Reinforcement Learning based Resource Allocation for Initial Disaster Response

by

Esat Tunahan Tuna

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

September 7 2023

Reinforcement Learning based Resource Allocation for Initial Disaster Response

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Esat Tunahan Tuna

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Asst. Prof. Barış Akgün (Advisor)

_____

Asst. Prof. Barış Yıldız

_____

Asst. Prof. Yakup Genç

Date: _____

To my beloved family...

ABSTRACT

Reinforcement Learning based Resource Allocation for Initial Disaster Response
Esat Tunahan Tuna
Master of Science in Computer Science and Engineering
September 7 2023

Effective, fair and quick disaster response is imperative in the aftermath of disasters. Resource limitations, particularly after large-scale disasters like earthquakes, pose challenges in distributing material and human resources. In this thesis, we present a reinforcement learning (RL) based resource allocation approach for disaster response, where a finite amount of resources are dispatched to affected locations.

Our RL formulation is as follows. A 2D map of continuous disaster severity constitutes our state space. Dispatching a single resource to a specific location constitutes the action space. We calculate rewards after allocating all the available resources by running a simple simulation to determine the amount of disaster relieved, reflecting effectiveness, and the spread of the resources across the map, reflecting fairness. We additionally define a per-step reward, based on the local disaster severity distribution, to alleviate issues with sparse rewards. We train two Deep Q-learning agents; one utilizing only terminal rewards and the other incorporating both rewards. Our 2D map formulation induces large state and action spaces. To reduce the number of learned parameters and to add inductive bias, we use convolutional neural networks to approximate the Q-values. We additionally devise a greedy algorithm incorporating per-step rewards as a baseline.

Our evaluation encompasses qualitative behavior assessment on toy maps and quantitative performance assessment on urban maps, both on unseen maps and disaster distributions. Our qualitative assessment reveals that the greedy algorithm places resource units to high disaster severity locations but does not take spread into account as expected, the sparse-reward agent is prone to missing highly concentrated disaster regions, and the other RL agent spreads the units while catching the concentrated regions.

Our quantitative assessment mirrors the qualitative ones; the greedy algorithm

falls behind in resource spread and the sparse reward agent falls behind in the amount of disaster relieved. The greedy algorithm evaluates each location in each allocation step during testing/inference. This leads to two orders of magnitude slower allocation speed, which is related to quickness, compared to the trained agents. Overall, the RL agent trained with both rewards achieves the best performance in terms of allocation speed, disaster relieved and resource spread for novel disaster scenarios.

In this thesis, our main aim is to show the feasibility of RL for large scale resource allocation. As such, we made some simplifying assumptions. We are assuming only one type of resource whereas different regions may require different types (e.g. excavators vs fire engines). This can be handled by training multiple RL agents for each resource type. We are also not taking the distribution cost of the resources into account and assume that the resources can get to where they want to without hindrance. Both of these can be readily incorporated into our disaster simulator and terminal rewards, while requiring additional work on the per-step rewards. Another assumption is about the static nature of the disaster as we started our work for earthquakes. Dynamic disasters such as large scale fires can be incorporated into the simulator stage as well but this would require additional work on the state space to incorporate information on how the disaster may evolve.

This thesis presents the first resource allocation approach for disaster response that can work with large state and action spaces without assumptions on the objective structure, in addition to the potential of using arbitrarily complex objectives and incorporating environment stochasticity, to the best of our knowledge. Our work paves the way for further developments that can incorporate further developments such as more complicated disaster scenarios and objective functions to remove the simplifying assumptions.

ÖZETÇE

Afetle Mücadelede Pekiştirmeli Öğrenme tabanlı Kaynak Yönetimi
Esat Tunahan Tuna
Bilgisayar Bilimleri ve Mühendisliği, Yüksek Lisans
September 7 2023

Afet sonrası müdahalelerin etkin, adil ve hızlı olması mecburidir. Deprem gibi büyük çapta etkileri olan afetlerin ardından kaynakların sınırlı olması, bu kaynakların yönetimi noktasında afetle mücadeleye güçlük oluşturmaktadır. Bu tez çalışmasında, afetten etkilenen bir bölgeye müdahalede sınırlı olan bu kaynakların yönetimi için pekiştirmeli öğrenme tabanlı kaynak yönetimi yaklaşımımızı sunuyoruz.

Pekiştirmeli öğrenme tabanlı yaklaşımımız şu şekildedir. İki boyutlu bir afet şiddet haritası bizim durum uzayımızı belirtmektedir.Bir kaynağın, destek biriminin, belirli bir noktaya atanması ise eylem uzayımızı ifade eder. Basit bir simulasyon yardımıyla sanallaştırdığımız afet senaryosunda destek birimlerinin sönümlendiği afet hasarları miktarı kaynak yönetiminin etkinliğini, kaynakların bölgeye yayılım miktarı kararların ne kadar adil olduğunu gösterir. Bu metrikleri kullanarak tüm atama sonrasında bir ödül hesaplaması yapıyoruz. Ayrıca, bu formulasyonda karşılaşılan problemleri azaltmak amaçlı her adımda ödüllendirme esaslı bir hesaplamamız daha mevcut. Çalışmamızda, iki farklı derin q-öğrenmesi tabanlı ajanlar eğittik. Biri sadece nihai durumda ödüllendirilirken, diğeri hem her adımda hem de nihai durumda ödüllendirildi. İki boyutlu olarak tasarladığımız dünya modellemesi durum ve eylem uzaylarının fazlaca geniş olmasını da beraberinde getirdi. Parametre sayılarını düşürmek ve model varsayımı eklemek için kullanacağımız modeli evrişimli sinir ağları üzerine kurguladık. Ayrıca, açgözlü yaklaşım ajanı geliştirip, her adımda ödüllendirme esaslı yaklaşımımıza bir taban oluşturduk.

Modelleri değerlendirme süreçlerimiz iki farklı şekildedir. Birincisi, oyuncak haritalar diye isimlendirdiğimiz senaryolarımız üzerinde niteliksel davranış değerlendirmeleri yapıyoruz. İkinci olarak, kentsel senaryolarımızda ise niceliksel değerlendirmelerimizi yapıyoruz. Her iki değerlendirmede de kullandığımız senaryolar modellerin eğitilme aşamasında hiç karşılaşmadığı haritalardır. Nitelik değerlendirmelerimizde açgözlü

yaklaşımın, beklendiği gibi, yayılımı dikkate almadığı, sadece afet şiddeti yoğun olan bölgelere kaynak ataması yaptığını gözlemledik. Nihai durumda ödüllendirdiğimiz ajanımızın ise çok yoğun olan bölgeleri gözden kaçırdığını, her adımda ve nihai durumda ödüllendirdiğimiz ajanımızın ise hem yayılımı hem de şiddetli noktaları dikkate aldığını ortaya koyduk.

Niceliksel değerlendirmelerimizde de niteliksel değerlendirmelerimize paralel çıkarımlar elde ettik. Açgözlü ajanımız yayılım hususunda başarısız olduğunu ve nihai durumda ödüllendirdiğimiz ajanımızın afet sönümlemelerinde geride kaldığını gözlemledik. Açgözlü ajanımız her lokasyonun değerini her adımda hesaplaması sebebiyle eğitilmiş modellerimize nazaran çok geride kaldığını, karar alım sürecinin çok uzun olduğunu belirledik. Genel olarak ifade etmek gerekirse, hem her adımda hem de nihai durumda ödüllendirdiğimiz ajanımız en iyi performansı gösterdi. Karar alımının hızlı olduğunu, atamalar sonrasında destek birimlerinin sönümlediği afet miktarının daha fazla olduğunu ve yayılımın daha geniş kapsama alanını etkilediğini niceliksel ve niteliksel olarak ortaya koyduk.

Bu tez çalışmasında, ana amacımız pekiştirmeli öğrenmenin geniş çaplı kaynak yönetimi problemlerinde uygulanabilir olduğunu göstermektir. Öyle ki, çalışmamızda bazı basitleştirici varsayımlar yaptık. Örneğin, afetten farklı etkilenen farklı lokasyonlar için değişik tiplerde destek birimleri gerekiyor olsa da, çalışmamızda tek tip bir kaynak üzerine yoğunlaştık. Aksi durumda, her bir tip destek birimi için birden fazla yapay zeka ajanı eğitiyor olmamız gerekebilirdi. Ayrıca, kaynakların dağılım maliyetini de göz önüne almadık. Kaynak ataması sonrasında destek birimlerinin ilgili yerlere herhangi bir engel olmadan ulaşılabilir olduğu bir dünya tasarladık. Her ne kadar her adım ödüllendirme esaslı yaptığımız çalışmalarda ayrıca bir geliştirme yapma ihtiyacımız doğsa da, bu hali ile afet simulatörümüz ve nihai durum ödüllendirmelerimizi yeterli bir şekilde kapsayacak altyapı ihtiyacımızı karşıladı. Diğer bir varsayımımız ise, durağan bir afet doğası üzerinde çalışıyor olmamızdır. Çalışmamıza deprem odaklı başlamamız sebebiyle araştırma süreçlerimiz bu şekilde ilerledi. Büyük ölçekli yangınlar gibi dinamik felaketler de simülatöre dahil edilebilir ancak bu, felaketin nasıl geliştiğine ilişkin bilgilerin de yaklaşımımız ve altyapımıza dahil edilmesi için durum uzayında ek çalışma yapılmasını gerektirecektir.

Bu tez çalışması, gelişigüzel karmaşık hedeflerin kullanılması ve ortam stokastisitesinin dahil edilmesi potansiyeline ek olarak, hedef yapıya ilişkin varsayımlar olmadan geniş durum ve eylem uzaylarıyla çalışabilen afet müdahalesine yönelik ilk kaynak tahsis yaklaşımını sunmaktadır. Çalışmalarımız, basitleştirici varsayımları

ortadan kaldıracak daha karmaşık afet senaryoları ve hedef fonksiyonları ile bu alanda yapılacak çalışmaların önünü açacaktır.

ACKNOWLEDGMENTS

I would first like to express my endless gratitude to my thesis advisor Assist. Prof. Barış Akgün. I have not only seen my professor as my advisor, but I have also seen as the architect of my academical success and happiness. I had always desired to start my masters education at Koç University with Assist. Prof. Barış Akgün, and my professor always supported me in this direction. He never gave up supporting me even if I deserved to be left many times. He has always been there to help, always. I will never forget his countless supports to the end of my life.

Masters education made me experience the *what doesn't kill you makes you stronger*. I want to thank Koç Family and Koç University. Their contributions to my life emerge as success, happiness, strength and many other. I also wanted to thank my colleagues and my managers in my workplace, Havelsan, for encouraging me to persue my education.

Last but not least, I want to thank my family. Without their endless support and encouragement, none of my accomplishments would be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

RL	Reinforcement Learning
IBB	The Istanbul Metropolitan Municipality
RARL	Resource Allocation with Reinforcement Learning
MDP	Markov Decision Processes

Chapter 1

INTRODUCTION

The initial response to a disaster plays a pivotal role in mitigating the adverse consequences endured by the affected population and in averting subsequent catastrophes [Tomasini et al., 2009]. A crucial part of disaster response is deploying the necessary material and human resources to the impacted areas. However, this is frequently hindered by the discrepancy between the demand and supply of resources, particularly after major disasters like earthquakes. As a result, ensuring effectiveness, fairness, and quickness in a disaster response remains a formidable challenge.

Towards this end, we propose a reinforcement learning (RL) based resource allocation approach for disaster response, called Resource Allocation with Reinforcement Learning (RARL). We target disasters covering a large region with multiple centers of devastation. We assume that a disaster severity assessment has been made and the limited amount of resources are to be dispatched based on the desired criteria. There are optimization and RL based approaches dealing with similar scenarios in the literature (see Section 2). However, their action spaces, and sometimes state spaces are too small to be realistic. Optimization methods that can deal with large spaces impose structural constraints on the objectives to be optimized.

In RARL, we reformulate resource allocation as a sequential decision making problem. Given finite amount of resources, the idea is to allocate each unit of resource one-by-one in order to maximize the sum of rewards. Since there are limited resources, this induces a finite horizon problem. In addition, resource allocation order does not matter which removes the need to use a discount factor. We use a 2D map representation of the affected geographical area, analogous to an image,

as our state space. In this map, each cell holds the disaster severity of the corresponding location. An action is defined as sending a single rescue unit to a location. We carefully design a transition function that reduces the disaster severity locally after allocation to avoid enlarging the state space with an additional channel, and to be robust against number of resource units, both of which helps with training and generalization. We use three metrics to assess the performance of an entire allocation; (1) the amount of disaster relieved corresponding to effectiveness, (2) the spread of the rescue units corresponding to fairness and (3) the speed of making the allocation decisions corresponding to quickness. We use a post-allocation simulation to calculate the first metric, calculate the second metric directly after the allocation and calculate the third metric during the allocation. The combination of disaster relieved and spread is used as our terminal reward which is sparse. We also devise a per-step (dense) reward by looking at the local disaster severity distribution of the action location for faster training.

