

Artificial Intelligence for Resource Allocation Tasks



Dissertation zur Erlangung des Doktorgrades
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

vorgelegt von

Niklas Strauss

Munich, den 17.09.2024

Tag der Einreichung: 17.09.2024

Erstgutachter: Prof. Dr. Matthias Schubert
LMU Munich

Zweitgutachter: Prof. Dr. Goce Trajcevski
Iowa State University

Drittgutachter: Prof. Dr. Sebastian Tschitschek
University of Vienna

Vorsitz: Prof. Dr. Eyke Hüllermeier
LMU Munich

Tag der Disputation: 20.12.2024

Eidesstattliche Versicherung

(siehe Promotionsordnung vom 12.07.2011, § 8, Abs. 2 Pkt. 5)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig und ohne unerlaubte Beihilfe angefertigt wurde.

Munich, den 17.09.2024

Niklas Strauss

Contents

Abstract	vii
Zusammenfassung	ix
1 Introduction	1
2 Background on Artificial Intelligence	5
2.1 Deep Learning	5
2.2 Reinforcement Learning	7
3 Resource Allocation Tasks	17
3.1 Discrete Resource Allocation	18
3.1.1 Dynamic Ambulance Redeployment	18
3.1.2 Dynamic Electric Ambulance Redeployment	22
3.2 Stochastic Resource Collection	24
3.3 Continuous Resource Allocation Tasks	27
3.3.1 Overview of Continuous Resource Allocation Tasks	28
3.3.2 Solving Continuous Allocation Tasks	31
4 Overview of Contributions	33
4.1 Spatial-Aware Deep Reinforcement Learning for the Traveling Officer Problem	33
4.2 Reinforcement Learning for Multi-Agent Stochastic Resource Collection . .	34
4.3 A Comparison of Ambulance Redeployment Systems on Real-World Data .	34
4.4 DEAR: Dynamic Electric Ambulance Redeployment	35
4.5 Constrained Portfolio Management Using Action Space Decomposition for Reinforcement Learning	36
4.6 Simplex Decomposition for Portfolio Allocation Constraints in Reinforcement Learning	37
4.7 Autoregressive Policy Optimization for Constrained Allocation Tasks	37

5 Conclusion and Outlook	39
5.1 Limitations and Future Work	39
References	41
Acknowledgements	59
Appendix	61
A Spatial-Aware Deep Reinforcement Learning for the Traveling Officer Problem	61
B Reinforcement Learning for Multi-Agent Stochastic Resource Collection . .	73
C A Comparison of Ambulance Redeployment Systems on Real-World Data .	92
D DEAR: Dynamic Electric Ambulance Redeployment	101
E Constrained Portfolio Management Using Action Space Decomposition for Reinforcement Learning	112
F Simplex Decomposition for Portfolio Allocation Constraints in Reinforcement Learning	129
G Autoregressive Policy Optimization for Constrained Allocation Tasks	138

Abstract

In recent years, innovations in artificial intelligence have led to advances in many different areas, ranging from natural language processing to computer vision. One area that is of special interest are resource allocation tasks. The field of resource allocation includes a diverse range of tasks. Oftentimes, resource allocation tasks are complex sequential decision making problems like portfolio optimization, the traveling officer problem, or redeploying ambulances to base stations. Deep reinforcement learning offers a way to solve these tasks. However, in order to achieve state-of-the-art performance, existing neural network architectures are not sufficient and, in this thesis, we propose several novel architectures. While spatial resource allocation tasks typically have discrete action spaces, some allocation tasks, like portfolio optimization, have continuous action spaces.

The contributions in this thesis handle three types of allocation tasks: discrete resource allocation, resource collection, and continuous resource allocation with allocation constraints.

First, we focus on discrete resource allocation and resource collection. More specifically, we look into different spatial resource allocation tasks, presenting a spatial-aware reinforcement learning-based approach for the traveling officer problem, a prominent resource collection task, achieving state-of-the-art performance. We also propose an approach for multi-agent stochastic resource collection featuring a novel neural network architecture. After that, we focus on dynamic ambulance redeployment. We develop a high-performance event-based simulator, conduct comparisons and benchmarks of existing approaches using real-world data. After that, we are the first to develop an approach for dynamically redeploying electric ambulances. Our method deploys the ambulance to the base station which minimizes the energy deficit.

Afterward, we continue with continuous resource allocation tasks featuring simplex action spaces accompanied with allocation constraints. One of the most important continuous action resource allocation tasks is portfolio optimization, where a portfolio manager allocates its wealth across various assets in each time step over an investment horizon. In practice, these tasks often come with allocation constraints. We develop approaches to efficiently and effectively incorporate one and two allocation constraints using a decomposition of the simplex, allowing us to learn a policy using standard deep reinforcement learning approaches. Our methods never violate the constraints, even during training. Furthermore, we propose an approach capable of handling an arbitrary number of constraints by iteratively sampling actions in each dimension autoregressively,

while utilizing linear programming to compute the action bounds. Our approach remains trainable using existing reinforcement learning algorithms.

Zusammenfassung

In den letzten Jahren haben Innovationen im Bereich der künstlichen Intelligenz zu Fortschritten in vielen verschiedenen Bereichen geführt, die von der Verarbeitung natürlicher Sprache bis hin zum maschinellen Sehen reichen. Ein besonders interessanter Teilbereich sind Ressourcenzuweisungsprobleme, welche ein breites Spektrum an verschiedenen Problemen und Anwendungen umfassen. Häufig handelt es sich bei Ressourcenzuweisungsproblemen um komplexe sequentielle Entscheidungsprobleme wie Portfoliooptimierung, das Problem des reisenden Parkbeamten oder die dynamische Umverteilung von Krankenwagen zu Basisstationen. Deep Reinforcement Learning bietet eine Möglichkeit, diese Probleme zu lösen. Für eine effektive Lösung, reichen die bestehenden neuronalen Netzarchitekturen jedoch nicht aus. Deshalb schlagen in dieser Arbeit wir mehrere neue Architekturen vor. Während räumliche Ressourcenzuweisungsprobleme typischerweise diskrete Aktionsräume aufweisen, haben einige Zuweisungsprobleme, wie zum Beispiel Portfoliooptimierung, kontinuierliche Aktionsräume.

In dieser Arbeit behandeln wir drei Arten von Zuweisungsproblemen: diskrete Ressourcenzuweisung, Ressourcensammlung und kontinuierliche Ressourcenzuweisung mit Allokationsnebenbedingungen.

Zunächst konzentrieren wir uns auf die diskrete Ressourcenzuweisung und Ressourcensammlung. Genauer gesagt untersuchen wir verschiedene räumlichen Ressourcenzuteilungsprobleme und stellen einen auf räumlichem Lernen basierenden Ansatz für das Problem des reisenden Parkbeamten vor, ein prominentes Problem der Ressourcensammlung, bei welchem unser Ansatz eine deutlich bessere Leistung als existierende Verfahren erzielt. Außerdem schlagen wir einen Ansatz für die stochastische Ressourcensammlung mit mehreren Agenten vor, welcher eine neuartige neuronale Netzwerkarchitektur beinhaltet. Danach konzentrieren wir uns auf die dynamische Umverteilung von Krankenwagen. Wir entwickeln zunächst eine leistungsstarke, ereignisbasierte Simulationsumgebung und führen Vergleiche und Benchmarks bestehender Ansätze mit realen Daten durch. Danach entwickeln wir als erste einen Ansatz zur dynamischen Umverteilung von elektrisch angetriebenen Krankenwagen. Unsere Methode verteilt den Krankenwagen zu der Basisstation, an der durch diesen das Energiedefizit am meisten minimiert wird.

