bed is available, it diverts a patient to the nearest ED. A naïve policy is constructed based on Lee and Lee (2018). Lee and Lee (2018) discusses a selective admission problem for a single ED-setting. Their model assumes when a diversion decision is made, the patient can be immediately admitted at the destination ED, and thus we refer to the policy solution as naïve. In addition to the two heuristic policies, we also present an upper bound of the problem. Solutions to P^{oracle} give an upper bound value of our dec-POMDP problem because it uses a priori knowledge on future arrival information. Thus, as an upper bound, we compute the average of optimal values of P^{oracle} for arrival instances.

5.2. Results and discussions

We compare the quality of policy solutions from the three alternatives: $\pi^{BC+MARL}$, Naïve, and FCFS policy. As a reference, we show the performance of the oracle solution, which is an upper bound of the problem. For each policy solution, we run 1,000 simulations while controlling the random seed in each simulation to ensure they are evaluated under the same condition. Note that different $\pi^{BC+MARL}$ is obtained every time we solve the dec-POMDP problem. For comparison with other policy solutions, we solve the problem to obtain five $\pi^{BC+MARL}$, and we report performance results from the best and worst of the five $\pi^{BC+MARL}$. Fig. 5 shows the results from the simulation experiments.

Table 3
Number of diversions and its types. Numbers in parenthesis represent 95% confidence interval.

		FCFS	Naïve	$\pi^{BC+MARL}$	
				Min	Max
Scenario 1	Total	8.07 (0.19)	8.17 (0.19)	5.83 (0.09)	5.88 (0.09)
	Selective	0.00 (–)	3.43 (0.10)	2.78 (0.07)	2.85 (0.07)
	Redundant	2.93 (0.16)	2.60 (0.15)	0.49 (0.04)	0.54 (0.04)

Fig. 5 shows that $\pi^{BC+MARL}$ yields better results than FCFS and Naïve policies in all scenarios. In scenario 1, the average value of the expected number of surviving patients from (Worst) $\pi^{BC+MARL}$ is 7.360 with 95% confidence interval of 0.042. This is higher than the results from Naïve and FCFS policies, with statistical significance. The same is also true in scenario 2 and scenario 3. We further examine these results, number of diversion instances in particular, to understand why $\pi^{BC+MARL}$ performs better than benchmark policies. In Table 3, we compare the number of diversion decisions in scenario 1 under each of FCFS, Naïve, and our solution. "Total" refers to the total number of diversions under each policy. Among all diversions, some diversion decisions are made when beds are still available at an ED; we refer these types of diversions in Table 3 as "Selective" diversions. Finally, some patients are diverted more than once – from ED A to B to C, etc. –, we identify these diversions as "Redundant". Table 3 shows that the total number of diversions made under $\pi^{BC+MARL}$ is far smaller

Table 4
Expected number of surviving patients from $\pi^{BC+MARL}$ and π^{MARL}. Numbers in parenthesis represent 95% confidence interval.

		$\pi^{BC+MARL}$	π^{MARL}
Scenario 1	Best	7.364 (0.042)	7.344 (0.041)
	Worst	7.360 (0.042)	7.334 (0.042)
Scenario 2	Best	18.034 (0.079)	17.919 (0.079)
	Worst	18.023 (0.079)	17.910 (0.078)
Scenario 3	Best	26.638 (0.109)	25.845 (0.107)
	Worst	26.599 (0.108)	20.714 (0.084)

Table 5
Comparison of computation time (minutes).

		T^{pre}	T^{MARL}
Scenario 1	Min	4.01	40.4–43.5*
	Max	4.19	65.8–69.0*
Scenario 2	Min	10.54	N/A†
	Max	11.20	
Scenario 3	Min	35.00	N/A†
	Max	36.72	

* T^{MARL} is given as a range because we evaluate interim policy at every 100th training step.
† π^{MARL} did not reach the value of π^{BC}.

than the other policies. In addition, $\pi^{BC+MARL}$ yields the fewest redundant diversions. On the other hand, FCFS yields the highest number of redundant diversions, while not making any selective diversions. Based on these results, we see that Naïve and $\pi^{BC+MARL}$ improves the FCFS policy by selectively admitting patients. This prevents early consumption of resources by less critical patients. Note that redundant diversion significantly delays care provision and thus is detrimental to patient outcomes. $\pi^{BC+MARL}$ avoids such redundant diversions; $\pi^{BC+MARL}$ divert a patient to an ED likely to immediately provide care to the diverted patient, minimizing the chance of additional diversions.