We utilize Deep Q-Learning to train two agents, one with just the sparse rewards (called the RARL-S agent) and another one with both of the rewards (called the RARL agent). We use pure convolutional neural networks (CNNs) from the state input to action value output to approximate the Q-values. This seamlessly matches with our state and action formulations, allows us to deal with large state and action spaces by keeping parameter numbers low and improves performance by injecting inductive bias. During testing/inference, the action with the highest value is picked at each time step. We additionally develop a greedy agent that goes over each location to calculate per-step rewards of allocating a unit there at each time step, and picks the action that result in the highest.

We train the RL agents on three maps and evaluate the performance of all three agents on previously unseen test maps. First, greedy algorithm runs two orders of magnitude slower than the RL agents due to its nature. Qualitative analysis on two toy maps show that the RARL-S agent is prone to missing high concentration regions and the greedy algorithm does not care about the spread as expected. Quantitative analysis on urban city maps, reflect this trend as well; RARL-S agent falls short on the amount of disaster relived and the greedy agent falls short on spread. Only the

RARL agent performs on par or better than the rest on all the metrics, satisfying our effectiveness, fairness and quickness ideals for allocation.

We argue that RARL can handle arbitrary finite terminal objectives as long as the state space is rich enough to represent them and that actions can affect them. This is due to generality of the RL itself (i.e. no assumptions on the reward distribution other than boundedness). RARL can handle the disaster response stochasticity as well based on similar reasoning. More complex and more stochastic settings can be incorporated into the post-allocation simulator to reflect the difficulties, intricacies and uncertainties of disaster response.

To summarize, we introduce a reinforcement learning based resource allocation method that (i) helps with an effective, fair and quick disaster response, (ii) works on large scale disasters on a large geographical region, (iii) works with scarce resources, and (iv) generates granular action decision. Our method;

1. Handles large action and state spaces. Requirements (ii) and (iv), and indirectly (iii) as scarcity dictates the need to make granular decisions.

2. Learns to make effective and fair decisions. Requirement (i).

3. Makes fast decisions (aka testing or inference). Requirement (i).

4. Generalizes to novel disaster distributions. Requirement (i)

5. Is robust against initial amount of resources. Requirement (iii)

1.1 Contributions

Our results show that RL, specifically with our careful formulation - RARL -, is a feasible way to build an initial resource allocation decision aid for disaster response. In summary, our contributions are:

- Converting a large scale resource allocation problem to a sequential decision making problem and using RL to tackle it.

- Utilizing pure CNNs to approximate Q-values which lets us handle large state and action spaces, in addition to the ability to run fast inference with common hardware and software.

- Designing a state space representation and a transition function to reduce state space size and increase robustness against available resources which help with training and generalization.

- Empirically demonstrating our method's ability to generalize to novel disaster distributions and even novel maps along with robustness to initial resource availability, in terms of the disaster relieved and spread metrics.

- Potential ability to incorporate arbitrarily complex terminal rewards and stochasticity by utilizing a disaster response simulator

1.2 Thesis Statement

Deep reinforcement learning in conjunction with using convolutional neural networks as approximators and a purpose designed problem formulation can be used to develop an effective, fair and quick decision aid for granular rescue personnel and material allocation across large geographical locations for initial disaster response. The learned models can be generalizable to novel situations in addition to being easy to train and practically operationalized due to the approximator choice. Such an approach outperforms heuristic baselines in terms of amount of disaster relived, fair spread of resources and decision speed.

1.3 Outline

The rest of this thesis is organized as follows:

- Ch. 2: Explains current state of literature regarding reinforcement learning based resource allocation in disaster management.

- Ch. 3: Describes our novel method in detail.

- Ch. 4: Presents our experimental setup and results.
- Ch. 5: Summarizes the contribution in this work and discusses possible future directions.

Chapter 2

LITERATURE REVIEW

Resource allocation for disaster management is a complex task with many constraints, dependencies and limitations. This complexity requires the problem formulations to have comprehensive environment representations, leading to large-scale contexts. There are traditional optimization-based resource allocation methods for disaster response and they either work on simplifying assumptions (linear objectives, simplified or softened constraints, simplified relation structure etc.) or employ heuristics to be able to get solutions in reasonable durations [Yu et al., 2018].

2.1 Resource Allocation in Disaster Management

[Wex et al., 2014] propose and compare several heuristic approaches for rescue unit assignment and their scheduling. These heuristics include a Monte Carlo-based heuristic the joint application of multiple construction heuristics, improvement heuristics, and GRASP meta-heuristics. They also found that even small-scale cases can not be solved optimally in a reasonable time without heuristics. [Yu et al., 2018] propose a combined exact-heuristic multi-period resource allocation problem in emergency logistics. They use a dynamic programming approach to optimally solve small-scale instances of a nonlinear problem and a heuristic delivery model to solve large-scale instances. However, even for their experiments with larger-scale problems (state space size with 100000), the amount of affected areas is kept low (1 to 10) to avoid blowing up the solution space. [Fiedrich et al., 2000] present an optimization approach for resource allocation for emergency response after an earthquake. They propose a dynamic optimization model to find the best assignment of machines to perform tasks with the goal of minimizing the total number of fatalities. They also include the time factor in their formulation, to maximize the outcomes of search

and rescue operations. However, they severely reduce their decision/action space, compared to their problem scales with the machines and tasks formulation, making the approach unrealistic. [Sheu, 2007] presents a clustering based resource allocation approach for earthquake relief. They incorporate temporal relief demand forecasts in their clustering algorithm along with dynamic relief supply. However, their state space is also small. [Gong and Batta, 2007] focuses on ambulance allocation and relocation for a post-disaster relief operation. They constructed a deterministic model to study on and they presented two iterative procedures to optimize the make span and the weighted total flow time. [Pérez-Rodríguez and Holguín-Veras, 2016] studied on inventory-allocation distribution models with consideration of deprivation cost for post-disaster humanitarian logistics. They developed an inventory-allocation-routing model for optimal assignment of critical supplies that minimizes social cost, designed suitable heuristic solution and assessed the performance of the heuristics using numerical experiments. [Xiang and Zhuang, 2016] studied on medical resource allocation problem for serving emergency victims. They proposed a queueing network model to maximize the service rate and minimize the total expected death rate and total waiting time. [Hu et al., 2016] studied on emergency resource allocation problem which involves multiple competing affected areas and one relief resource center under supply shortage and uncertainty in the post-disaster phase. They formulated a bi-objective robust model (BRERA) to maximize both efficiency and fairness.

2.2 Reinforcement Learning for Resource Allocation

Reinforcement Learning (RL) is a sub-field of machine learning, in which the aim is to train an agent to make sequential decisions from environment interaction data by mostly relying on reward feedback from the environment [Sutton and Barto, 2018]. RL has become a powerful tool to tackle complex sequential decision-making problems, especially after the advent of deep reinforcement learning [Mnih et al., 2015]. It has found applications in many diverse fields such as control, robotics, natural language processing, marketing, gaming and more. RL approaches has also

found their way into resource allocation problems, for example within the domains of communications ([Aihara et al., 2019]; [Ye et al., 2019]), transportation ([Jiang et al., 2019];[Jiang et al., 2018]), inventory management ([Kara and Dogan, 2018]), water source management ([Ni et al., 2013]), cloud computing ([Vengerov, 2007]) etc.

[Kwon et al., 2008] developed a case-based myopic reinforcement learning (CMRL) algorithm for a supply-chain inventory control problem. Their method aims to alleviate the slow convergence of traditional RL algorithms when applied to large action spaces. Even though their action space is also large, their action selection and the demand are constrained depending on the state, simplifying the problem. [Kara and Dogan, 2018] applied Q-learning and SARSA algorithms to an inventory management problem of perishable goods under random demand and deterministic lead time. They used two different state representations, quantity based 41 discrete actions, a retailer cost based reward function. [Jiang and Sheng, 2009] proposed a case-based reinforcement learning algorithm (CRL) for dynamic inventory control in a multi-agent supply-chain system. [Jiang et al., 2018] studied on coordinated passenger inflow control of urban rail transit in peak hours. They proposed a Q-learning algorithm to optimize inflow volume at each station with the aim of minimizing the safety risks imposed on passengers. [Šemrov et al., 2016] developed a Q-learning based train scheduling algorithm. They state that empirical results show that Q-learning lead to rescheduling solutions that are at least equivalent and often superior to those of several basic rescheduling methods that do not rely on learning agents. [Valente Klaine et al., 2018] studied on availability of network operators during an emergency. They state that the deployment of an intelligent, mobile and adaptable network via UAVs is possible alternative for emergency situations. Therefore, they proposed an RL based solution to find the best positions of multiple drone small cells (DSCs) in an emergency scenario. The solution's main goal is to maximize the amount of users covered by the system, while drones are limited by both backhaul and radio access network constraints.

RL has been applied to disaster management as well, with rescue unit allocation, inventory management, transportation, and search and rescue applications.

[Su et al., 2011] studied on selection of rescue path when a disaster strikes. They developed a Q-learning based path selection algorithm as a disaster response. [Nadi and Edrissi, 2016] proposed a stochastic programming model to evaluate real-time relief demand and network condition at the onset of a natural disaster. They also proposed a reinforcement learning model for optimization of the total relief assessment time to address the time sensitivity of the emergency response. [Nadi and Edrisi, 2017] proposed a Markov decision process, which works online and offline, to coordinate relief assessment and emergency response teams. [Yan et al., 2020] worked on a mobility based rescue team dispatching problem. They proposed a SVM based model to predict potential rescue requests and developed a RL based rescue team dispatching model to effectively do the assignment. [Yu et al., 2021] proposed a RL based method for rescue team dispatching during a disaster. They perform Q-learning with rewards reflecting efficiency, effectiveness and fairness. This is the most similar paper to our work but we aim to allocate all the initially available resources for immediate disaster response and their state and action spaces are a few orders of magnitude smaller than ours.

None of the aforementioned methods perform rescue unit allocation using large, map-like state spaces and equally large action spaces which is the focus of our work. In addition, most of the classical approaches cannot handle stochastic environments seamlessly and cannot employ arbitrary cost functions at these scales.

Chapter 3

RESOURCE ALLOCATION WITH REINFORCEMENT LEARNING

The big picture disaster response involves (1) disaster severity assessment, (2) deciding on the amount and type of initial resources to send (a percentage of resources are kept back in case another disaster hits), (3) performing allocation and (4) dispatching and monitoring the units. We define **disaster severity** as a high level concept that reflects the impact of multiple adversarial affects caused by the disaster, such as amount of building damage, ongoing fires, amount of infrastructure damage, number of fatalities, number of injured people etc. We define a **rescue unit** as another high level concept, that reflects the combination of materials and human rescuers needed for the disaster in question. In our case, we assume that (1) is available to us as input. We do not assume anything about (2) and design our approach to be robust to initial amount of resources. We argue that the actual task of (4) is out-of our scope but we use a simple simulator to gauge the allocation performance. Thus, our approach is mainly about performing the allocation.

In our approach, we formulate the resource allocation problem as a sequential decision making problem and tackle it with reinforcement learning. The idea is to allocate each unit one-by-one (an action), use the problem snapshot with partial-allocation as the state, and use the allocation objective as the terminal reward. This is a finite horizon problem since there are limited resources. Since this is a finite horizon problem and that the order of allocation does not matter, there is no need for discounting. We are targeting disasters covering a large region with multiple centers of devastation. As such our method must be able to handle large state and action spaces. We aim to be effective and fair in our allocation, which must be captured by our reward formulation. We also aim to be quick, which means that inference

must be fast. Implicitly, the underlying machine learning model must be expressive enough to handle large input and output spaces while capturing the nuances of the long-term rewards, and small enough to perform fast inference. We designed our approach, called Resource Allocation with Reinforcement Learning (RARL), to handle these goals.

We formulate the problem as a Markov Decision Process (MDP) which is defined as the tuple $< S, A, R, T >$ where the individual elements define the state space, action space, reward function and transition dynamics respectively. We then use this formulation with Deep Q-Learning to train resource allocation agents. In this section, we describe the MDP elements along with other components that we utilize.

3.1 Representations of State and Action

We start by defining the state and action spaces, and our designed transition dynamics here.