Danach befassen wir uns mit kontinuierlichen Ressourcenzuweisung, welche Simplex-Aktionsräume mit Allokationsnebenbedingungen aufweisen. Eines der wichtigsten Probleme der kontinuierlichen Ressourcenzuweisung ist Portfoliooptimierung, bei

der ein Portfoliomanager sein Vermögen in jedem Zeitschritt über einen Anlagehorizont auf verschiedene Vermögenswerte aufteilt. In der Praxis sind diese Anwendungen oft mit Allokationsnebenbedingungen verbunden. Wir entwickeln Ansätze zur effizienten und effektiven Einbeziehung von ein und zwei Allokationsnebenbedingungen unter Verwendung einer Dekomposition des Simplexes, die es uns ermöglicht, eine Strategie mit Hilfe von bestehenden Deep Reinforcement Learning-Ansätzen zu lernen. Unsere Methoden verletzen niemals die Nebenbedingungen, auch nicht während des Trainings. Darüber hinaus schlagen wir einen Ansatz vor, der in der Lage ist, eine beliebige Anzahl von Nebenbedingungen zu handhaben, indem wir iterativ Aktionen in jeder Dimension autoregressiv sampeln. Hierbei nutzen wir lineare Programmierung zur Berechnung der Aktionsgrenzen verwenden. Unser Ansatz kann mit bestehenden Deep Reinforcement Learning-Algorithmen trainiert werden.

Chapter 1

Introduction

In recent years, innovations in artificial intelligence have led to advances in many different areas, including natural language processing [163, 40, 125, 162], computer vision [44, 128, 68], and reinforcement learning [108, 144, 145]. One area that is of particular interest and the focus of this thesis is resource allocation. The term resource allocation refers to the process of allocating limited resources to entities, a task which occurs in a multitude of real-world applications, such as financial economics [102, 103, 172, 143, 46], managing blood inventories [121], manufacturing [80, 75, 20, 21, 158], or managing ambulances [19, 129, 148, 164]. Effective resource allocation ensures that limited resources are optimally allocated in order to achieve objectives, such as maximizing profits, minimizing costs, and many others. Numerous complex decision making problems can be formulated as resource allocation tasks. Notable instances include portfolio optimization [102, 103, 143, 46], the dynamic redeployment of ambulances between base stations [19, 129, 148, 164], allocating air-defenses or attack missiles to cities [37], and selecting drill sites to extract natural resources [85, 117]. In all resource allocation tasks, a limited resource is distributed among entities to optimize a certain objective. For instance, in portfolio optimization the investor distributes its money (resource) among several assets (entities) to maximize returns (objective). Although various types of resource allocation tasks exist, we focus on three key areas in this thesis: discrete resource allocation, resource collection, and continuous resource allocation with allocation constraints. Each of these areas comes with unique challenges, applications, and solutions, which we will explore in this thesis.

In discrete resource allocation, the resources are not divisible, as they represent countable instances, such as persons, ambulances, trucks, machines, and so forth. In this thesis, we focus on a specific discrete resource allocation task: dynamically redeploying ambulances. In this problem, ambulances are dynamically redeployed to base stations to minimize the response times to subsequent incidents. In addition to clas-

sical ambulance redeployment, we also introduce a new setting: the redeployment of electrical ambulances, which adds an additional layer of complexity due to recharging and managing battery levels. This problem is of particular interest because emergency medical system operators in both the UK [67] and US [27] plan to purchase electric ambulances. However, the existing research on electric ambulances is still very limited. Thus, we address this the lack of redeployment methods for electric ambulances in this thesis.

Resource collection represents a special type of (discrete) resource allocation, where the allocation is reversed. In this task, the agent (resource) is allocated to a set of collection resources (entities) with the objective of collecting as many collection resources as possible. In this thesis, we focus specifically on stochastic resource collection in both single-agent and multi-agent scenarios. Resource collection tasks are often spatial problems. The traveling officer problem represents a prominent instance of stochastic resource collection [136, 137, 150]. In this problem, a parking officer attempts to fine as many parking violations as possible by traversing a road network. The current state of each parking spot is provided by a sensor network. However, the task is highly stochastic and complex due to the inherent uncertainty in the future states of parking spots, which must be considered when planning.

Subsequently, we turn to continuous resource allocation with a specific focus on allocation constraints. In contrast to discrete resource allocation, continuous resource allocation allows the resource to be arbitrarily divided and, thus, an allocation represents the proportion of the resource assigned to each entity. Portfolio optimization is among the most important applications of continuous resource allocation. In many instances, continuous resource allocation tasks include allocation constraints [19, 97, 132]. Typically, allocation constraints are explicitly defined and expressed as a set of linear inequalities on the allocations, e.g., only allocate a maximum of 20% to a group of entities. In this thesis, we propose specialized approaches for handling allocation constraints that outperform previous methods. By leveraging the geometric properties of the constrained action space, which forms a convex polytope, we develop methods to decompose the action space into dependent and independent subspaces, allowing for more effective optimization.

Resource allocation tasks are often characterized by their complexity and dynamic nature, i.e., they are complex sequential decision making problems. Thus, reinforcement learning is a well-suited framework for solving resource allocation tasks. However, due to the unique characteristics specialized solutions are often required to successfully apply reinforcement learning to resource allocation tasks. In this thesis, we focus mostly on resource allocation in spatial problems and portfolio optimization. We propose numerous novel solutions designed for the specific requirements of these tasks, often based on deep reinforcement learning.

The remainder of this thesis is structured as follows. In Chapter 2 we give a brief background on artificial intelligence, including neural networks and reinforcement learning. In Chapter 3, we categorize resource allocation before we detail various different resource allocation tasks, their applications, and review related work. In Chapter 4, we present an overview of the main contributions in this thesis. The works included in this dissertation can be found in the Appendix. Finally, we conclude the thesis with Chapter 5, where we summarize the key findings, discuss the limitations of current methods, and outline potential directions for future research.

Chapter 2

Background on Artificial Intelligence

In recent years, artificial intelligence achieved breakthroughs in many fields and thus gained a lot of interest [40, 125, 162, 128, 108, 144, 145]. The field of artificial intelligence is very broad and different definitions of what artificial intelligence is exist [131]. In this thesis, our focus is on resource allocation tasks. Thus, we concentrate on the part of artificial intelligence for decision making and reinforcement learning. In particular, deep learning has played a crucial role in many of these recent advancements. Furthermore, the combination of reinforcement learning and deep learning (deep reinforcement learning) has achieved remarkable results, a technique that we often utilize in this thesis to solve resource allocations tasks.

Before we go into the specifics of resource allocation, we provide background on artificial intelligence. We start with a brief introduction to neural networks. Afterward, we continue with reinforcement learning. At the end of this chapter, we briefly look into multi-agent reinforcement learning and constrained reinforcement learning.

2.1 Deep Learning

Most of the recent successes in artificial intelligence have been achieved by deep learning. Examples are large language models [40, 125, 162], text-to-image models [128, 124], or agents achieving superhuman performance playing popular games [108, 144, 145]. Since deep learning plays an important role in many of our solutions, we want to give a brief overview. For a more detailed introduction, we refer the reader to the book of [60].

Multi-Layer Perceptron

A multi-layer perceptron (MLP) or feed-forward neural network is one of the most common deep learning models [60]. It consists of an input layer and one or more hidden

layers. Each layer consists of nodes, often called neurons, that are connected to all neurons in the subsequent layer, forming a fully connected network. Neurons use a non-linear activation function to process the weighted sum of inputs received from the previous layer. Each hidden layer in an MLP has a non-linear activation function σ , a learnable weight matrix $\mathbf{W} \in \mathbb{R}^{d_{in} \times d_{out}}$, a learnable bias term $\mathbf{b} \in \mathbb{R}^{d_{out}}$, and receives an input vector $\mathbf{x} \in \mathbb{R}^{d_{in}}$. Formally, a hidden layer is defined as a function:

$$f(\mathbf{x}) = \sigma(\mathbf{W}^T \mathbf{x} + \mathbf{b}) \quad (2.1)$$

Common activation functions include ReLU ($\max\{0, \alpha\}$), (logistic) sigmoid ($\frac{1}{1+\exp(-\alpha)}$), or the hyperbolic tangent ($\frac{\exp(\alpha)-\exp(-\alpha)}{\exp(\alpha)+\exp(-\alpha)}$) [60]. These activation functions introduce non-linearities that enable the network to model and learn complex relationships between the input and outputs. Training an MLP typically involves adjusting its parameters to minimize the error between the output of the network and the actual target values. This error is measured by a loss function. Backpropagation is used to calculate the gradient of the loss function with respect to the parameters of the network. Optimization algorithms, such as stochastic gradient descent (SGD) or Adam [82], use these gradients to update the network's parameters. While the MLP is a widely used deep learning architecture due to its simplicity and effectiveness, more specialized architectures tailored for specific tasks and data exist.