These results suggest that for an ED to make an optimal decision, it is indeed important to consider the other EDs participating in the response efforts; goodness of an individual ED's selective admission decision depends on the states or decision policies of other EDs. In both FCFS and Naïve heuristic policies, these factors are not considered, and these heuristic policies may end up making improper decisions to admit or to divert a patient to a wrong destination. On the other hand, $\pi^{BC+MARL}$, as a solution to the dec-POMDP problem, takes into account possible influences from the other EDs in the system. In the MARL approach, a policy is continually improved by evaluating decisions in previous experiences, and when the quality of a decision depends on other EDs' decisions, the MARL algorithm improves the policy based on these evaluations.

In addition to outperforming the two heuristic policies, $\pi^{BC+MARL}$ performs very close to the oracle solution. Recall that the oracle solution is the upper bound for this problem as it uses complete information on all future arrivals. The average gaps in the objective function between (Worst) $\pi^{BC+MARL}$ and the oracle solution are less than 1% in all scenarios; differences in the expected number of surviving patients from the two are statistically insignificant. Since the oracle solution gives an upper bound value for the dec-POMDP problem, it can be said that the performance of $\pi^{BC+MARL}$ is quite close to the optimal dec-POMDP policy.

Next, we examine the effect of augmenting the MARL algorithm by pretraining via BC. We compare the proposed solution method – pretraining via BC and improving π^{BC} through MARL – with MARL without pretraining. We use π^{MARL} to refer to a policy solution from MARL without pretraining. To obtain π^{MARL}, we start the same actor-critic algorithm without the BC step and learn a policy solution for 10,000 steps. Table 4 summarizes the performance of $\pi^{BC+MARL}$ and π^{MARL}. In Table 4, we observe that $\pi^{BC+MARL}$ gives a better result, though marginal in some scenarios, compared with π^{MARL}.

It is the computation time that pretraining with BC seems to provide a substantial advantage in these examples. To examine the effects of pretraining on computation time, we define pretraining time, T^{pre} in the proposed BC+MARL method. It includes time to solve P^{oracle}, generate demonstration τ, conduct supervised learning (BC), and initialize a value network. For MARL without pretraining, we define a timestamp, T^{MARL}, to be the time when the performance of an interim policy solution reaches that of π^{BC}. That is, we measure the time for MARL to develop, from scratch, a policy solution that has a similar quality as π^{BC}, the initial policy for the MARL step in the BC+MARL method. Table 5 presents the time stamps for each method. This result shows that it takes longer for MARL without pretraining to reach a similar performance of π^{BC} than the time to pretrain neural networks in BC+MARL. In fact, π^{MARL} did not reach the value of π^{BC} even at the end of total training steps, which took more than 23 h for scenario 2 and 103 h for scenario 3. Pretraining clearly helps to reduce the computation time by providing a high-quality initial policy for the MARL algorithm to jump-start its learning process. This implies that when we have a fixed computational budget, BC+MARL has an advantage over MARL without pretraining in that we quickly obtain a decent initial policy from BC and use the remaining budget to further improve the initial policy through MARL.

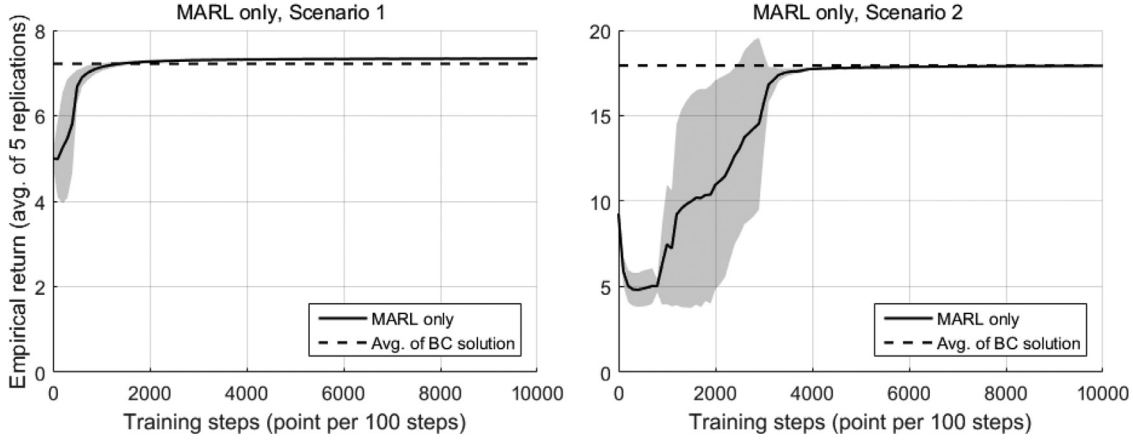

Fig. 6. Learning curve of MARL algorithm without pretraining; Scenario 3 is similar to Scenario 2 and thus not shown here.