State Space: We use a MxN map representation, analogous to an image, of the disaster affected geographical area as our state. Each cell of this map stores the normalized disaster severity of its corresponding location. Furthermore the disaster severity is non-negative, thus individual cells store real values between 0 and 1. A single state is a snapshot of the disaster severity distribution in the map. Formally $S = \{s : s \in \mathbb{R}^{M \times N}, 0 \leq s_{ij} \leq 1, i \in [0, M), j \in [0, N)\}$, where s_{ij} is an individual cell within the map.

Action Space: An action is defined as allocating a single rescue unit to a location. Since the action space has the same dimensionality as the state space, we treat an action as picking a coordinate in a MxN map. Formally, $A = \{a : a = (i, j), i \in [0, M), j \in [0, N)\}$. This representation allows us to model the Q-values as a 2D map.

Transition Dynamics: The state space only contains disaster severity and not the amount of allocated units, thus allocating a unit to a location does not have any natural transition dynamics. However, we design the dynamics to our advantage. After each allocation, the disaster severity is lowered in a patch by an amount

Algorithm 1 Transition Function

1: **Input:** The current disaster state s and action (i, j) which is the location to send a rescue unit
2: **Output:** Next state and reward after taking an action
3: $r \leftarrow reward(s, (i, j))$
4: **for** $k, l \in \mathcal{N}_{ij} \cup (ij)$ **do**
5: impact $\leftarrow calculate\_impact(s_{i,j}, s_{kl})$
6: $s_{kl} \leftarrow s_{kl}$ - impact
7: **end for**

proportional to the Chebyshev distance to the allocation location.

As mentioned in Chapter 1, we use a post-allocation simulator (Chapter 3.2), to evaluate the success of an allocation. During the simulation, allocated rescue units move on to relieve the disasters in their neighborhoods. The transition function represents this to an extent and as such helps with the problem.

The implementation of our transition function is presented in Alg.1. In this algorithm, s_{ij} is the disaster severity at (i, j) and $\mathcal{N}_{ij}$ denotes the $n \times n$ neighborhood centered around (i, j) (excluding itself) with odd n. The *reward* function (line 3) is described in Chapter 3.4. The impact is calculated using Eq. 3.1 as follows:

$$I((ij), s_{kl}) = d_{rb}(s_{ij}, s_{kl}).\frac{W_{RS} \cdot \tau}{n^2} \tag{3.1}$$

where W_{RS} is the amount of disaster that a rescue unit relieves per simulation time-step, τ is the total number of simulation time-steps and $d_{rb}(s_{ij}, s_{kl})$ is a custom distance function calculated as follows:

$$d_{rb}(s_{ij}, s_{kl}) = c_u + \frac{(c_l - c_u)(d_C(s_{ij}, s_{kl}) - 1)}{(d - 1)} \tag{3.2}$$

where d_C is the Chebyshev distance, $d = (n - 1)/2$, is the farthest distance within the neighborhood, c_u and c_l set the upper and lower values that d_{rb} can take.

The reason for such a transition choice is three-fold; (1) to keep the state space size under control, since adding another channel to keep allocated units would double the size, (2) to reflect the effect of allocations, albeit indirectly, and (3) to make the

method robust against the initial number of rescue units. Implicitly representing the effects of the allocations removes the ambiguity induced by an explicit rescue unit representation. For example using a normalized representation w.r.t. the initial number of units leads to a distribution shift if available units in inference time is different than the training time. Keeping raw counts results in scale issues w.r.t. the disaster severity and an induced upper limit based on training. This transition dynamics is only exploited for RL training, and simulator is initialized with the original disaster severity values.

3.2 Disaster Simulator

As mentioned above, we assign units one-by-one until there are none left which defines an episode in RL terms. We call it a **complete allocation** if all the units are allocated. At this point, we want to compute the value of the "allocation objective" of a complete allocation, to use as the terminal reward of our finite horizon problem.

Realistic and generic objectives cannot be directly defined on the disaster severity distribution and the complete allocation. The objective maybe nonlinear, complex, affected by interaction of resource units and potentially be affected by environment stochasticity. For example, in a realistic scenario, rescue units may potentially move between neighboring locations due to wrong severity estimates, lack of necessary equipment for the current location (e.g. an earth digger for a severe wreckage, lack of an electric generator within reach etc.) or simply finishing their work. Similarly, they may never even reach their desired destination due to road blockages. Multiple rescue unit may also be assigned to a region with a large concentrated disaster severity with relatively safe neighbors. After they are done with the current location, they may not have anything close by to help with, resulting in wasted opportunity. As such, we argue that a disaster simulation should be used to assess the performance and calculate the allocation objective. Such a simulator can be made arbitrarily complex without affecting the other parts of the problem, as long as the objectives and stochasticity can be represented with the states and affected by the actions.

For this work, we build our own rescue unit behavior simulator. This simulator

takes the original disaster severity (unaffected by the transition dynamics) state and the complete allocation as input, simulates individual units for a duration, and returns the final disaster severity state. We use a discrete time simulation approach. Each individual rescue unit relieves the disaster severity at their assigned location by a constant number for each time step. Multiple rescue units can be at a single location and their effects are combined additively. When the disaster severity at a location becomes zero, a rescue unit moves to the location with the highest disaster severity within its 8-neighborhood, with random tie-breaks. The simulator stops after a number of steps.

Our disaster simulator is relatively simple yet it can be applied to a variety of disasters due to its simplicity. More features such as additional sources of randomness, break times, potential of injury/equipment failure, travel time etc. can be added. Even more complex simulations (e.g. [Lee and Lee, 2021] where each allocated agent is an RL agent itself) can be incorporated. However, care must be taken to balance the realism with computational issues. We argue that building a complex yet fast enough simulator would be a separate work onto itself, making it out of our current scope.

3.3 Allocation Objectives and Metrics

We want to evaluate the effectiveness, fairness and quickness of a complete allocation, as motivated in Chapter 1. Effectiveness is important to avoid wasting resources. On top of this, the resources must be fairly distributed among the affected population. Lastly, allocation decisions must be made quickly. There is a not so obvious requirement for the last one; There may be multiple sources of disaster severity assessments (reporting from the ground, simulations, drone or satellite footage, self-assessment etc.) and that the response team may want to evaluate multiple options quickly. We define quantitative metrics to represent these ideals so we can perform appropriate allocations.

Disaster Relived: The amount of disaster relived given a complete allocation. This metrics corresponds to effectiveness. It is calculated as the difference between

the total disaster severity of the starting distribution (s^0) and that of the distribution after simulation (s^f), calculated as follows:

$$J_{dr} = \sum_{i,j} s_{ij}^0 - \sum_{i,j} s_{ij}^f \tag{3.3}$$

Spread: The determinant of the covariance matrix of the allocation locations. The covariance matrix describes how spread the allocations are to an extent and its determinant gives its area. Note that this is calculated before the simulation. This metric is one aspect of fairness. It is calculated as follows:

$$J_{spr} = \det\left(\frac{\sum_{k=1}^{K}(a_k - \bar{a})(a_k - \bar{a})^T)}{K-1}\right) \tag{3.4}$$

$$\bar{a} = \frac{1}{K}\sum_{k=1}^{K} a_k \tag{3.5}$$

where K is the number of allocated units, $a_k = [i_k, j_k]^T$ is the k^{th} action and $\bar{a}$ is the mean action vector.

Unique Locations Visited: This is number of unique locations visited by rescue units. It is counted as the initial locations with at least one allocated rescue unit plus the number of new locations visited by the rescue units during simulation. This metric is another aspect of fairness.

Allocation Speed: This is measured as the wall-clock time to perform a complete allocation. This metric corresponds to quickness. It is mainly a practical metric since it is affected implementation details, and availability of parallel computation hardware and software, in addition to the algorithmic complexity.

The first three metrics are related to the allocation decisions and the last one is related to the allocation algorithm. Furthermore, number of unique locations visited is partially dependent on the number of available units which we want to avoid (Sec. 3.1, transition dynamics discussion). As as result, we take the first two as the major allocation objectives and report the last two as additional results in our evaluation.

3.4 Reward Functions

We need to define a reward function in order to perform reinforcement learning. The terminal reward comes naturally with the metrics defined in Sec. 3.3 as the linear combination of disaster relieved and spread. It is calculated as follows:

$$rs_t(s,a) = \begin{cases} w_0 J_{dr} + w_1 J_{spr}, & \text{if } t = K \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

where t denotes the allocation step, K is the number of resource units, J_{dr} and J_{spr} comes from Eqs. 3.3 and 3.4, and w_0 and w_1 are hyperparameters to tune the contributions of each component.

This is a sparse reward since it is only received at the end of an episode. Sparse rewards usually lead to slow convergence rates, even to the extent of making the problem infeasible. To alleviate this, we engineer a dense reward. This reward must be in line with the allocation objectives and must not overpower the terminal rewards. On the other hand, it must be fast to evaluate. As such we develop the following dense reward that looks at the local disaster severity distribution of the allocation point:

$$rd_t(s, a = (i,j)) = s_{ij} + \sum_{k,l \in \mathcal{N}_{ij}} d_{rb}(s_{ij}, s_{kl}) s_{kl} \tag{3.7}$$

where s_{ij} is the disaster severity at (i,j), $\mathcal{N}_{ij}$ denotes the $n \times n$ neighborhood centered around (i,j) (excluding itself) with odd n and $d_{rb}(s_{ij}, s_{kl})$ is calculated with Eq. 3.2.

This dense reward looks at the local disaster severity distribution and not just the severity on the target location. As such, it promotes patches with larger disaster which is inline with resource unit behavior of moving around if needed. Coupled with the transition dynamics of dampening the disaster severity of patches after allocations, it indirectly helps with the spread.

3.5 RL Agents

In RL nomenclature, we call the decision making program an agent. We can readily train two distinct RL agents using our formulation, one with the sparse rewards and the other one with both sparse and dense rewards. We keep all the other properties the same. These agents are:

- **RARL-S Agent**: This agent is trained only with terminal rewards given by Eq. 3.3.

- **RARL Agent**: This agent is trained with both of the rewards, represented by Eq. 3.8.

$$r_t(s,a) = rs_t(s,a) + rd_t(s,a) \tag{3.8}$$

3.6 Greedy Agent and Relation to Other Classical Methods

We develop a greedy agent incorporating per-step rewards in addition to RL agents for two main reasons. The first reason is that there is an absence of analogous methodologies with comparable objectives, and state and action spaces both in terms of content and scale. As such, we utilize this greedy algorithm as a baseline. The second reason is that this greedy agent would almost be as good as the best RL agent trained only with dense rewards (discount factor being 1 along with the cumulative sum of rewards objective of RL). As such, it allows us to test the efficacy of dense rewards without full RL training.

The decision rule of greedy agent at each time step is given in Eq. 3.9. The greedy algorithm applies this for K times (number of initial rescue units). We apply the same transition dynamics on the state as the RL agents.

$$a_t = \arg\max_{a \in A}(rs_t(s,a)) \tag{3.9}$$

Even though we do not perform an analysis, we think that the greedy agent is optimal with respect to the cumulative per time-step metric. The reasons for this thought are; (1) we are not taking the path of the rescue units in the cost

function and (2) the transition function affects the states linearly. Assuming that the greedy agent is optimal, we do not need to compare against any other classical algorithm such as the ones used for Prize Collecting Travelling Salesman Problem, Prize Collecting Vehicle Routing Problem, or Set Cover Problem. In fact, we can show that under the cumulative return maximization problem is a special case of these under our assumptions.

We want to highlight one major disadvantage of all related classical algorithms for our setting; They cannot utilize arbitrary terminal rewards. This is the most important advantage of reinforcement learning against these methods for our study. They can only perform as good as the per time-step reward's ability to estimate the terminal rewards. On the other hand, RL agents also benefit from good per time-step rewards. In addition, terminal rewards can include problem non-linearity and stochasticity, making the per time-step reward design difficult and making things harder for classical algorithms. RL agents can focus on the terminal rewards, natively handling these issues. On the other hand, our current scenarios assume that the rescue units "teleport" to their locations, i.e., we do not penalize the cost of getting to the allocation location. This is the biggest disadvantage of our RL agents. However, we note that this is not an inherent limitation of RL. In fact, RL agents are adept at "shortest-path" problems. Our decision to exclude path costs is due to concentrating on the other aspects (e.g. large-state action spaces, problem formulation, neural network architecture design etc.).

Chapter 4

EXPERIMENTS AND RESULTS

We describe our data, training details and evaluation approach in this section. We use three types of data; toy maps, Istanbul maps with various disaster distributions and other urban maps. We train our models on one toy map and two Istanbul maps and test their performance on unseen maps consisting of each type.