Convolutional Neural Networks

For example, convolutional neural networks [89] (CNN) are commonly utilized in computer vision tasks, such as object detection [58], image classification [86], or semantic segmentation [109]. CNNs are special kind of neural network designed to process data structured in a regular grid-like topology [60]. For example, images can be seen as 2-D grid of pixels, while time-series data can be thought of as a 1-D grid [60]. By applying learnable filters, also known as convolution kernels, across the input data, a CNN can capture simple patterns, such as edges, textures, or colors, in the early layers and more complex features, like complete objects, in deeper layers [90, 86, 68].

Recurrent Neural Networks

In contrast, recurrent neural networks [130] (RNN) are specifically designed for processing sequential data which occurs in time-series forecasting [71], speech recognition [61], or natural language processing [154]. RNNs share parameters across time steps and the network is recursively applied, i.e., the output of the previous time step is an input to the current time step, resulting in a very deep computational graph. However, traditional RNNs struggle with long-term dependencies due to the vanishing gradient

problem [72, 14]. To overcome this problem, the long short-term memory (LSTM) has been introduced by [73]. The LSTM uses memory cells and gating mechanisms, which allow to maintain and update information over long sequences, making them capable of learning long-range dependencies and capturing temporal dynamics effectively. RNNs are also a popular architecture for solving partially observable reinforcement learning problems [133].

Transformer

The transformer is currently the most popular architecture in natural language processing, outperforming traditional recurrent neural networks, like the LSTM, on many tasks [40, 125, 92, 32, 162, 24]. In [163], the authors introduced the transformer architecture, which relies on an attention mechanism to process the input data in parallel, rather than sequentially. This parallel processing allows to more effectively model long-term dependencies. The transformer is the backbone of models like BERT [40] and GPT [125], which achieved impressive results in tasks such as machine translation or question-answering systems. The success of the transformer is not only limited to natural language processing. Many transformer-based approaches achieved state-of-the-art performance in domains like computer vision [44, 100, 17] or reinforcement learning [26, 93].

Graph Convolutional Neural Network

Graph convolutional networks [83] (GCN) are a class of deep learning architectures designed for graph data. In contrast to CNNs, which are designed for data with a regular grid-like topology, GCNs generalize convolutions to graph-structured data, enabling them to aggregate information from neighboring nodes to learn representations of graphs. This makes GCNs particularly well-suited for tasks like node classification, link prediction, or graph generation in applications such as social network analysis, molecular chemistry, or recommender systems [83, 165, 57, 59, 79, 66, 179].

Although the field of deep learning is vast, we will now shift our focus to reinforcement learning. For a more comprehensive introduction to deep learning, we refer the reader to the book of [60].

2.2 Reinforcement Learning

Machine learning can be divided into four categories [110]: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. In supervised learning a problem is solved using labeled samples. Unsupervised learning consists of

finding patterns in the data without labels. The combination of supervised and unsupervised learning is known as semi-supervised learning. In this setting, only part of the data is labeled. The goal in reinforcement learning is to solve a problem by trial-and-error. Therefore, an agent interacts with the environment (often a simulator) to learn a policy. The agent receives a reward signal from the environment which guides learning [151]. A visualization of this process can be found in Figure 2.1.

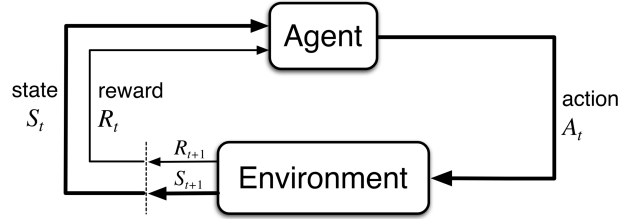


Figure 2.1: The agent-environment interface used in reinforcement learning. From Sec. 3.1 [151].

Markov Decision Process

Formally, reinforcement learning uses the framework of Markov decision processes (MDP) to define the interaction between the agent and its environment [151]. An MDP can be defined as a tuple $\langle S, A, P, R, \rho_0 \rangle$, where S denotes the set of all possible states, A the set of all possible actions, $P : S \times A \times S \rightarrow [0, 1]$ is the transition function, $R : S \times A \rightarrow \mathbb{R}$ is the reward function, and $\rho_0 : S \rightarrow [0, 1]$ is the distribution of the initial state S_0 . Oftentimes, a discount factor $\gamma \in [0, 1]$ is used. A trajectory τ , also referred to as rollout or episode, is a sequence of states, actions, and rewards:

$$\tau = S_0, A_0, R_1, S_1, A_1, R_2 \dots S_T \quad (2.2)$$

Here, S_t , A_t , R_t denote the state, action, and reward at time-step t , respectively. The sequence terminates at time T with the terminal state S_T . The probability that the process transitions to the new state S_{t+1} is given by transition function P and only depends on the current state S_t and the action A_t , independent of all previous states and actions. This is known as the Markov property. The transition function can be either deterministic $S_{t+1} = P(S_t, A_t)$ or stochastic $S_{t+1} \sim P(\cdot | S_t, A_t)$. The actions are generated by a policy function $\pi(S_t)$, which also can be either deterministic $A_t = \pi(S_t)$ or stochastic $A_t \sim \pi(\cdot | S_t)$. The return G_t is defined as the sum of rewards from time-step t to T :

$$G_t = R_{t+1} + R_{t+2} + \dots + R_T \quad (2.3)$$

In the infinite-horizon discounted case, the return G_t is total discounted reward from time-step t :

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (2.4)$$

The goal in reinforcement learning is to find a policy π that maximize the expected discounted reward:

$$J(\pi) = \mathbb{E}_{S_0 \sim \rho_0} [V_\pi(S_0)] = \mathbb{E}_{(S_0, A_0, \dots) \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \right] \quad (2.5)$$

This can be expressed as: $\pi^* = \arg \max_{\pi} J(\pi)$, where π^* denotes the optimal policy. However, finding an optimal policy can be hard and there are different ways to accomplish this.

Value Functions The estimation of value functions is a key component in almost all reinforcement learning algorithms. Value functions are functions of states (or state-action pairs) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a particular state) [151]. The notion of "how good" is defined in terms of future rewards that can be expected, or more precisely, the expected return [151]. The rewards the agent can expect to receive in the future depends on what actions it will take. As a consequence, value functions are defined with respect to a specific way of acting, i.e., a policy [151].

Formally, the value of a state $s \in S$ is the expected return when starting in s and following the policy π thereafter:

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] \quad (2.6)$$

In a similar way, an action-value function is defined. It represents the value of taking action a in state s under a policy π , i.e., the expected return of starting in state s , taking action a , and thereafter following policy π . This function is commonly known as Q-function:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \quad (2.7)$$

Value functions can be defined in terms of immediate rewards and future discounted rewards. This recursive relationship is considered a fundamental property of value functions and used throughout reinforcement learning.

This equation is known as the Bellman equation:

$$\begin{aligned}
V_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\
&= \mathbb{E}_\pi[R_t + \gamma G_{t+1} | S_t = s] \\
&= \mathbb{E}_\pi[R_t + \gamma V_\pi(S_{t+1}) | S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s',r} P(s', r | s, a) [r + \gamma V_\pi(s')]
\end{aligned} \tag{2.8}$$

Let us note that the Bellman equation can also be defined for action-value functions.