Fig. 6 shows learning curves during the training process of the MARL algorithm without pretraining. We draw a learning curve in scenario 1, and scenario 2 by plotting the return values averaged over five different runs, and the shaded area represents their standard deviation. It shows that, without pretraining, the algorithm struggles to improve the policy at the beginning, particularly in scenario 2. This is because the algorithm searches a policy without any prior knowledge about the target problem and many sample experiences are required before reaching a decent solution. It is well known an MARL algorithm can easily fall into a local optimal point during a learning process as it is essentially based on local search. Previous studies attempt to overcome this issue by improving exploration, and the use of demonstrations has been shown to be effective for this purpose in a single-agent environment (Gao et al., 2018; Subramanian et al., 2016). We conjecture that the effectiveness of our proposed solution method is similarly explained in terms of its efficient exploration. Initializing the MARL algorithm from the demonstrations obtained through the solutions of P^{oracle} helps the algorithm to start at a better point in the solution space.

6. Conclusion

In the aftermath of a mass casualty incident, multiple decision-makers are involved in disaster response efforts, and in their decision making, they most likely have an access to only partial information on the current disaster situation. Most of the existing literature on disaster response operations management, however, build their models and analyses on an implicit assumption that decisions are made by a single decision-maker who has complete knowledge of the current situation, i.e., fully-observable single-agent decision problem. In this paper, we present a multi-agent decision making problem where multiple agents seek to achieve a common goal while provided with only partial information on the system states.

We formulate this partially-observable multi-agent problem as a decentralized-POMDP (dec-POMDP) model. While it seems most appropriate to formulate decision making problems in disaster responses as a dec-POMDP model, to our best knowledge, it is the first dec-POMDP study in the disaster response operations research literature. Then, we tackle the computational challenge of solving dec-POMDP models. It is well known that optimally solving a dec-POMDP model is very difficult. In the absence of an efficient exact algorithm, multi-agent reinforcement learning (MARL) has appeared as and will continue to be an attractive solution method for dec-POMDP problems. To that end, we propose a method to augment the standard approach to solve dec-POMDP models. We adopt behavioral cloning (BC) to pretrain initial neural networks of the MARL algorithm. Specifically, we use BC to obtain a policy solution from demonstration samples, which is then used as an initial policy in a multi-agent actor-critic algorithm for MARL.

To examine the effectiveness of the proposed solution method, we solve a decentralized, selective patient admission problem for three disaster scenarios. The results demonstrate that the proposed method is a viable solution approach to solving dec-POMDP problems. Compared with other plausible heuristic policies – FCFS and naïve –, the proposed method obtains a better policy solution; in fact, in the three test scenarios, the performance of the policy solutions from the proposed method is very close (within 1%) to the upper bound value. In addition, augmenting MARL with BC for its pretraining seems to offer advantages over plain MARL. As long as we can identify a decent reference policy, pretraining the neural networks in the MARL helps to improve the quality of the policy solutions and reduce the computation time by providing a high-quality initial policy for the MARL algorithm to jump-start its learning process.

Our experimental results have some practical implications for MCI response operations. First, when an ED admits all arriving patients until it is fully occupied (FCFS policy), there will be many missed opportunities to save more lives. Rather than operating with FCFS policy, EDs may need to leave its bed(s) to serve more critical patients in future. Second, it is important for response teams to train for a situation where access to information on EMS resources is limited. If well prepared for operation under limited information, its decision making on resource utilization can in fact be quite close to the quality of decision making made with complete situational awareness.