We evaluate all the agents qualitatively and quantitatively. We perform qualitative assessment on toy maps to investigate the behavior of the agents. We then perform quantitative analysis on Istanbul maps to comparatively assess the agents based on the criteria described in Chapter 3.3. We also perform quantitative analysis on non-Istanbul maps to further demonstrate their generalization performance.

We first describe our maps, followed by training details. We then move on to evaluations.

4.1 Maps

4.1.1 Toy Maps

We designed small-scale disaster severity maps to qualitatively analyze the model behavior. These maps reflect the objectives we consider to formulate our approach for the underlined resource allocation problem.

For instance, Figure 4.1 was designed to test model behavior in terms of effectiveness. As indicated, amount of disaster relieved at the end of the simulation reflects the effectiveness of the distribution. In addition, consideration of regions with high concentration indirectly reflects the effectiveness in a decision since it implies that an agent tends to prioritise affected areas. There are identical 10 small circles that represent remarkable places in the map. They are randomly located and there is no connection or relation among them. The expectation is that an agent should notice

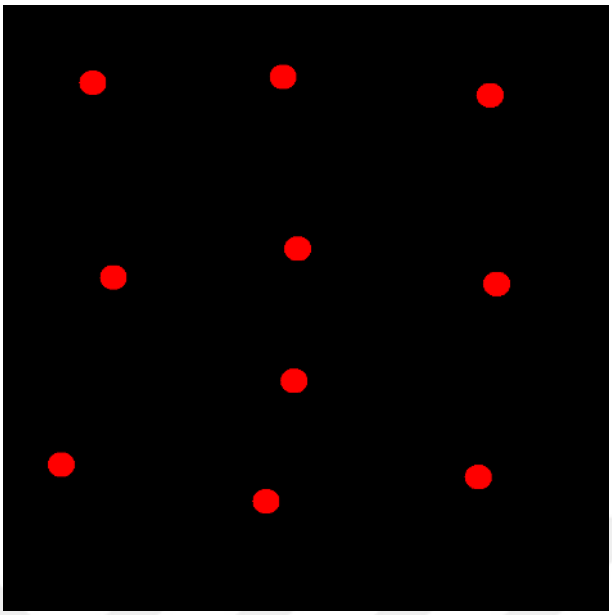

Figure 4.1: Example Toy Map 1 - Remarkable Places

all those circles since these circles were designed as such one rescue squad would be enough to relieve all the disaster damages in a circle. We use 10 rescue squads for this map.

For another instance, Figure 4.2 was designed to test model behavior in terms of fairness. There are 8 squares randomly located in the image. Each square represents affected local regions. A model is expected to distribute limited resources fairly to those squares. To interpret the fairness of the distribution, we test a model on this map with 8 rescue units.

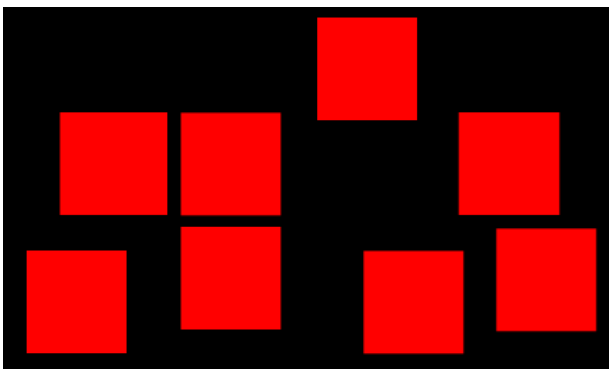

Figure 4.2: Example Toy Map 2 - Randomly Located Squares

Figure 4.3: Istanbul Map 1 - Population based representation

4.1.2 Istanbul Maps

The Istanbul Metropolitan Municipality (IBB) studied on an earthquake scenario and published the results and expectations of underlined disaster [The Istanbul Metropolitan Municipality, 2021]. The published source includes the expected possible deaths, numbers of wounded people, number of people needing immediate medical attention, infrastructural damage, and so on. We used this source to generate our disaster severity maps. These are given on a neighborhood level. Thus, we map the corresponding disaster density values to each neighborhood to create a complete disaster map.

In generated Istanbul maps, each neighborhood has a disaster density value calculated using a linear combination of indicators presented in IBB earthquake scenario study. For example, one map is composed of human related factors such as number of critically wounded people. We used this value as disaster density for corresponding neighborhood. For another example, we used weighted sum of building related information such as the number of razed buildings, critically damaged buildings, mildly damaged buildings etc. We designed 10 such maps by combining different aspects of the IBB results and present three examples below.

Figure 4.3 directly corresponds to the population ([AtlasBig, 2023]). The values

are normalized between 0 and 1. We denote these as follows

$$z_{ij} = g_p \tag{4.1}$$

where z_{ij} is the disaster severity at $(i, j) \in Neighborhood\, g$, g_p is the total population of $Neighborhood\, g$.

Figure 4.4: Istanbul Map 2 - Building Status

We also take into account the building factors presented in [The Istanbul Metropolitan Municipality, 2021] to generate another Istanbul map. In this map, presented in Figure 4.4, the formula we use as follows. Again, the values reflect the disaster severities of corresponding locations, normalized between 0 and 1.

$$z_{ij} = \alpha g_{bc} + \beta g_{bh} + \theta g_{bm} + \omega g_{bl} \tag{4.2}$$

where z_{ij} is the disaster severity at $(i, j) \in Neighborhood\, g$, $g_{bc}, g_{bh}, g_{bm}, g_{bl}$ are the number of buildings damaged in $Neighborhood\, g$ with the levels critical, high, medium and low, respectively [The Istanbul Metropolitan Municipality, 2021]. $\alpha, \beta, \theta, \omega$ are 0.4, 0.3, 0.2 and 0.1, respectively. These parameters are adjustable.

Figure 4.5, represents another map based on how population is affected and is created as follows:

$$z_{ij} = \alpha g_d + \beta g_i + \theta g_{is} \tag{4.3}$$

where z_{ij} is the disaster severity at $(i, j) \in Neighborhood\, g$, g_d, g_i, g_{is} are the number of people in $Neighborhood\, g$ dead, injured and seriously injured, respectively

Figure 4.5: Istanbul Map 3 - Human related factors

[The Istanbul Metropolitan Municipality, 2021]. We pick the weights $\alpha = 0.5, \beta = 0.3, \theta = 0.3$.

Figure 4.6: Istanbul Map 4 - Mixed map

Figure 4.6 shows a map that combines more varied information and is created as below:

$$z_{ij} = \alpha g_d + \beta g_i + \theta g_{is} + \omega g_{ms} + \kappa g_{bc} + \zeta g_{bh} \tag{4.4}$$

where z_{ij} is the disaster severity at $(i, j) \in Neighborhood\, g$, g_d, g_i, g_{is}, g_{ms} are the number of people in $Neighborhood\, g$ dead, injured and seriously injured, and medically supported, respectively. The building damage assessments, critical (g_{bc}) and high (g_{bh}) are also included. The weights $\alpha, \beta, \theta, \omega, \kappa, \zeta$ are taken as 0.3, 0.2, 0.1, 0.1, 0.2 and 0.1 respectively.

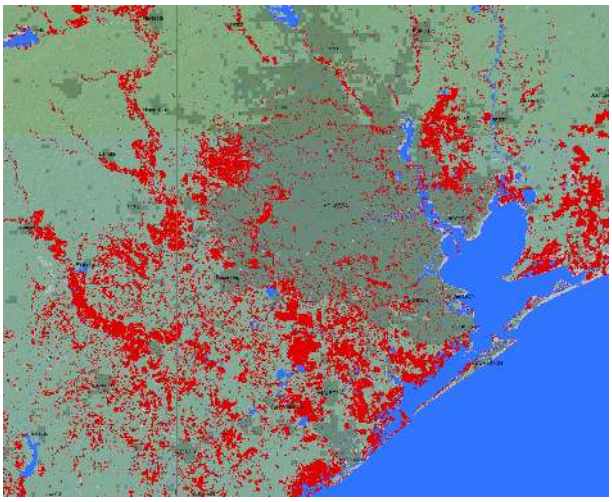

Figure 4.7: Flood Disaster ([The DFO Flood Observatory, 2023])

4.1.3 Other Maps

In addition to the toy and the Istanbul maps, we utilize other sources to further evaluate generalization.

On such map it presented in Figure 4.7, retrieved from [The DFO Flood Observatory, 2023]. This map represents previous and current flooding as well as the current water level. We utilize the latest flooding information, given by the color red.

Figure 4.8 shows a fire-disaster map, retrieved online from [Fire Information for Resource Allocation System, FIRMS, 2023]. Red parts indicate the regions that are on fire. We use this color as our disaster severiy input.

The aim of using these images is to measure further generalization performance of our models.

4.2 Training and Model Details

We use Deep Q-Learning [Mnih et al., 2013] to train our RL agents with Polyak averaging. We use a constant number of initial rescue units for all the training sets. We interleave maps during the training phase.

Figure 4.8: Fire Disaster ([Fire Information for Resource Allocation System, FIRMS, 2023])

4.2.1 Training and Implementation Details

In our formulation, an episode is considered as a complete allocation of rescue squads to locations, which means an episode ends when all the rescue squads are assigned to regional points. We have an experience replay buffer that stores state, action, next state and reward information of an episode, with size 20000. Our agent approximates the Q-values that are given as follows;

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_{a} Q(s',a) - Q(s,a)] \tag{4.5}$$

The equation states that the Q-value yielded from being at state s and performing action a is the immediate reward plus the highest Q-value possible from the next state. Gamma is the discount factor which implies the contribution of future rewards. In our problem, the discount factor is set to 1 to maximize the cumulative rewards. Action is a (i, j) coordinate, where $i \in [0, M), j \in [0, N)$, and M and N are the width and the height values of the map. During training, epsilon-greedy action selection is applied with epsilon as 1 decayed to 0.2. As indicated, our model

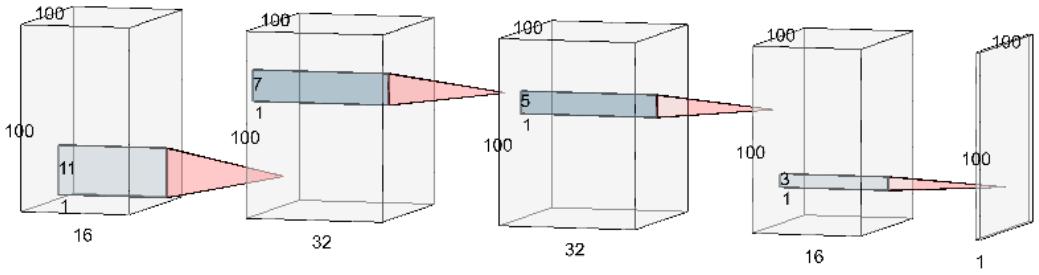

Figure 4.9: Network Architecture

is a pure CNN based deep-q-network, depicted in Figure 4.9. It takes the $M \times N$ environment snapshot, the state, and outputs $M \times N$ q-values. The fully CNN architecture is selected since both the states and actions should consider geometrically close neighbors. However, looking at more neighbors may smooth out the appropriate response. The kernel size of each layer is one of the most important aspect that dictates the extent of neighborhood relationships. We reduce the kernel size from layer-to-layer to specialize on smaller and smaller regions.

The batch size is set to 256, and learning rate, α, for q-learning is set to 0.001. All the hidden-layer activation functions are selected as Leaky ReLU. We also utilize a target network during Q-learning and use the following formula to update it:

$$w_t = \tau w_q + (1 - \tau) w_t \tag{4.6}$$

where w_t represent the weights of the target network, and w_q represents the weights of the main network. Target network is updated in every 10 episodes with τ as 0.001.