Dynamic Programming In the context of reinforcement learning, the term dynamic programming refers to a number of methods that compute optimal policies given a complete and perfect model of the environment as an MDP. However, given the requirement of a complete model of the environment and their great computational expenses, classical dynamic programming is of limited use in reinforcement learning [151]. Nonetheless, its theoretical foundations are essential to reinforcement learning. The idea in dynamic programming is to use the Bellman equations to iteratively evaluate functions and improve the policy. Policy evaluation, sometimes referred to as prediction, is the process of determining the state-value function for a given policy. Using the Bellman equation, the value function of a state is updated iteratively:

$$V_{k+1}(s) = \mathbb{E}_\pi[R_{t+1} + \gamma V_k(S_{t+1}) | S_t = s] = \sum_a \pi(a|s) \sum_{s',r} P(s', r | s, a) [r + \gamma V_k(s')] \tag{2.9}$$

The initial approximation V_0 is chosen arbitrarily. Each iteration of iterative policy evaluation updates the value of every state once to obtain a new approximation of the value function V_{k+1} . The update is repeated until the value function converges.

Policy improvement uses the value function obtained by policy evaluation to find a better policy. The idea is to improve the current policy by constructing a greedy policy π' from the current value function. For this we calculate the action-value function from the current value function:

$$Q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s, A_t = a] = \sum_{s',r} P(s', r | s, a) [r + \gamma V_\pi(s')] \tag{2.10}$$

Acting greedy according to this action-value function is guaranteed to yield a policy π' that is as good as, or better than the current policy π , according to the policy improvement theorem [151].

Now that we have obtained a better policy π' , we can obtain its value function $V_{\pi'}$ and improve it to get an even better policy π'' . Thus, we can iteratively obtain an optimal

policy by alternating between policy evaluation and policy improvement. This process is called policy iteration. In some variations of policy iteration, the evaluation and improvement step are not strictly separated. Instead, they can occur simultaneously. This is referred to as generalized policy iteration and many reinforcement learning algorithms can be viewed in this framework [151].

Value-Based and Policy-Based Reinforcement Learning

The MDP and its dynamics are often unknown [151]. Reinforcement learning is a method that can discover optimal policies by interacting with the MDP, therefore no prior knowledge of the environment dynamics are necessary [151]. While classical reinforcement learning employs methods such as table-based Q-learning, this approach is not scalable to high-dimensional and complex state/action spaces like image inputs or continuous control problems. To overcome these problems, deep learning can be combined with reinforcement learning. Deep reinforcement learning can be broadly classified into two paradigms. Policy-based methods directly optimize the policy parameterized by a neural network, whereas value-based methods do not explicitly learn a policy but derive it from a value function parameterized by a neural network. One of the most famous policy-based methods is REINFORCE [170]. DQN [108] is an example for a popular value-based method. However, before we can proceed to detail these methods, it is first necessary to introduce some fundamental concepts.

Monte Carlo Methods Value functions and optimal policies can be estimated from experience without knowledge of the complete environment [151]. Monte Carlo methods are one such class of methods that solve the reinforcement learning problem by averaging sample returns. To compute these empirical returns, they require experience in the form of complete episodes (trajectories) – sequences of states, actions, and rewards – sampled from the environment. It is required that all episodes must terminate eventually. Using the sampled episodes, we can perform policy evaluation to estimate the state or state-action values as follows:

$$Q(s, a) = \frac{\sum_{t=1}^T \mathbb{1}[S_t = s, A_t = a] G_t}{\sum_{t=1}^T \mathbb{1}[S_t = s, A_t = a]}, \quad (2.11)$$

where $\mathbb{1}[S_t = s, A_t = a]$ is a binary indicator function that either counts every visit of the state-action pair in the episode ("every-visit") or only the first encounter of the state-action pair in the episode ("first-visit").

In order to learn the optimal policy by using Monte Carlo, we use the idea of generalized policy iteration. This approach alternates between complete steps of policy improvement and policy evaluation. In policy improvement, a greedy policy is constructed

with respect to the current action-value function estimates. The policy evaluation step is performed as previously described.

Temporal-Difference Learning An important advancement in reinforcement learning are temporal-difference learning methods, which integrate ideas from both Monte Carlo learning and dynamic programming [151]. Similar to Monte Carlo methods, temporal-difference methods can directly learn from episodes of experience. Moreover, like dynamic programming methods, temporal-difference methods are able to learn from incomplete episodes, i.e., there is no requirement to wait until an episode terminates. Temporal-difference methods update estimates in part on learned estimates, without waiting for the final outcome (they bootstrap) [151].

The simplest temporal-difference method makes the following update to estimate the state-value:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \quad (2.12)$$

Here, the learning rate $\alpha \in (0, 1]$ is a constant hyperparameter. In contrast to Monte Carlo methods, this update can be made immediately upon transitioning to S_{t+1} and receiving reward R_{t+1} . This method is referred to as *TD(0)* or one-step TD. The key idea is to update the value function $V(S_t)$ towards an estimated return $R_{t+1} + \gamma V(S_{t+1})$ known as the temporal-difference target. In contrast, in Monte Carlo methods the target for the update is G_t .

Having introduced how to use temporal-difference learning for predictions, we now turn our attention to its application for control, i.e., learning an optimal policy. Like in Monte Carlo methods, we again follow the framework of generalized policy improvement. However, temporal-difference methods are used for the evaluation or prediction part.

SARSA is an on-policy temporal-difference control algorithm that uses the following update based on (state, action, reward, state, action) quintuples (hence the name):

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (2.13)$$

The actions are selected according to a policy derived from Q (e.g., ϵ -greedy).

Q-learning [168] is a popular off-policy temporal-difference method for learning optimal policies. The Q-function is updated according to the following equation:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (2.14)$$

In this approach, the learned action-value function Q directly approximates the optimal action-value function, independent of the policy being followed to pick the second action A_{t+1} [151].

Deep Q-Networks Although it is theoretically possible to memorize the Q-function for all state-action pairs in Q-learning in a table, this becomes quickly infeasible for large state or action spaces. Therefore, function approximators (such as neural networks) are used to approximate Q-values. However, Q-learning can suffer from instability and convergence issues when combined with non-linear function approximators and bootstrapping [151]. Deep Q-Networks (DQN) is an algorithm that combines Q-Learning with deep neural networks as function approximators [108]. To improve and stabilize training, DQN uses two key techniques: experience replay and a periodically updated target network. The loss function is defined as follows:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_{\theta}(s, a))^2 \right], \quad (2.15)$$

where $U(D)$ is the uniform distribution over the replay memory and θ^- denotes the parameters of the frozen target Q-network, which is periodically updated.

Policy Gradient Methods All methods introduced until this point aim to learn state-value or action-value functions and then derive the policy from the value function. In contrast, policy gradient algorithms directly learn a policy as a parameterized function π_{θ} . The policy may be parameterized by a neural network. We aim to learn the policy parameters based on the gradient of some scalar performance measure $J(\theta)$. The performance measure $J(\theta)$ is defined differently for the episodic and continuing case. For the sake of simplicity, we will only derive the episodic case and direct the reader to the book of [151] for the continuing case. In the episodic case, we define the performance measure as the value of the start state of the episode S_0 :

$$J(\theta) = V_{\pi_{\theta}}(S_0) \quad (2.16)$$

The policy parameters can now be optimized by updating them using the gradient $\nabla_{\theta} J(\theta)$. However, computing this gradient may seem challenging, given that the performance depends on both the action selections (which are directly determined by π_{θ}) and the distribution of states in which those selections are made (which are indirectly determined by π_{θ}) [151]. While the effect of the policy parameters on the actions in a given state can be computed relatively straightforwardly, the effect of the policy on the state distribution depends on the environment and is generally unknown. Fortunately, the pol-

icy gradient theorem provides an analytical solution for the gradient of the performance measure $\nabla_{\theta} J(\theta)$ that does not require the derivative of the state distribution:

$$\nabla_{\theta} J(\theta) \propto \sum_s \mu(s) \sum_a Q_{\pi}(s, a) \nabla_{\theta} \pi_{\theta}(a|s) = \mathbb{E}_{\pi_{\theta}}[Q_{\pi}(s, a) \nabla_{\theta} \ln \pi_{\theta}(a|s)] \quad (2.17)$$

In the episodic case, the constant of proportionality corresponds to the average length of an episode and μ is the on-policy distribution under π . For a detailed proof and further derivations, we refer the reader to the book by [151].