Finally, we offer a few potential areas of future work to extend this study. One area of future work is to broaden the scope of the decision problem. While this work focuses on a decision problem at the hospital phase of disaster response operations, we may need to incorporate overall EMS operations to achieve more integrated disaster responses. In fact, we see increasing interests from disaster response operations research literature to address integrated decision making for multiple tasks of the disaster response process (e.g., Repoussis et al., 2016; Wilson et al., 2013). Yet, these studies assume a single (centralized) decision-maker with complete information, and much research is needed to expand to multi-agent partially observable environment where our proposed method can offer a valuable starting point. Another area of future work is to improve its practical applicability in the context of disaster response operations. One may pursue to expedite computation of an acceptable policy solution by carefully assessing the trade-off between the quality of policy solution and its computation time. For example, if a decent reference policy is available, the imitated policy from the BC step may offer acceptable performance within a short computation time. Also, one can use our method to study dec-POMDP policies under various disaster scenarios and investigate to develop practically implementable heuristic policies.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1A2B4014323).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2020.09.018.

References

Amato, C., Dibangoye, J. S., & Zilberstein, S. (2009). Incremental policy generation for finite-horizon Dec-POMDPs. In *ICAPS*.

Argon, N. T., Ziya, S., & Righter, R. (2008). Scheduling impatient jobs in a clearing system with insights on patient triage in mass casualty incidents. *Probability in the Engineering and Informational Sciences, 22*(3), 301–332.

Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research, 27*(4), 819–840.

Cha, M.-i., Kim, G. W., Kim, C. H., Choa, M., Choi, D. H., Kim, I., … Lee, K. H., et al. (2017). A study on the disaster medical response during the mauna ocean resort gymnasium collapse. *Journal of The Korean Society of Emergency Medicine, 28*(1), 97–108.

Chan, C. W., Farias, V. F., Bambos, N., & Escobar, G. J. (2012). Optimizing intensive care unit discharge decisions with patient readmissions. *Operations Research, 60*(6), 1323–1341.

Chan, T. C., Killeen, J., Griswold, W., & Lenert, L. (2004). Information technology and emergency medical care during disasters. *Academic Emergency Medicine, 11*(11), 1229–1236.

Cheng, C.-A., Yan, X., Wagener, N., & Boots, B. (2018). Fast policy learning through imitation and reinforcement. In *Conference on uncertainty in artificial intelligence*. Monterey, California, USA.

Cohen, I., Mandelbaum, A., & Zychlinski, N. (2014). Minimizing mortality in a mass casualty event: fluid networks in support of modeling and staffing. *IIE Transactions, 46*(7), 728–741.

Cruz Jr, G. V., Du, Y., & Taylor, M. E. (2018). Pre-training neural networks with human demonstrations for deep reinforcement learning. In *ALA 2018 workshop*.

Dibangoye, J. S., Amato, C., Buffet, O., & Charpillet, F. (2016). Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research, 55*, 443–497.

Einav, S., Aharonson-Daniel, L., Weissman, C., Freund, H. R., Peleg, K., Group, I. T., et al. (2006). In-hospital resource utilization during multiple casualty incidents. *Annals of Surgery, 243*(4), 533.

Fischer, D., Posegga, O., & Fischbach, K. (2016). Communication barriers in crisis management: a literature review. In *ECIS*. ResearchPaper168.

Foerster, J. N., Assael, Y. M., de Freitas, N., & Whiteson, S. (2016). Learning to communicate to solve riddles with deep distributed recurrent q-networks. *CoRR*. abs/1602.02672.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual multi-agent policy gradients. In *AAAI conference on artificial intelligence* (pp. 2974–2982). New Orleans, Louisiana, USA: AAAI Press.

Gao, Y., Xu, H., Lin, J., Yu, F., Levine, S., & Darrell, T. (2018). Reinforcement learning from imperfect demonstrations. *CoRR*. abs/1802.05313.

Gerchak, Y., Gupta, D., & Henig, M. (1996). Reservation planning for elective surgery under uncertain demand for emergency surgery. *Management Science, 42*(3), 321–334.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Green, L. V., Savin, S., & Wang, B. (2006). Managing patient service in a diagnostic medical facility. *Operations Research, 54*(1), 11–25.

Gupta, J. K., Egorov, M., & Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. In *International conference on autonomous agents and multiagent systems* (pp. 66–83). Sao Paulo, Brazil: Springer.

Hansen, E. A., Bernstein, D. S., & Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th national conference on artificial intelligence* (pp. 709–715). San Jose, California, USA: AAAI Press.

der Heide, E. A. (2006). The importance of evidence-based disaster planning. *Annals of Emergency Medicine, 47*(1), 34–49.