After having this setup, we connect our trainer application with disaster simulator. We deploy training disaster severity maps for the simulation. We utilize 100 rescue units. After that, the simulator launches the environment and publishes the environment information to the decision maker. The agent makes the initial rescue squad distribution with epsilon-greedy action selection, then, the simulator starts dynamic environment to observe the consequences of initial distribution. Rescue

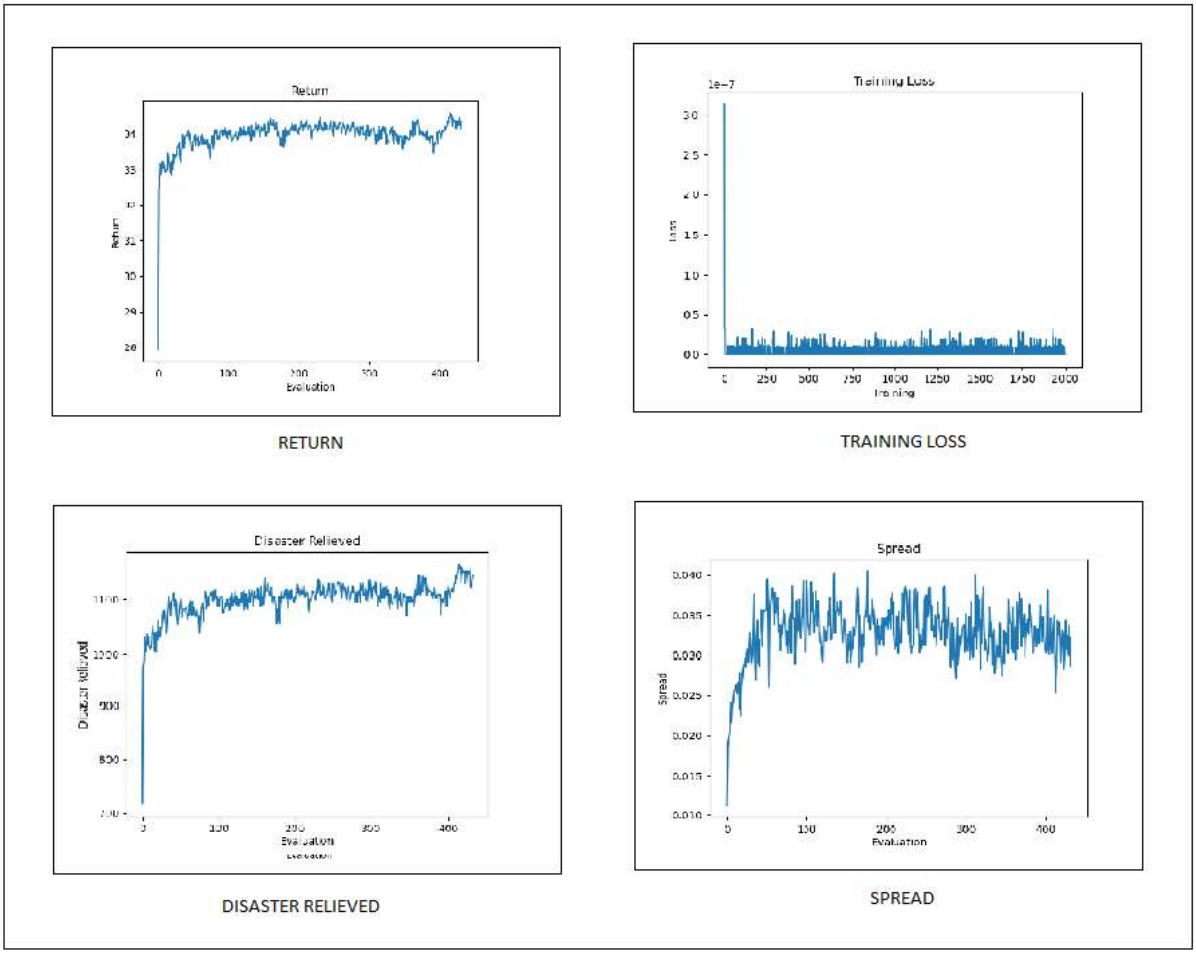

Figure 4.10: Return, loss, amount of disaster relieved and spread during training. Training loss is tracked for 2000 episodes and the other ones are evaluated every 5 episodes greedily.

teams start working for T time steps, and finally the simulator reports the disaster relieved J_{dr}, initial distribution spread J_{spr}, number of unique visited affected locations L and initial decision duration D. Having these information, the agent learns and demonstrate itself.

4.3 Learning Curves

As mentioned before, we use two Istanbul maps and one toy map during training. In the interest of brevity, we present the RARL agents learning curves, which are given in. Figure 4.10. We present return, loss, amount of disaster relieved and spread

during training. The loss is calculated at every episode. We greedily evaluate the policy every 5 episodes and calculate the return, amount of disaster relieved and spread. The results show that the RL training has converged.

4.4 Qualitative Analysis

We interpret the agent performance behaviorally first, by running them on toy maps. These maps allow us to assess whether a trained model distribute the rescue teams effectively and fairly. We expect to have the model notice important places if it embraces qualitatively effective behavior. We also expect to see that a model fairly distributes the resources along the map.

4.4.1 Case Study - 1: Effectiveness Testing

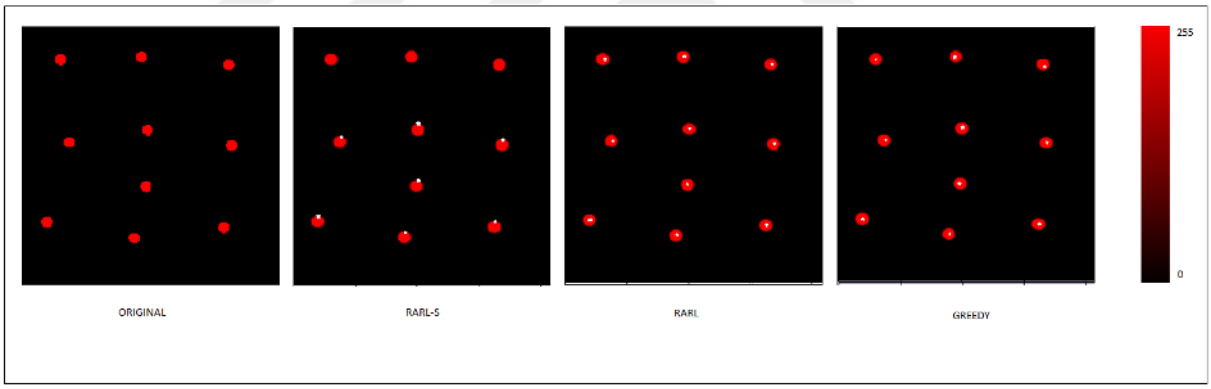

Figure 4.11: Toy map 1 - point indicators

	Spread	Disaster Relieved	Location Visited	Duration
RARL-S	0.037	95.3	96	0.61
RARL	0.087	119.12	120	0.61
Greedy	0.086	118.83	118	94.3

Table 4.1: Quantitative results of Toy Map 1

In this test (Figure 4.11), we evaluate the model in terms of effectivenes. As described in Part 4.1, there are randomly located 10 circles in the map with high disaster severity. Each place contains d amount of disaster severity in total which one rescue squad is enough to relieve. We set the initial number of rescue units as 10. Thus, a good method should allocate one rescue unit to one circle. Note that the map is 100×100, resulting in 10000 potential actions per decision time-step.

As seen in Figure 4.11, our Dense RL agent, RARL, and the greedy agent assigned rescue units to all 10 important areas. We also see that RARL-S has only allocated to 7 locations. The Table 4.1 shows the quantitative results which reflect the qualitative observations. The success of the greedy agent was expected. The success of the RARL agent shows good training. On the other hand RARL-S is either under-trained or is not generalizing well. The disaster input is relatively distinct from all the training inputs. Lastly, the greedy agent takes a lot longer to compute a complete allocation.

4.4.2 Case Study - 2: Fairness Testing

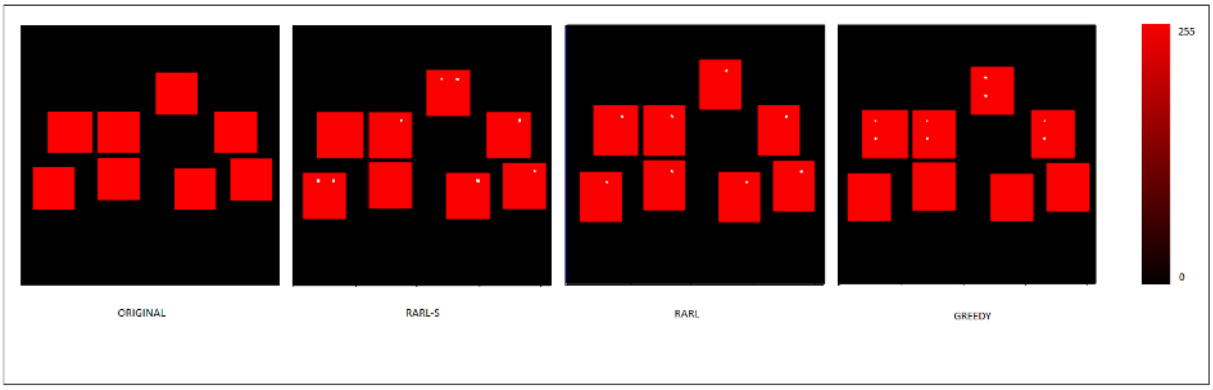

Figure 4.12: Toy map 2 - identical squares randomly ordered

In this test (Figure 4.12), we evaluate the model in terms of fairness. As described in Chapter 4.1, there are randomly located identical 8 squares in the image. Each square is a distinct region and reflects locality. Main aim of considering spread is to make the model catch those localities and make the distribution as much fairly as it

	Spread	Disaster Relieved	Locations Visited	Duration
RARL-S	0.078	95.99	96	0.51
RARL	0.057	95.99	96	0.52
Greedy	0.034	95.99	96	96.81

Table 4.2: Quantitative results of Toy Map 2

can. We utilized 8 rescue units for 8 distinct squares. The map is again 100×100, which means there are 10000 locations and same amount of actions that agents can take. We expect agents to assign one rescue unit to one square.

As seen in Figure 4.12, RARL agent have assigned rescue squads to all 8 distinct randomly placed identical squares whereas greedy agent have assigned to 4 of them and RARL-S have assigned to 6 of them. Greedy has no notion of spread so its failure was expected and the scenario is chosen to highlight this. On the other hand, RARL-S has higher spread than RARL, as presented in Table 4.2, even though it did not catch all the parts. Notice that the RARL-S agent placed rescue units closer to the edges of the map. It is exploiting the spread metric which relies on covariance. In hindisght, we can say that our current spread metric is slightly flawed. As before, the greedy agent takes a lot longer to compute a complete allocation.

4.5 Quantitative Analysis

Istanbul, a well-known city in Turkey, has a high potential of experiencing a destructive earthquake in the near future [Şimşek and Gunduz, 2021]. As mentioned in Chapter 4.1, we create 10 Istanbul maps and use 8 of them for testing.

The aggregate results for each method and for a variety of initial rescue units is given in Table 4.3. We can draw the following conclusions:

- The RARL agent relieves the most disaster among all the methods and RARL-S the least. This is expected as the per time-step reward is geared more towards disaster relieved. Since the greedy agent only has the transition function to

rely on, it doesn't spread the rescue units as well, leading to inefficiencies. The gap widens as the number of initial rescue units are increased.

- The RARL-S agent has the best spread and the greedy has the least. The per-step reward dominates the spread reward for the RARL agent, which leads it to pay less attention to this. The greedy agent has no direct notion of spread.

- The greedy agent takes two order of magnitude longer to come up with a complete allocation compared to the trained agents.

- All the methods are robust to the number of rescue units. We can conclude that the state space representation and the designed transition function work well to induce this.

- RARL-S agent on average leads to most locations visited. However, for 100 and 200 rescue units, this is not the case. This maybe due to the flawed spread metric; this agent is trying to allocate rescue units away from the "center" and not necessarily create a uniform spread and as a result end up clustering the units towards the edges as their number increases.

Next, we look closer at four Istanbul maps as case studies and to present further analysis. All of these maps are 100×100, we use 100 initial rescue units, we run the simulator for 72 time-steps, each rescue unit relieves 0.15 disaster per simulation time-step. Note that the simulator time-step and allocation time-step are different things!

4.5.1 Case Study - 3: Population Scenario

In this case study, we use the population based Istanbul map (4.3). The total disaster severity in this map is 684.26.