We now introduce REINFORCE [170], also known as Monte Carlo policy gradient, which is a very popular policy gradient method. It relies on returns that are estimated at the end of an episode as in Monte Carlo methods. More specifically, G_t are the returns from time step t to the end of the episode. Using the sampled expectation of returns, we can derive the following gradient:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [G_t \nabla_{\theta} \ln \pi_{\theta}(A_t|S_t)] \quad (2.18)$$

A widely used variation of REINFORCE is to subtract a baseline from the returns G_t in order to reduce the variance of the gradient estimation while keeping the bias unchanged. One such commonly used baseline is to subtract the estimated state-value $V(s)$.

Actor-critic methods combine policy-based and value-based reinforcement learning [151]. In addition to the policy-function, these methods learn a value-function, which can help learning policy [151]. Despite the fact that REINFORCE with a baseline learns both a policy and a state-value function, it is typically not considered to be an actor-critic method because the baseline is not used as a critic [151]. The key difference between a critic and a baseline is that a baseline is not used for bootstrapping. It only serves as a baseline for the state whose estimate is being updated [151]. Hence, in contrast to a critic, a baseline does not introduce bias by bootstrapping [151]. While REINFORCE with a baseline is unbiased, it tends to learn slowly due to the high variance, a common issue in Monte Carlo methods [151]. To leverage the advantages of both temporal-difference methods and policy gradient methods, actor-critic approaches with a bootstrapping critic are used [151]. Many widely used reinforcement learning methods are actor-critic based, including prominent examples such as PPO [139] or SAC [65]. PPO [139] is an actor-critic based algorithm that replaces the trust-region of TRPO [138] with clipped likelihood ratios, thereby enabling first-order optimization. SAC [65] is an off-policy actor-critic variant that optimizes an objective that balances expected returns and entropy. With this, we conclude our background on reinforcement learning.

Multi-Agent Reinforcement Learning

In many cases, multiple agents interact with the environment simultaneously. This setting is referred to as multi-agent reinforcement learning (MARL). MARL is particularly challenging due to the complex interactions between the agents.

In single-agent reinforcement learning, the environment is typically modeled as a Markov Decision Process (MDP). Since multi-agent reinforcement learning is very broad there exists no one-fits-all formalism. However, oftentimes a stochastic game can be used to model the environment in multi-agent reinforcement learning. A stochastic game can be seen as the extension of an MDP to multiple agents. Formally, it is defined as a tuple $M = \langle I, S, A, P, R, \gamma \rangle$, where $I = \{1, \dots, N\}$ is a set of agents, S is a set of states, $A = \langle A_1, \dots, A_N \rangle$ is the joint action space, P the joint transition function, $R = \langle R_1, \dots, R_N \rangle$ is a set of reward functions, and γ is a discount factor. Many different variations of multi-agent settings exist. For example, when all agents share the exact same individual reward function $r_i = r_j$ for $i, j \in I$, the game is referred to as fully cooperative. In case of a two-agent game with opposing reward functions $r_1 = -r_2$, the game is called a zero-sum game, where the agents are in direct competition, i.e., what one agent gains is lost by the other. General-sum games are between these two extremes, where agents may cooperate to some extent while still being in some competition [74].

A key challenge in multi-agent reinforcement learning is that the amount of possible joint actions exponentially increases with the number of agents. The joint action space is usually so large that it is infeasible to solve it directly [63]. Independent learners [152] are a common technique, where each agent is trained using single-agent reinforcement learning in the same environment concurrently. However, this method can lead to issues such as the credit assignment problem, where it becomes difficult to determine the impact of an individual action. Apart from that, centralized training decentralized execution (CTDE) is a common paradigm, where agents use global information during training but only have access to local information during execution [63]. For a more extensive overview of multi-agent reinforcement learning, we refer the reader to the survey of [63].

Constrained Reinforcement Learning

Many tasks in reinforcement learning involve constraints to limit the agent's behavior to ensure safety. This is especially common in resource allocation tasks [19, 97, 132]. Most of the literature in constrained reinforcement learning, also known as safe reinforcement learning, focuses on soft cumulated constraints [99, 64]. This setting can be formalized as a constrained Markov Decision Process (CMDP) [8]. A CMDP extends the standard MDP by incorporating a number of cost functions CF_1, \dots, CF_n . Similar to reward functions, the cost functions are typically not explicitly defined in a closed form in

safe reinforcement learning [99, 64]. Most of the literature in safe reinforcement learning focuses on expected cumulated costs [99, 64]. In this setting, the expected cumulated costs of each cost function J_{CF_i} should be smaller than a limit ϵ_i :

$$J_{CF_i}^{\pi_\theta} = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t CF_i(s_t, a_t) \right] \leq \epsilon_i \quad (2.19)$$

In a CMDP and safe reinforcement learning the objective is to maximize the expected rewards while ensuring that the expected costs stay within the specified limit:

$$\begin{aligned} & \max_{\theta} J_R(\pi_\theta) \\ & \text{s.t. } J_{CF_i}(\pi_\theta) \leq \epsilon_i \end{aligned}$$

While most of the research in safe reinforcement learning focuses on cumulated constraints, there is also a type of constraints, which must be satisfied at every step. These type constraints are known as instantaneous constraints and can be defined in a similar way:

$$\begin{aligned} & \max_{\theta} J_R(\pi_\theta) \\ & \text{s.t. } CF_i(s_t, a_t) \leq \epsilon_i \quad \forall s_t, a_t \end{aligned}$$

In practice, since CF_i is typically not explicitly defined, reinforcement learning agents often violate these constraints due to their lack of prior knowledge about the environment and the cost functions, necessitating learning through trial and error.

In contrast, hard constraints must be strictly satisfied and these constraints are explicitly defined in closed form:

$$\begin{aligned} & \max_{\theta} J_R(\pi_\theta) \\ & \text{s.t. } CF_i(s_t, a_t) \leq 0 \quad \forall s_t, a_t \end{aligned}$$

While hard constraint are less commonly encountered in the safe reinforcement learning literature, linear hard constraints are especially common in resource allocation tasks [19, 97, 132]. Note that this setting can be modeled as a CMDP and thus solved using standard safe reinforcement learning techniques for soft constraints. However, this does not guarantee that the constraints will be satisfied at all times [99, 43].

Chapter 3

Resource Allocation Tasks

Resource allocation is an umbrella term for a broad class of different tasks. In this chapter, we will give an overview of different classes of resource allocation tasks and their research challenges. At the core, resource allocation refers to allocating a limited resource over various entities in order to fulfill a goal or optimize an objective [75]. In this thesis, we focus on resource allocation tasks that go on over a time-frame in which allocations must be made periodically [80]. Resource allocation occurs in many applications from operations research [85, 37], managing the location of ambulances [19, 129, 148, 164], financial economics [102, 103, 172, 143, 46], or manufacturing [80, 75, 20, 21, 158]. A prominent resource allocation task, for example, is optimizing an investment portfolio. Here, the portfolio manager distributes its wealth (resource) over various assets (entities) to maximize the return (objective) [102, 103, 172, 143, 46]. Another prominent example is dynamically allocating ambulances (resource) to base stations (entities) in order to reduce the response times (objective) [19, 129, 148, 164]. Other tasks include the weapons allocation problem, where one needs to allocate air-defenses or attack missiles to cities [37]. Drilling for oil can also be formulated as a resource allocation problem. Here, one needs to select where to drill among possible drill sites in order to provide best returns [85, 117]. While there exist many different resource allocation tasks, in this thesis we focus on three different areas: discrete resource allocation, continuous resource allocation with allocation constraints, and resource collection.