Helm, J. E., AhmadBeygi, S., & Van Oyen, M. P. (2011). Design and analysis of hospital admission control for operational effectiveness. *Production and Operations Management, 20*(3), 359–374.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., … Osband, I., et al. (2018). Deep q-learning from demonstrations. In *Thirty-second AAAI conference on artificial intelligence*.

Hick, J. L., Hanfling, D., & Cantrill, S. V. (2012). Allocating scarce resources in disasters: emergency department principles. *Annals of Emergency Medicine, 59*(3), 177–187.

Hogan, D. E., Waeckerle, J. F., Dire, D. J., & Lillibridge, S. R. (1999). Emergency department impact of the oklahoma city terrorist bombing. *Annals of Emergency Medicine, 34*(2), 160–167.

Huh, W. T., Liu, N., & Truong, V.-A. (2013). Multiresource allocation scheduling in dynamic environments. *Manufacturing & Service Operations Management, 15*(2), 280–291.

Jacobson, E. U., Argon, N. T., & Ziya, S. (2012). Priority assignment in emergency response. *Operations Research, 60*(4), 813–832.

Jenkins, J. L., McCarthy, M. L., Sauer, L. M., Green, G. B., Stuart, S., Thomas, T. L., & Hsu, E. B. (2008). Mass-casualty triage: Time for an evidence-based approach. *Prehospital and Disaster Medicine, 23*(1), 3–8.

Kang, B., Jie, Z., & Feng, J. (2018). Policy optimization with demonstrations. In *International conference on machine learning* (pp. 2474–2483).

Kang, S., Yun, S. H., Jung, H. M., Kim, J. H., Han, S. B., Kim, J. S., & Paik, J. H. (2013). An evaluation of the disaster medical system after an accident which occurred after a bus fell off the Incheon bridge. *Journal of the Korean Society of Emergency Medicine, 24*(1), 1–6.

Kilic, A., Dincer, M. C., & Gokce, M. A. (2014). Determining optimal treatment rate after a disaster. *Journal of the Operational Research Society, 65*(7), 1053–1067.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations*. San Diego, USA: Ithaca.

Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems 12* (pp. 1008–1014). denver, Colorado: MIT Press.

Lakshminarayanan, A. S., Ozair, S., & Bengio, Y. (2016). Reinforcement learning with few expert demonstrations. In *Nips workshop on deep learning for action and interaction*. Barcelona, Spain.

Lee, H.-R., & Lee, T. (2018). Markov decision process model for patient admission decision at an emergency department under a surge demand. *Flexible Services and Manufacturing Journal, 30*(1–2), 98–122.

Lee, H.-R., & Lee, T. (2019). Improved cooperative multi-agent reinforcement learning algorithm augmented by mixing demonstrations from centralized policy. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems* (pp. 1089–1098). International Foundation for Autonomous Agents and Multiagent Systems.

Li, D., & Glazebrook, K. D. (2010). An approximate dynamic programming approach to the development of heuristics for the scheduling of impatient jobs in a clearing system. *Naval Research Logistics (NRL), 57*(3), 225–236.

Li, D., & Glazebrook, K. D. (2011). A bayesian approach to the triage problem with imperfect classification. *European Journal of Operational Research, 215*(1), 169–180.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems 30* (pp. 6379–6390). Long beach, California, USA: Curran Associates, Inc.

Manoj, B. S., & Baker, A. H. (2007). Communication challenges in emergency response. *Communications of the ACM, 50*(3), 51–53.

Mills, A. F., Argon, N. T., & Ziya, S. (2013). Resource-based patient prioritization in mass-casualty incidents. *Manufacturing & Service Operations Management, 15*(3), 361–377.

Mills, A. F., Argon, N. T., & Ziya, S. (2018). Dynamic distribution of patients to medical facilities in the aftermath of a disaster. *Operations Research*.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). Overcoming exploration in reinforcement learning with demonstrations. In *2018 ieee international conference on robotics and automation (ICRA)* (pp. 6292–6299). Brisbane, QLD, Australia: IEEE.

Oliehoek, F. A. (2013). Sufficient plan-time statistics for decentralized POMDPs. In *Proceedings of the 23rd international joint conference on artificial intelligence* (pp. 302–308). Beijing, China: AAAI Press.

Oliehoek, F. A., & Amato, C. (2016). *A concise introduction to decentralized POMDPs* (1st ed.). Springer.