Figure 4.13 shows the complete allocation of RARL-S, RARL and Greedy agents on the population based Istanbul map. RARL-S has the most spread distribution

	SPREAD			DISASTER RELIEVED			LOCATION VISITED			DURATION		
#RS	**Rs**	**R**	**G**	**Rs**	**R**	**G**	**Rs**	**R**	**G**	**Rs**	**R**	**G**
10	0.036	0.004	0.001	100.5	109.8	103.8	254.7	187.2	199.7	0.7	0.7	100.2
20	0.045	0.008	0.004	184.7	194.8	185.0	547.6	406.6	412.0	1.4	1.3	201.9
30	0.045	0.013	0.009	251.2	264.7	258.7	785.4	623.8	634.4	2.1	2.0	298.8
40	0.045	0.014	0.012	320.2	322.5	314.0	1062.1	818.9	834.3	2.9	2.7	399.4
50	0.048	0.015	0.013	381.4	384.5	372.3	1330.3	1062.1	1062.9	3.6	3.4	502.7
70	0.045	0.020	0.018	460.4	500.2	473.0	1713.7	1592.1	1549.7	5.0	5.1	763.4
90	0.052	0.024	0.021	515.3	566.1	536.6	2014.2	1933.5	1918.8	6.5	6.9	1020.5
100	0.052	0.028	0.025	530.4	645.0	612.9	2117.2	2443.9	2429.4	7.2	10.3	1537.9
200	0.053	0.033	0.030	573.2	698.1	657.4	2362.6	2811.9	2777.9	14.8	13.9	2047.1
Avg	**0.0468**	**0.0177**	**0.0148**	**368.6**	**409.5**	**390.4**	**1354.2**	**1320.0**	**1313.2**	**4.9**	**5.2**	**763.5**

Table 4.3: Evaluation results on the 8 Istanbul Test Maps. Results are composed of average values of different testing maps with respect to various number of rescue units. Agents are Rs: RARL-S, R: RARL, G: Greedy.

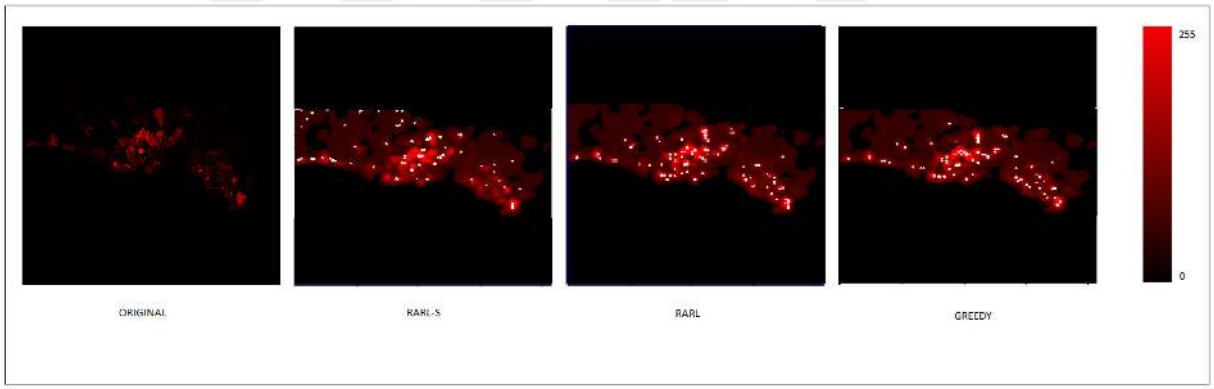

Figure 4.13: Istanbul Map 1 - Population based illustration

	Spread	**Disaster Relieved**	**Location Visited**	**Duration**
RARL-S	0.032501	620.0924	2411	7.08032
RARL	0.016346	624.099	2544	6.365986
GREEDY	0.012315	619.2521	2522	954.5744

Table 4.4: Quantitative results of population based Istanbul map

but it misses several high concentration areas. On the end, the greedy agent mainly focused on the regions with high concentration. RARL not only focused on those remarkable areas, but it also considered distribution spread. RARL agent is close to greeddy but it manages to divert resources to several parts of the map where greedy missed.

These qualitative results are backed by the quantiative ones as seen in the Table 4.4. The RL agents have greater spread compared to greedy. RARL has the greatest disaster relieved value compared to other agents but by a small margin. RARL-S relieved more disaster than greedy. The greedy agent's allocation resulted in wasted opportunity as it concentrated the rescue units, which affected both spread and disaster relieved. As before, the greedy agent took a lot longer than the RL agents for a complete allocation.

4.5.2 Case Study - 4: Buildings Scenario

In this case study, we use building status based Istanbul map (Figure 4.4). The total disaster severity in this map is 849.16. We utilize 100 rescue units.

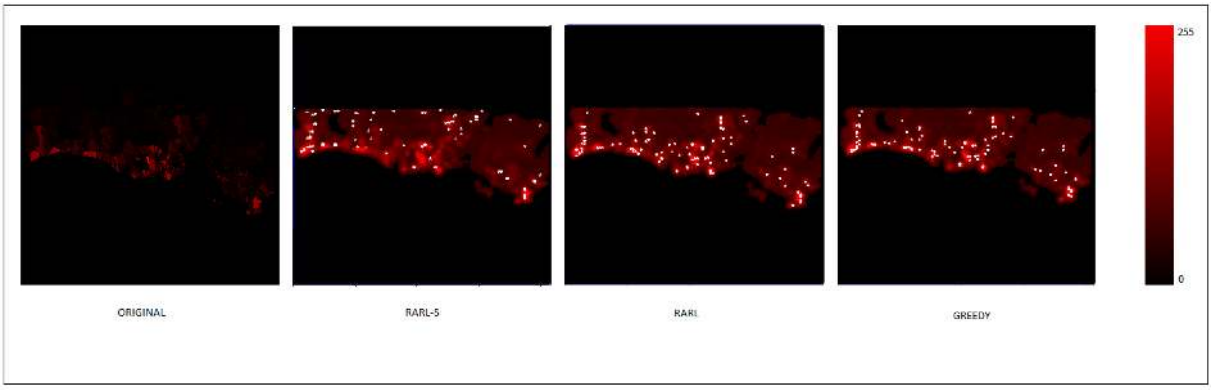

Figure 4.14: Istanbul Map 2 - building based illustration

Figure 4.14 shows the complete allocation of RARL-S, RARL and Greedy agents on the building based Istanbul map. Again, greedy mainly focused on high concentration areas while RARL-S tried to increase the spread. However, the behavior of RARL-S on increasing spread caused to decrease the amount of disaster relieved

	Spread	Disaster Relieved	Locations Visited	Duration
RARL-S	0.06766	687.0208	2225	7.8244
RARL	0.035412	784.9394	2285	6.369084
GREEDY	0.033019	777.6326	2319	951.8884

Table 4.5: Quantitative results of building based Istanbul map

due to missing those high concentration areas and limited amount of simulation time steps. RARL agent performed similar to the greedy agent with a slight edge on the metrics. These observations are supported by the Table 4.5 as well.

4.5.3 Case Study - 5: Survivals Scenario

In this case study, we use human factors based Istanbul map (4.5). The total disaster severity in this map is 622.17.

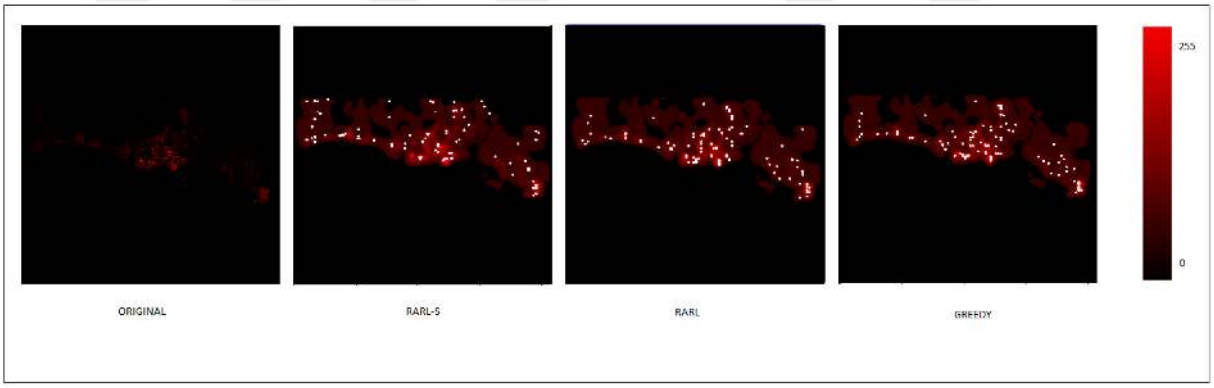

Figure 4.15: Istanbul Map 3 - illustration of human factors

Figure 4.15 shows the complete allocation of RARL-S, RARL and Greedy agents on the human-factors based Istanbul map. The conclusions are more or less the same, other than the fact that greedy is more behind the RL agents. Furthermore, it can be argued that the RARL-S agent may even be the preferred agent based on the Table 4.6 results. However, it sent too many rescue units to the top left of the map to try to maximize the covariance. Qualitatively speaking, less would have been

	Spread	Disaster Relieved	Locations Visited	Duration
RARL-S	0.04843	565.7004	2250	7.089
RARL	0.024867	567.6359	2471	7.897889
GREEDY	0.020159	550.3484	2584	1170.797

Table 4.6: Quantitative results of human factors based-Istanbul map

preferred.

This actually highlights an important point to use these agents as decision aids. The human decision maker can mix and match the allocations. The speed at which the RL agents work easily allow this.

4.5.4 Case Study - 6: Multi-Factor Scenario

In this case study, we use Mixed Istanbul map (Figure 4.6). The total disaster severity in this map is 690.35.

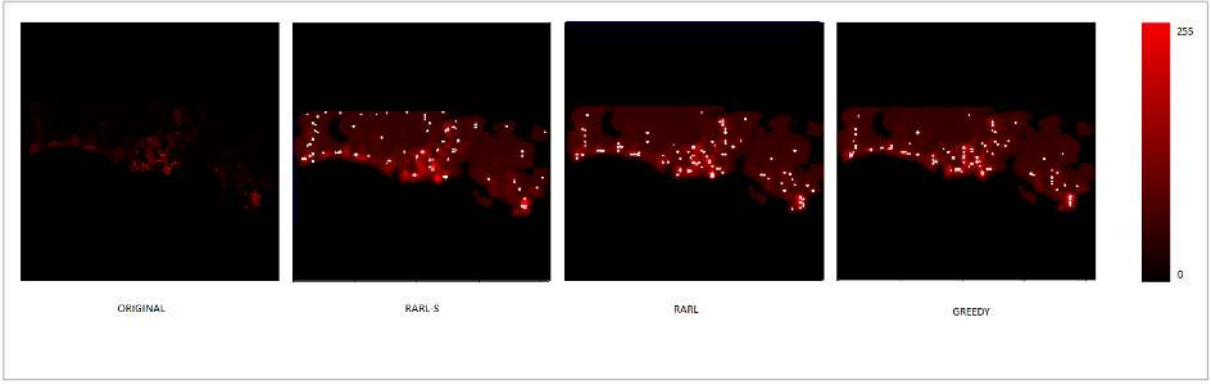

Figure 4.16: Istanbul Map 4 - illustration of multi-factor disaster severities.

Figure 4.15 shows the complete allocation of RARL-S, RARL and Greedy agents on the multi-factor based Istanbul map. The distributions are similar to the other cases where RARL-S has the most spread, and RARL and greedy being similar with an edge on RARL. The quantitative metrics, presented in Table 4.7, shows that indeed RARL has an edge, but a relatively significant one, on both spread and

	Spread	Disaster Relieved	Locations Visited	Duration
RARL-S	0.04795	636.1577	2440	7.15512
RARL	0.023699	641.1964	2628	6.836084
GREEDY	0.01827	619.7174	2531	1023.645

Table 4.7: Quantitative results of multi-factor Istanbul map

disaster relieved. RARL-S is also doing well albeit with certain unneeded allocations along the top of the map while missing certain parts in the lower sections.

4.6 Further Generalization Analysis

We also use other disaster severity map representatives to make inferences further on generalization performance. We found them online and converted them to disaster severity maps. As indicated, one of our main goals is to have a context-independent solution applicable to any case.

4.6.1 Case Study - 7: Flood Scenario

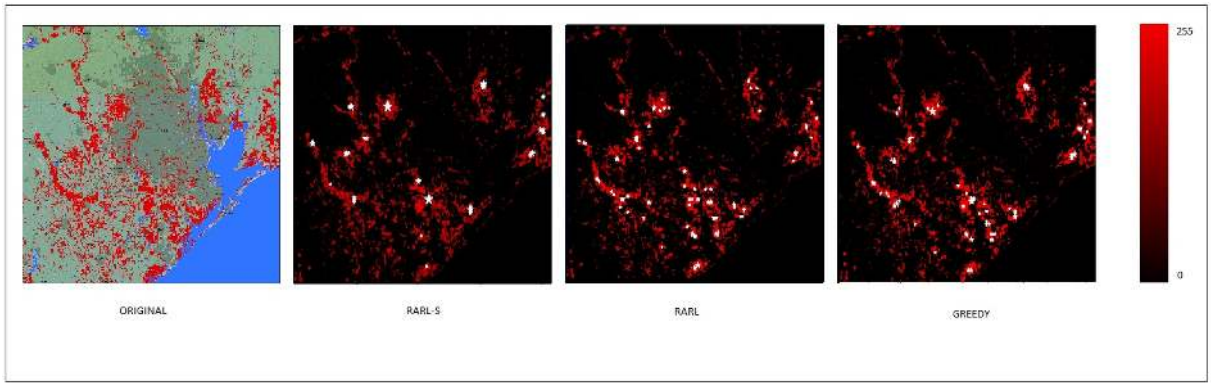

Figure 4.17: Flood Disaster [Fire Information for Resource Allocation System, FIRMS, 2023]

In Figure 4.17, images are the visualizations of RARL-S, RARL and Greedy algorithms' initial assignments for Flood Disaster. As seen in the Table 4.8, RL

	Spread	Disaster Relieved	Locations Visited	Duration
RARL-S	0.04972	499.8451	514	6.68280
RARL	0.03927	534.00120	526	6.47637
GREEDY	0.03661	527.4046	517	990.2693

Table 4.8: Quantitative results of Flood Disaster

solutions RARL-S and RARL have greater spread values compared to greedy. RARL have the greatest disaster relieved value compared to other agents whereas greedy relieved disaster more than RARL-S.