In discrete resource allocation, the agent needs to allocate a discrete resource to entities. Discrete resources are not divisible since they represent persons, ambulances, trucks, machines, or other countable instances. Formally, we have a set of m resources $R = \{r_i\}_{i=1\dots m}$ and a set of n entities $E = \{e_1, \dots, e_n\}$. Each resource r_i must be allocated to an entity. An allocation maps each resource to an entity: $a = \{a_1, \dots, a_m\}$, where $a_i \in \{1, \dots, n\}$ is the index of the entity assigned to resource r_i .

Resource collection can be seen as a special case of (discrete) resource allocation. Here, the agent needs to periodically choose from set of collection resources, i.e., the agent (resource) is allocated the collection resources (entities) to collect as many of them as possible (objective). Many spatial optimization problems, such as the taxi dispatch problem or the traveling officer problem, can be formulated as resource collection tasks.

In continuous resource allocation, we have a limited resource that must be distributed over a set of n entities E . In contrast to discrete allocation tasks, the resource is divisible, i.e., the allocation $a = \{a_1, \dots, a_n\} \in \mathbb{R}_{0,+}^n$ s.t. $\sum_{i=1}^n a_i = 1$ represents the proportion of the resource assigned to each entity. Thus, the allocation takes non-negative real values and needs to be completely allocated.

Having outlined the three key areas resource allocation which we address in this thesis, we now proceed with providing a detailed examination of each. In particular, we will detail the specific tasks in each area and give a comprehensive analysis of existing solutions.

3.1 Discrete Resource Allocation

In a discrete resource allocation task, the agent is required to allocate a discrete resource to the entities. These resources, representing items like humans, machines, or ambulances, cannot be divided. Formally, we define a set of m resources $R = \{r_i\}_{i=1\dots m}$ and a set n entities $E = \{e_i\}_{i=1\dots n}$. Each resource r_i must be allocated to one of the entities. An allocation is a function that maps each resource to an entity: $a = \{a_1, \dots, a_m\}$, where $a_i \in \{1, \dots, n\}$ is the index of the entity assigned to resource r_i . Let us note that multiple resources can be assigned to a single entity and that some entities might not have any resource assigned to them.

While there exist many discrete resource allocation tasks, we will focus on dynamic ambulance redeployment. In this task, the emergency medical system (EMS) operator dynamically assigns ambulances (resource) to base stations (entities) to reduce response times or achieve response time targets with fewer ambulances (objective) [148].

The shift to electric vehicles leads to the emergence of electric ambulances. Due to charging, this task is more complex than that of combustion engine ambulances. This variation, known as dynamic electric ambulance redeployment, is introduced in this thesis. However, before we introduce this task, we first discuss classical dynamic ambulance redeployment without considering electric ambulances.

3.1.1 Dynamic Ambulance Redeployment

Ambulances are stationed at base stations from where they are dispatched to incidents. After handling an incident, they return to a base station and wait until they are dis-

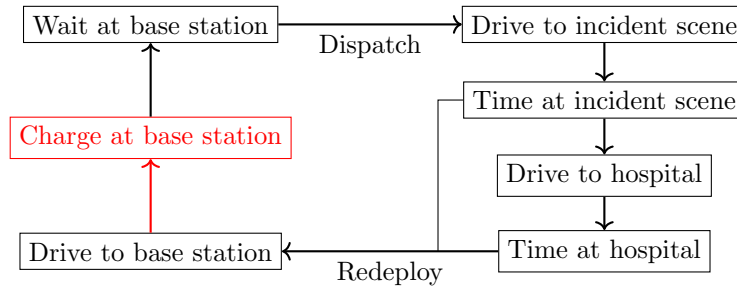


Figure 3.1: A visualization of the EMS process. Red indicates the additional step for electric vehicles. From [129].

patched to the next incident. This process is illustrated in Figure 3.1. Due to varying demands of ambulances, strategically redeploying them to base stations can improve response times and reduce the number of ambulances needed to operate the EMS system [4]. Many approaches allow redeployment only after an ambulance finished handling an incident [76, 78], a setting also adopted in this thesis. Some approaches, however, periodically redistribute ambulances [19, 180].

Redeployment Methods

The ambulance location problem, i.e., where to place ambulances, has been the subject of extensive research since it was first introduced in the 1970s [34, 113, 155, 15, 134, 164, 18]. Existing methods can be classified into either static or dynamic approaches.

Static Ambulance Redeployment In static approaches, each ambulance is assigned to a fixed base station, returning to the same station after handling an incident [15]. Many static approaches can be formulated as mixed integer linear programs [15]. One of the earliest solutions is the Maximum Coverage Location Problem (MCLP) [34]. In this approach, the ambulances are located (i.e., assigned to base stations) to maximize a coverage objective. However, this model assumes that ambulances are always available and the model is unable to place more than one ambulance at a base station. In [55], the authors introduce the Double Standard Model (DSM), which extends the MCLP to cover each demand location by at least two ambulances. The authors of [38] propose the Maximum Expected Covering Location Problem (MEXCLP), an approach that assumes that ambulances are busy with a certain probability. The Average Response Time Model (ARTM) [45] uses an objective that minimizes the average response time from the nearest station. In the Expected Response Time Model (ERTM) [15], the authors build on ARTM and optimize an objective that minimizes the expected response times, i.e., they take the availability of ambulances in a given area into account, allowing for the possibil-

ity that an ambulance might be dispatched from a base station that is further away. The authors of [18] propose a static model that further takes into account the stochasticity and focuses on robustness. In [180], the authors use simulated request and a greedy approach for the static allocation of ambulances.

Dynamic Ambulance Redeployment Dynamic approaches, in contrast, allocate ambulances dynamically to base stations taking into account real-time information, such as the current availability and locations of ambulances, as well as time-dependent shifts in incident frequencies at different locations. Such shifts may be due to daily or seasonal patterns. Therefore, dynamic redeployment can result in improved response times compared to static policies. This is due to the stochastic nature of incoming emergency calls and the potential for imbalances in the ambulance distribution that are not accounted for in static approaches. Consequently, EMS operators are able to provide a better service level without the necessity of increasing the number of ambulances [76]. Studies of North American EMS operators show that in recent years many EMS operators start utilizing dynamic redeployment strategies [25, 169, 4, 76]. There are numerous different dynamic approaches that have been developed [56, 76, 4, 16, 78, 105, 104, 113, 155, 180, 134]. Some approaches are based on lookup tables [4]. They calculate optimal solutions for each possible state in advance and the dispatcher attempts to redeploy ambulances in a manner that aligns with the configuration suggested by the lookup table. However, a major downside of these methods is the workload required (i.e., the movement of ambulances) to ensure that the system is in compliance with the lookup table. Furthermore, in busy regions where the number of idle ambulances fluctuates rapidly, the system will not be in compliance with the lookup table most of the time [76]. In a different line of research, approaches explicitly model the stochasticity of the EMS system and obtain solutions through either dynamic programming [16, 11, 135] or heuristics [76, 134, 56]. However, an exact dynamic programming formulation is intractable. Therefore, approximate dynamic programming approaches have been proposed [105, 104, 135]. As noted by [76], these approaches rely heavily on expert knowledge for their implementation, which makes them impractical and inaccessible. Additionally, their performance is highly dependent on the choice of base functions and thus unlikely to work in general settings [76]. Several heuristics have been proposed that are based on meta-heuristics [134, 56]. One of the earliest approaches is that of [56], which based on a tabu search. In [134], the authors use meta-heuristics based on genetic algorithms. However, the redistribution of ambulance only occurs at shift changes (e.g., from night shift to day shift). Another prominent approach is DMEXCLP [76], which is a dynamic version of MEXCLP that, at each redeployment, selects the base station that yields the largest increase in coverage (according to the MEXCLP objective). In [180], the authors further extend their greedy