Oliehoek, F. A., Spaan, M. T., & Vlassis, N. (2008). Optimal and approximate q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research, 32*, 289–353.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., & Vian, J. (2017). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th international conference on machine learning: 70* (pp. 2681–2690). Sydney, Australia: PMLR.

Park, Y., Jeong, I., Seo, J., & Kim, J. (2015). A study on the construction of a disaster situation management system in korea based on government 3.0 directive. *WIT Transactions on The Built Environment, 150*, 59–66.

Peleg, K., & Kellermann, A. L. (2009). Enhancing hospital surge capacity for mass casualty events. *JAMA, 302*(5), 565–567.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., & Levine, S. (2018). Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of robotics: Science and systems*. Pittsburgh, Pennsylvania, USA. https://doi.org/10.15607/RSS.2018.XIV.049.

Ramirez-Nafarrate, A., Hafizoglu, A. B., Gel, E. S., & Fowler, J. W. (2014). Optimal control policies for ambulance diversion. *European Journal of Operational Research, 236*(1), 298–312.

Repoussis, P. P., Paraskevopoulos, D. C., Vazacopoulos, A., & Hupert, N. (2016). Optimizing emergency preparedness and resource utilization in mass-casualty incidents. *European Journal of Operational Research, 255*(2), 531–544.

Ross, S., & Bagnell, D. (2010). Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 661–668).

Sacco, W. J., Navin, D. M., Fiedler, K. E., Waddell II, R. K., Long, W. B., & Buckman Jr, R. F. (2005). Precise formulation and evidence-based application of resource–constrained triage. *Academic Emergency Medicine, 12*(8), 759–770.

Sacco, W. J., Navin, D. M., Waddell, R. K., Fiedler, K. E., Long, W. B., Buckman Jr, R. F., et al. (2007). A new resource-constrained triage method applied to victims of penetrating injury. *Journal of Trauma and Acute Care Surgery, 63*(2), 316–325.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *International conference on learning representations*. San Juan, Puerto Rico.

Seo, J., Jeong, I., Park, Y., Kim, J., & Lim, J. (2015). Development of open platform for enhancing disaster risk management. In *Information and communication technologies for disaster management (ICT-DM), 2015 2nd international conference on* (pp. 287–288). IEEE.

Sokat, K. Y., Dolinskaya, I. S., Smilowitz, K., & Bank, R. (2018). Incomplete information imputation in limited data environments with application to disaster response. *European Journal of Operational Research, 269*(2), 466–485.

Subramanian, K., Isbell Jr, C. L., & Thomaz, A. L. (2016). Exploration from demonstration for interactive reinforcement learning. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems* (pp. 447–456). International Foundation for Autonomous Agents and Multiagent Systems.

Sun, W., Bagnell, J. A., & Boots, B. (2018). Truncated horizon policy search: Combining reinforcement learning & imitation learning. In *International conference on learning representations*.

Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., & Bagnell, J. A. (2017). Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International conference on machine learning* (pp. 3309–3318).

Sung, I., & Lee, T. (2016). Optimal allocation of emergency medical resources in a mass casualty incident: Patient prioritization by column generation. *European Journal of Operational Research, 252*(2), 623–634.

Szer, D., Charpillet, F., & Zilberstein, S. (2005). Maa*: a heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the twenty-first conference on uncertainty in artificial intelligence* (pp. 576–583). Edinburgh, Scotland: AUAI Press.

Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., … Vicente, R. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PLoS One, 12*(4), e0172395.

Thompson, S., Nunez, M., Garfinkel, R., & Dean, M. D. (2009). Or practice efficient short-term allocation and reallocation of patients to floors of a hospital during demand surges. *Operations Research, 57*(2), 261–273.

Timbie, J. W., Ringel, J. S., Fox, D. S., Pillemer, F., Waxman, D. A., Moore, M., … Kellermann, A. L. (2013). Systematic review of strategies to manage and allocate scarce resources during mass casualty events. *Annals of Emergency Medicine, 61*(6), 677–689.

Vecerík, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., … Riedmiller, M. A. (2017). Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR*.

Waeckerle, J. F. (1991). Disaster planning and response. *New England Journal of Medicine, 324*(12), 815–821.

Wilson, D. T., Hawe, G. I., Coates, G., & Crouch, R. S. (2013). A multi-objective combinatorial model of casualty processing in major incident response. *European Journal of Operational Research, 230*(3), 643–655.