4.6.2 Case Study - 8: Fire Scenario

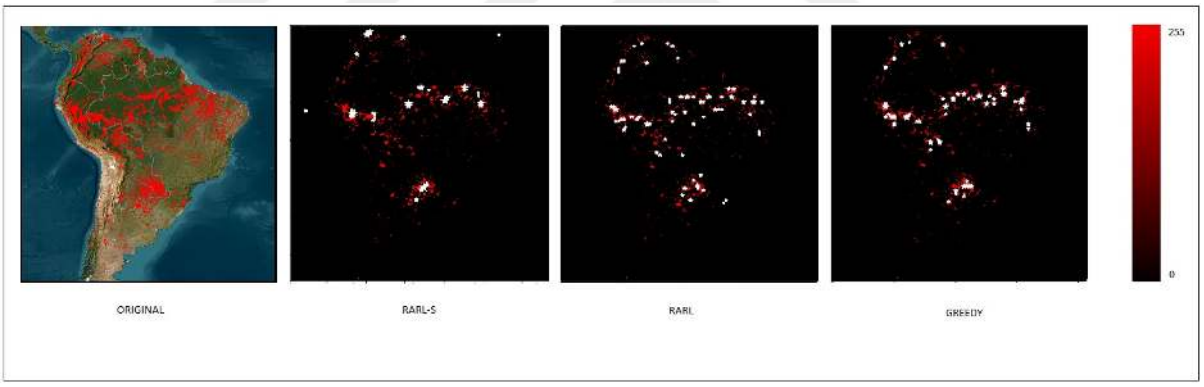

Figure 4.18: Fire Disaster

	Spread	Disaster Relieved	Locations Visited	Duration
RARL-S	0.01581	405.3825	576	7.2149
RARL	0.01331	429.6364	661	6.1788
GREEDY	0.01119	411.9419	670	1015.657

Table 4.9: Quantitative results of Fire Disaster

In Figure 4.18, images are the visualizations of RARL-S, RARL and Greedy algorithms' initial assignments for Fire Disaster. As seen in the Table 4.9, RL solutions RARL-S and RARL have greater spread values compared to greedy. RARL have the greatest disaster relieved value compared to other agents whereas greedy relieved disaster more than RARL-S.

4.7 Scaling Analysis

Our approach makes use of convolutional neural networks and supports arbitrary cost definitions. This contributes the model to perform independently from representations of a formulation. We resized the disaster map as 50x50 and 200x200 to interpret the performance on different sized state and action spaces. For following case studies, we trained two different RARL agents for 50x50 and 200x200 map representations. Thus, we only present the results of RARL and greedy agent.

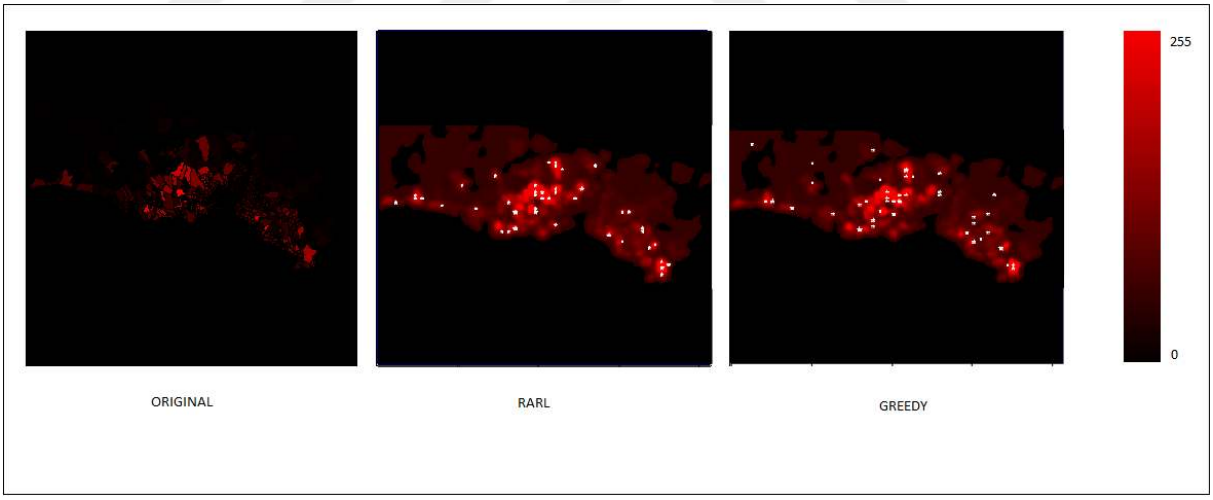

Figure 4.19: RARL and Greedy results on 50x50 Istanbul Population Map

	Spread	**Disaster Relieved**	**Location Visited**	**Duration**
RARL	0.0215	214.40	639	2.77
Greedy	0.0204	211.65	655	134.12

Table 4.10: Quantitative results on 50x50 map

We hold a case study for 50x50 map. We utilized 50 rescue units and kept anything else related to the simulation setup and evaluation details same. In Figure 4.19, visual distributions of RARL and greedy is presented. Similar to previous case studies, RARL kept its consistency. It notices the high-concentrated regions while it is expanding the distribution scope. On the other hand, greedy agent notices the high-concentration regions without considering spread. As a result, in Table 4.10, RARL distribution received higher spread value. However, greedy also notices some regions which are far away from cumulative disaster centers by qualitatively inferring a fairer behavior. The consequent benefit of this issue is reflected on the number of location visited. In spite of a small difference, these two agents relieved almost the same amount of disaster.

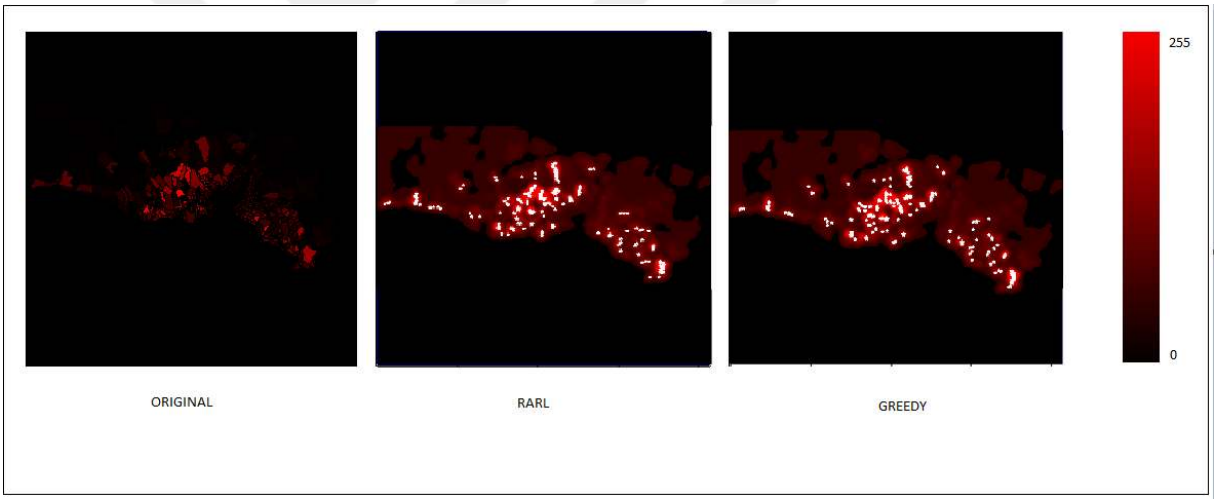

Figure 4.20: RARL and Greedy results on 200x200 Istanbul Population Map

	Spread	**Disaster Relieved**	**Location Visited**	**Duration**
RARL	0.01288	1715.635512	3764	15.38
Greedy	0.01165	1693.50464	3952	8114.50

Table 4.11: Quantitative results on 200x200 map

We also hold a case study for 200x200 map. We utilized 200 rescue units and kept anything else related to the simulation setup and evaluation details same. In

Figure 4.20, visual distributions of RARL and greedy is presented. Qualitatively interpretting, the distributions seem very similar. There are minor differences on neighboring local regions that bring the RARL a little bit prominence on quantitative results. These minor differences infer that RARL intuitively learnt to distribute rescue units to cause minimal conflicts on worked regions. The outcome of the underlined strategy are reflected on the quantitative results given in Table 4.11.

4.7.1 Per Time-Step Returns

The main objectives are the disaster-relieved and spread which we have been using thus far. However, we also present the per time-step returns of each method, averaged over test maps versus number of rescue unit in Figure 4.21. It is no surprise that the greedy agent performs the best based for this metric on our discussion in Chapter 3.6. The interesting aspect it that the RARL agent can closely follow the Greedy agent while also utilizing the terminal cost. This shows that, under the assumptions of this work, the RL agent can be within close proximity of a classical approach and get the results much faster.

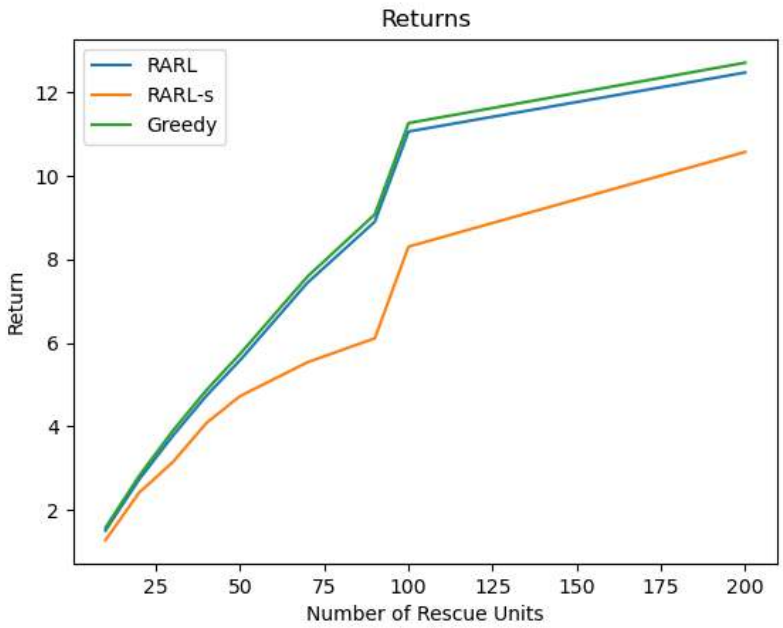

Figure 4.21: Test map total returns of each agent versus number of rescue units.

4.8 Conclusion

In this section, we presented our experimental setup and results. We described our disaster scenarios, the disaster severity maps. We also presented our model and training setup in detail including parameters and network architecture. Given the results, we interpret the model performance qualitatively and quantitatively. In addition, we tested our agents on other disaster severity maps we use to analyze generalization performance.

Evaluation results are in two different perspectives, qualitative and quantitative. Considering the setup, environment is represented with large state and action spaces in any case studies. The world is 100x100 matrix which has 10000 locations and same amount of distinct actions. While handling the large state and action spaces, relevant and irrelevant objectives have also been in considerations. The q-network we use to solve this large-scale problem, on the contrary, is small, having 5 convolutional layers and around 60000 parameters in total. Furthermore, the RL agents have been trained only on 3 different maps and tested successfully on at least 14 maps. In addition, both the architecture and infrastructure also allow us to evaluate all the actions simultaneously by applying parallel computation.