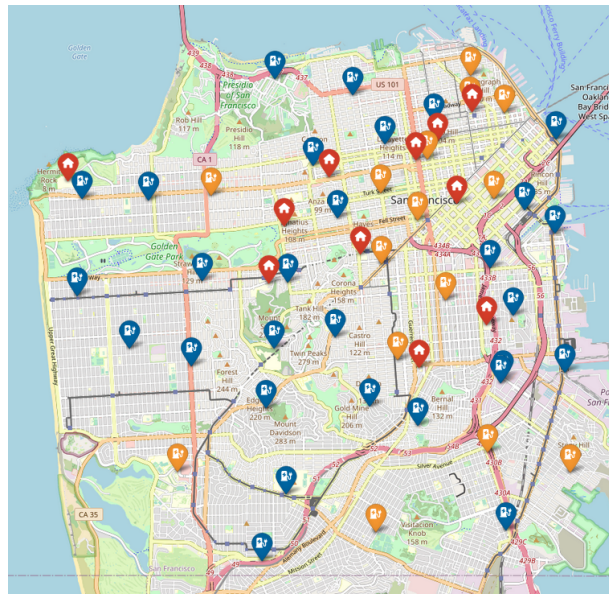


Figure 3.2: A map of San Francisco. Hospitals are indicated with red markers. Blue markers indicate base stations without fast-chargers, while orange indicates base stations with fast-chargers. From [129].

approach to a dynamic setting, where ambulances are periodically redistributed, by using a myopic greedy policy for each time-period. In [78], the authors present a deep reinforcement learning-based approach, which makes use of a demand forecast based on historical averages as well as the current distribution of ambulances over base stations. In [19], the authors use reinforcement learning for resource allocation tasks with hierarchical constraints and evaluate their method in an ambulance redeployment task. Their approach periodically reassigns all ambulances to base stations, upon which idle ambulances travel to the newly assigned station. In [119], the authors build a solution based on Monte Carlo tree search to dynamically position ambulances after an incident is reported. Following the same framework, in a very recent paper [146], the authors use multi-agent reinforcement learning. Specifically, one policy selects a region and another one rebalances ambulances within that region.

Related Tasks

Furthermore, research also extends to similar tasks such as ambulance dispatch, where one selects which ambulance is dispatched to an incident [39, 111, 11, 77]. In ambulance demand forecasting, researchers try to predict how many incidents, i.e., the demand for ambulances, will occur in a certain area [140, 167, 186, 185]. This information is important for many redeployment approaches [78].

Simulation and Evaluation In order to evaluate and train redeployment policies, a simulator is crucial. For example, the EMS operator can test a redeployment policy in a simulator first before rolling it out, since if the policy does not perform well, lives are at risk. Furthermore, reinforcement learning agents usually use a simulator for training, as they require a large number of trials to learn effectively. However, existing simulators for ambulance redeployment have limitations: they are not suited for reinforcement learning [134, 135, 115, 18], are not publicly available [39, 134, 119, 135, 70]¹, or cannot make use of real-world data [19, 180, 6, 76, 184]. For example, the openly available simulator of [115], is limited to static policies and does not allow dynamic redeployment. In contrast, the simulator of [6] is well suited for reinforcement learning, however, it is unable to replay real-world incident data. Additionally, many real-world datasets are not publicly available or only synthetic data fitted on the data is published, which limits comparability of approaches [134, 119, 146, 164, 70]. To the best of our knowledge, we are the first to use real-world EMS incident data from the city of San Francisco that is publicly available for ambulance redeployment [148]. A visualization of the area used, including hospitals and base stations, can be found in Figure 3.2.

In ambulance redeployment, researchers mainly consider two metrics: the response time target (RTT) and the average response time (ART). The RTT measures the fraction of incidents where an ambulance arrived within a specific threshold at the incident. Typical values are 8, 10, or 12 minutes, depending on regulations or EMS guidelines [134, 15]. In San Francisco, for example, the EMS operator’s target is that an ambulance arrives within 10 minutes for at least 90% of the time for live-threatening incidents [129]. Another metric that researchers commonly look at is the ART, the average of response times [76].

3.1.2 Dynamic Electric Ambulance Redeployment

Today, the majority of ambulances are internal combustion engine (ICE) vehicles. Ambulances are often modified versions of standard vehicles. However, due to the planned phase-out of combustion engine vehicles, it is likely that ambulances will also be based on electric vehicles in the future. In fact, this transition is already happening. Electric ambulances have already been developed and pilot-tested in real-world operations [1]. Furthermore, operators in both the UK [67] and the US [27] are making plans to purchase electric ambulances. However, the transition to electric ambulances requires significant changes in operational procedures, as the battery level and the associated charging requirements of the ambulances must be taken into account. Charging is a much more time-consuming process than refueling, particularly when using regular power outlets instead of fast chargers. Furthermore, the installation of fast chargers is costly. As a re-

¹[70] was initially publicly available, however, all links are dead (08/2024).

sult, not every station might be outfitted with a fast charger and the number of available fast chargers will be limited. It is therefore essential to consider the impact of charging times when redeploying electric ambulances. Additionally, the current battery level and resulting range must be taken into account, given that ambulances are only dispatched in a way that ensures that they will not run out of battery while handling an incident. Although EMS operators are planning to purchase electric ambulances, the research on this topic is extremely limited. To the best of our knowledge, [127] are the first to discuss the high-level challenges associated with switching to electric ambulances in a vision paper without providing concrete solutions. Given the challenges associated with electric ambulances, dynamically redeploying electric ambulances is an important aspect to enable a successful transition to electric ambulances. We are the first to formalize this task, evaluate existing redeployment strategies developed for ICE vehicles, propose a dynamic redeployment method for electric ambulances, develop a publicly available simulator, and show the feasibility of using electric ambulances in a realistic simulation in [129]. This work is presented in Appendix D. Operating electric ambulances presents a more complex scenario than operating ICE ambulances. This includes factors such as the availability and location of fast chargers, varying battery capacities of different ambulance models, and differences in charging speeds depending on both the ambulance model and installed charger. Many of these factors cannot be changed easily and, thus, the transition to electric ambulances requires careful planning in advance. For instance, installing a fast charger is costly and takes a significant amount of time (due to obtaining permits and installation). Similarly, purchasing electric ambulances requires careful and complex planning that involves selecting the right model (considering battery capacity, range, charging speeds), determining the number of ambulances required, and training the crews. This underscores the importance of a flexible and realistic simulator so that EMS operators are able to assess the impact of these factors and can successfully transition to electric ambulances.

In a recent pre-print, the authors of [42] examine the impact of transitioning to electric ambulances by conducting a case study in the region of Utrecht, Netherlands. Their case study is based on a simulator and the authors worked closely with EMS operators in the Netherlands, who showed interest in adapting electric ambulances. Although the researchers did not dynamically redeploy ambulances and instead used the static MEX-CLP model, which does not consider the specifics of electric ambulances, their case study indicates that transitioning to electric ambulances is feasible under certain conditions.