As seen in Part 4.4, RL solutions performs better than greedy with respect to our performance metrics. Interpreting on effectiveness, Dense RL solution RARL have always performed more effectively than greedy. In addition to successfully noticing important regions on the map, optimized decisions lead the amount of disaster relieved increased (Figure 4.11). Spread values were neither directly represented in state nor became an input to the model indirectly. However, model learnt to innate it in their decisions. In Figures 4.12, the models were expected to notice all those squares and assign at least one rescue squad to each of them. RARL has behaved as expected by allocating at least one rescue squad to each squares. To sum up, it is proven that RL solutions satisfy our fairness metric as well. Decision making duration of greedy agent is along 100 seconds which is 160 times greater than that of RL solutions, which shows RL solutions are more responsive.

In Part 4.5, models were evaluated in larger-scale environments. We make the

similar inferences with part 4.4. In addition to those, we also capture that RL solutions noticed details. We observe that RARL allocates resources to the locations that greedy missed. These locations contain less disaster severities compared to others, yet greedy's behavior violates the fairness since these areas could never be supported. On the contrary, RARL notices those areas and assigns at least one rescue squad on closer areas. In any case, we qualitatively and quantitatively measure and see extensive coverage of RARL's decisions.

Overall, the RARL agent performed either on par or better than the other agents in multiple different scenarios, both qualitatively and quantatively. Only RARL-S has an edge in spread which is due to in part to the dominating effect of the per time-step rewards towards disaster relieved. However, our current spread definition is flawed and qualitative look at the allocations show that RARL has promising fairness.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this work, we presented a Reinforcement Learning (RL) based resource allocation approach for initial disaster response. We are targeting large regions with multiple centers of devastation. We assume that a disaster severity assessment is already made and the rescue units are going to be dispatched one-by-one as a sequential decision making based on the three criteria effectiveness, fairness and quickness.

We developed a simple disaster simulator and designed disaster severity maps that reflect our distribution ideals. We use a 2D map representation of the affected area, analogous to an image, as our state space, selection of a location on the map as our action space and we carefully designed our transition and reward functions.

We utilize Deep-Q-Learning to train two different agents, one with just the terminal reward, and another one with dense reward in addition to terminal reward. We use pure convolutional neural networks (CNNs) from the state input to action value output to approximate the Q-values. This technique we use allows us to handle large-scale representations. We trained these models on three disaster severity maps and evaluated the performance on unseen test maps. We additionally developed a greedy agent that goes over each location to calculate per-step rewards. The results show greedy algorithm runs slower than RL agents due to its nature. In addition to this, qualitative and quantitative analysis shows that greedy agent does not care about the spread as well. RARL-S agent deficiently considers high concentration regions, so it falls short on the amount of disaster relieved whereas greedy falls short on the spread. Only the RARL performs on par or better, and satisfies the performance metrics.

Simplicity of the simulation, complexity of the environment and difficulties in

representing a few components of the disaster have been our biggest limitations during our study. We plan to expand our study with the items as stated in Part 5.2.

5.2 Future Work

Main aim of this study is to show applicability of reinforcement learning based artificial intelligence solutions on large-scale resource allocation problems. In this study, initial disaster response for disaster management has been the challenge, represented in large-scale context. Besides, the introduced approach and infrastructure we developed open up new research directions.

We made some simplifying assumptions and kept the environment representation as simple as possible during research activities. City, neighborhood and many other component of our formulation are available, yet they only exist conceptually. For instance, even though we designed our disaster scenarios upon neighborhood-based consideration, the map is a flat surface that does not incorporate the relationships between regional areas. In other words, neighborhood is only conceptually available, but not represented in the environment. In addition, we did not utilized road networks or links between locations in the map. These simplifications lead us to only use visual distance in the image. Furthermore, we did not take arrival cost into consideration due to underlined simplification. Rescue squads show up the map as soon as the initial allocation gets finalized. Therefore, first item of our future work plan is to represent the the disaster map upon neighborhood-based formulation, containing road networks or links among the regions. This will lead us to work on more complex, larger and realistic studies in addition to providing us more structural environment representation.

We focused on static disaster and only one type of rescue unit. There is no diversity such as firefighters, ambulances, polices, etc. We did not also consider disaster expansion or damage types due to disaster being static. Thus, we did not need those diverse components in our environment, meaning that, these are also the consequences of simplifying assumptions. As a second future work item, we plan to

represent dynamic disaster environment that contain damage types, rescue squad types which each type is responsible for one type of disaster damage and following possible impacts of disaster such as fire after an earthquake.

In this study, we did not include survivals and human driven factors during the disaster. We designed disaster severity maps reflecting support-demanding regions as if rescue squads are for serving for this purpose. However, survivals and people must be comprehensively involved in such studies since the action rescue is required to be interpreted in survival-oriented perspective. In addition, survivals affect the crisis as well, which creates another dynamism. Mobility, diversity of survival needs, media and social media are the most important factors that are deserved to be considered. Therefore, one of our our next future work items is to involve human driven factors in our environment representation. It will not only help us represent the real world environment, but it also serve for dynamism and stochasticity of the world.

One of our future work items is disaster severity prediction. In our study, as mentioned, we have disaster density values calculated using IBB data and other statistical information found online. However, we plan to work on disaster severity prediction to design our environment.

Including the work items above, our problem space will become larger and larger. To overcome the challanging issues we may encounter, we also plan to design a new network architecture that represent all the dynamics of our formulation. We also may make use of other algorithms such as graph neural networks since the environment is going to become more complex and have more relations and connection-based structures. Furthermore, we may need multiple neural networks for each aspects of our formulation to distribute the responsibility of environment interpretations.

BIBLIOGRAPHY

[Aihara et al., 2019] Aihara, N., Adachi, K., Takyu, O., Ohta, M., and Fujii, T. (2019). Q-learning aided resource allocation and environment recognition in lorawan with csma/ca. *IEEE Access*, 7:152126–152137.

[AtlasBig, 2023] AtlasBig (2023). Istanbul'un mahalle bazinda nüfus yogunlugu haritasi. data retrieved from atlasbig.com.tr, `https://atlasbig.com.tr/istanbul-mahalleleri-nufus-yogunlugu`.

[Fiedrich et al., 2000] Fiedrich, F., Gehbauer, F., and Rickers, U. (2000). Optimized resource allocation for emergency response after earthquake disasters. *Safety Science*, 35(1):41–57.

[Fire Information for Resource Allocation System, FIRMS, 2023] Fire Information for Resource Allocation System, FIRMS (2023). Fire information for resource allocation system. data retrieved from NASA Fire Information for Resource Allocation System, `https://firms.modaps.eosdis.nasa.gov/`.

[Gong and Batta, 2007] Gong, Q. and Batta, R. (2007). Allocation and reallocation of ambulances to casualty clusters in a disaster relief operation. *IIE Transactions*, 39(1):27–39.

[Hu et al., 2016] Hu, C., Liu, X., and Hua, Y. (2016). A bi-objective robust model for emergency resource allocation under uncertainty. *International Journal of Production Research*, 54(24):7421–7438.

[Jiang and Sheng, 2009] Jiang, C. and Sheng, Z. (2009). Case-based reinforcement learning for dynamic inventory control in a multi-agent supply-chain system. *Expert Systems with Applications*, 36(3, Part 2):6520–6526.

[Jiang et al., 2018] Jiang, Z., Fan, W., Liu, W., Zhu, B., and Gu, J. (2018). Reinforcement learning approach for coordinated passenger inflow control of urban rail transit in peak hours. *Transportation Research Part C: Emerging Technologies*, 88:1–16.

[Jiang et al., 2019] Jiang, Z., Gu, J., Fan, W., Liu, W., and Zhu, B. (2019). Q-learning approach to coordinated optimization of passenger inflow control with train skip-stopping on a urban rail transit line. *Computers & Industrial Engineering*, 127:1131–1142.

[Kara and Dogan, 2018] Kara, A. and Dogan, I. (2018). Reinforcement learning approaches for specifying ordering policies of perishable inventory systems. *Expert Systems with Applications*, 91:150–158.

[Kwon et al., 2008] Kwon, I.-H., Kim, C. O., Jun, J., and Lee, J. H. (2008). Case-based myopic reinforcement learning for satisfying target service level in supply chain. *Expert Systems with Applications*, 35(1):389–397.

[Lee and Lee, 2021] Lee, H.-R. and Lee, T. (2021). Multi-agent reinforcement learning algorithm to solve a partially-observable multi-agent problem in disaster response. *European Journal of Operational Research*, 291(1):296–308.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518:529–533.

[Nadi and Edrisi, 2017] Nadi, A. and Edrisi, A. (2017). Adaptive multi-agent relief assessment and emergency response. *International Journal of Disaster Risk Reduction*, 24:12–23.

[Nadi and Edrissi, 2016] Nadi, A. and Edrissi, A. (2016). A reinforcement learning approach for evaluation of real-time disaster relief demand and network condition. *World Academy of Science, Engineering and Technology, International Journal of Environmental, Chemical, Ecological, Geological and Geophysical Engineering*, 11(1):5–10.

[Ni et al., 2013] Ni, J., Liu, M., Ren, L., and Yang, S. X. (2013). A multiagent q-learning-based optimal allocation approach for urban water resource management system. *IEEE Transactions on Automation Science and Engineering*, 11(1):204–214.

[Pérez-Rodríguez and Holguín-Veras, 2016] Pérez-Rodríguez, N. and Holguín-Veras, J. (2016). Inventory-allocation distribution models for postdisaster humanitarian logistics with explicit consideration of deprivation costs. *Transportation Science*, 50(4):1261–1285.

[Sheu, 2007] Sheu, J.-B. (2007). An emergency logistics distribution approach for quick response to urgent relief demand in disasters. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):687–709. Challenges of Emergency Logistics Management.

[Su et al., 2011] Su, Z.-P., Jiang, J.-G., Liang, C.-Y., and Zhang, G. (2011). Path selection in disaster response management based on q-learning. *International Journal of Automation and Computing*, 8:100–106.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[The DFO Flood Observatory, 2023] The DFO Flood Observatory (2023). Maximum observed flooding. data retrieved from The DFO Flood Ob-

servatory, https://floodobservatory.colorado.edu/Events/2017USA4510/2017USA4510.html.

[The Istanbul Metropolitan Municipality, 2021] The Istanbul Metropolitan Municipality (2021). Analysis of earthquake scenario. data retrieved from The Istanbul Metropolitan Municipality, https://data.ibb.gov.tr/dataset/deprem-senaryosu-analiz-sonuclari/resource/9c3ac492-de4b-4245-b418-7ad3df67a193.

[Tomasini et al., 2009] Tomasini, R., Van Wassenhove, L., and Van Wassenhove, L. (2009). *Humanitarian logistics*. Springer.

[Valente Klaine et al., 2018] Valente Klaine, P., Nadas, J., Souza, R., and Imran, M. (2018). Distributed drone base station positioning for emergency cellular networks using reinforcement learning. *Cognitive Computation*, 10.

[Vengerov, 2007] Vengerov, D. (2007). A reinforcement learning approach to dynamic resource allocation. *Engineering Applications of Artificial Intelligence*, 20(3):383–390.

[Wex et al., 2014] Wex, F., Schryen, G., Feuerriegel, S., and Neumann, D. (2014). Emergency response in natural disaster management: Allocation and scheduling of rescue units. *European Journal of Operational Research*, 235(3):697–708.

[Xiang and Zhuang, 2016] Xiang, Y. and Zhuang, J. (2016). A medical resource allocation model for serving emergency victims with deteriorating health conditions. *Ann. Oper. Res.*, 236:177–196.

[Yan et al., 2020] Yan, L., Mahmud, S., Shen, H., Foutz, N., and Anton, J. (2020). Mobirescue: Reinforcement learning based rescue team dispatching in a flooding disaster. pages 111–121.

[Ye et al., 2019] Ye, H., Li, G. Y., and Juang, B.-H. F. (2019). Deep reinforcement

learning based resource allocation for v2v communications. *IEEE Transactions on Vehicular Technology*, 68(4):3163–3173.

[Yu et al., 2021] Yu, L., Zhang, C., Jiang, J., Yang, H., and Shang, H. (2021). Reinforcement learning approach for resource allocation in humanitarian logistics. *Expert Systems with Applications*, 173:114663.

[Yu et al., 2018] Yu, L., Zhang, C., Yang, H., and Miao, L. (2018). Novel methods for resource allocation in humanitarian logistics considering human suffering. *Computers & Industrial Engineering*, 119:1–20.

[Şimşek and Gunduz, 2021] Şimşek, P. and Gunduz, A. (2021). A big earthquake awaits istanbul: Mini review. *Afet ve Risk Dergisi*, 4.

[Šemrov et al., 2016] Šemrov, D., Marsetič, R., Žura, M., Todorovski, L., and Srdic, A. (2016). Reinforcement learning approach for train rescheduling on a single-track railway. *Transportation Research Part B: Methodological*, 86:250–267.