3.2 Stochastic Resource Collection

Resource collection can be viewed as a special subclass of (discrete) resource allocation, where the roles of resources and entities are reversed. In this task, the agent is required to choose among a set of collection resources. Whether a resource can be collected changes over time and the agent can observe the state of each resource at the current time t . As the collection of a resource does not occur instantaneously but rather requires the agent to reach the resource, it is uncertain whether the resource will still be available upon arrival. In the context of resource allocation, the agent is the resource that gets assigned to the collection resources, i.e., the entities. Most resource collection tasks are spatial problems. More formally, an agent tries to collect as many resources as possible that are located within a road network $G = (V, E, C)$, where V is a set of nodes, E denotes a set of edges, and $C : E \rightarrow \mathbb{R}^+$ are the corresponding costs for traversing an edge. Each resource $p \in P$ is mapped to an edge $e \in E$ of the road network. In this thesis, we focus on the most common stochastic resource collection task, the traveling officer problem (TOP). In this task, an agent traverses a road network with the objective of fining as many parking violations as possible [136]. The agent has access to the current state of each parking spot, which is obtained from a sensor network. However, the agent must also factor in the stochasticity associated with the dynamic state of parking spots. For example, a parking offender may leave before the officer's arrival, or a new violation may emerge [137].

Relation between Stochastic Resource Collection and Other Spatial Optimization Problems Stochastic resource collection is a spatial task with unique characteristics [137, 150, 136]. In the following, we highlight the differences and similarities to other spatial optimization problems. The vehicular routing problem (VRP) is a well-known area of research that received a lot of attention [161, 112, 106, 22]. In the VRP, the goal is to find the optimal route for one or more vehicles, with the objective of visiting a given set of customers in the shortest possible time. A crucial distinction to resource collection is that customers must be visited exactly once [161]. In contrast, in stochastic resource collection, resources (customers) can reappear. Furthermore, in contrast to VRP, resources in SRC change their collectability state and may not be collectable upon the agent's arrival. While there exist many variations of the VRP [161, 54, 22, 136], to the best of our knowledge, none of those fully consider the dynamic and uncertainty of SRC. For instance, some variants include the appearance of new customers. However, they do not include the disappearance of customers after an unknown time interval [54, 22, 137, 136]. Another prominent spatial problem is the traveling salesman problem (TSP), which can be viewed as a special case of the VRP with just one route [13, 84, 51, 106, 54]. SRC can be simplified as a TSP by planning a route with

the resources that are currently collectable [182, 142]. However, this simplification ignores a crucial aspect of SRC: resources may change their status over time. Ignoring this leads to sub-optimal policies [137, 136]. Additionally, resource collection often contains a large number of resources for which solving a TSP can be difficult [13, 84, 106]. The taxi dispatch problem (TDP) is a spatial task that is similar to SRC [137, 153, 81, 136]. Here, a dispatcher tries to optimize the distribution of taxis throughout the city in order to pick up customers more quickly [153, 81]. Despite the similarities between the two problems, there are significant differences between SRC and TDP. First, in TDP the dispatcher assigns each customer to a nearby taxi and in general the customer waits for the taxi to arrive. The dispatcher ensures that customers are not required to wait for an extended period of time by only assigning taxis that arrive within a certain maximum wait time. In contrast to SRC, the agent does not need to manage the risk of a customer disappearing before the taxi's arrival. Additionally, another key distinction is that in TDP, the taxi must drive to a location determined by the passenger, which is beyond the taxi driver's control. Furthermore, most of the time, the TDP is formulated on a grid rather than directly on the road network [81, 153, 7, 96]. While the task of SRC shows important differences to resource collection, TDP, TSP, and VRP can still be regarded as an instance of the broader field of resource allocation.

Solutions for Stochastic Resource Collection A variety of different approaches specifically designed for stochastic resource collection and its prominent instance, the traveling officer problem exist [137, 182, 69, 142, 150, 141]. These include simple heuristics [142], ant colony optimization [142], as well as various reinforcement learning-based approaches [137, 182, 69]. In [142], the authors propose a greedy heuristic that relies on a probabilistic model to select the resource that is most likely to be in violation. This straightforward heuristic is remarkably effective. Additionally, they propose a more sophisticated solution based on ant colony optimization by transforming the TOP in a time-varying traveling salesman problem. An imitation learning-based approach for TOP has been proposed to approximate optimal policies [141]. In [123], the authors utilize a genetic algorithm to solve the multiple traveling officers problem, a variation of the traveling officer problem where multiple officers fine parking violations concurrently. Several reinforcement learning-based approaches have been proposed [137, 182, 69]. In [137], the authors formulate stochastic resource collection as a Semi-Markov Decision Process (SMDP) and use a special graph-convolution-based architecture and Semi-double DQN. The authors of [182] take the well-known attention-based approach for VRP of [84] and apply it to the TOP. Similarly, in [69] the authors adopt pointer networks, proposed by [13], to the TOP.

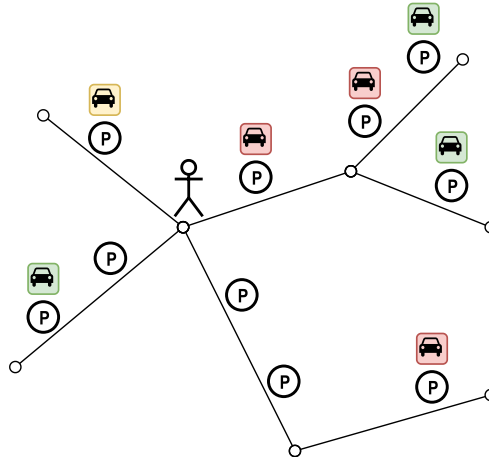


Figure 3.3: A visualization of the traveling officer problem. A green car indicates an occupied parking spot, red a violation, and yellow a fined parking spot. A parking spot without a car is currently free.

The Traveling Officer Problem

The Traveling Officer Problem (TOP) is a very prominent instance of a resource collection task [137, 182, 69, 142, 150, 141]. In this task, a parking officer traverses a road network and tries to fine as many parking violations as possible. While the officer has access to real-time information the current state of parking spots, such as whether a vehicle is illegally parked, the state of these spots can change by the time the officer reaches them. For example, a parking offender might leave before the officer arrives or a new violation occurs in another parking spot. As a result, the traveling officer problem is a highly dynamic and stochastic task. A visualization of the traveling officer problem is shown in Figure 3.3.

Formally, the TOP can be modeled as a fixed-horizon Semi Markov Decision Process (SMDP) [137]. A SMDP is defined as a tuple $\langle S, A, P, R, \gamma \rangle$, where S is the set of states, A denotes the set of possible actions, P represents the transition function, R is the reward function, and γ denotes the discount factor. Unlike a regular MDP, the transition function also determines the duration of an action, denoted as τ . In this formulation, which we follow in this thesis, the set of actions corresponds to all edges that have parking spots, i.e., the officer travels on the shortest path from its current position to the end of the edge associated with the action. As a result the duration of an action varies. The officer fines all violation encountered on its way.

Multi-Agent Stochastic Resource Collection

Many real-world stochastic resource collection tasks are too large and complex for a single agent to manage the task effectively. For example, when a single agent must collect

many resources spread over a large area in a short period of time, it might be quickly overwhelmed. Thus, in many applications multiple agents collect resources concurrently as a team [150, 123]. However, using multiple agents introduces additional complexity as it is no longer enough for each agent to act independently. Instead, they must also coordinate their actions to avoid redundant actions and conflicts to achieve a good overall result. The multiple traveling officer problem (MTO) is a prominent instance of a multi-agent stochastic resource collection task. In this task, multiple parking officers concurrently traverse a road network and try to fine as many parking violations as possible. The interaction and coordination between multiple parking officers adds another layer of complexity, in addition to the challenges of the single-agent traveling officer problem. More specifically, officers need to coordinate their plans and intentions. For example, if two or more officers target the same area, maybe even patrol the same street simultaneously, the presence of the second officer might not or only marginally increase the amount of parking violations fined. Instead, it would be more effective if one of the officers targets another area to increase the overall number of fined parking violations. However, if agents fail to cover important areas of the road network in order to prevent being too close to other agents, violations in these areas will not be fined, resulting in a poor overall performance. Therefore, agents must consider the plans, intentions, and policies of other agents in their decisions.

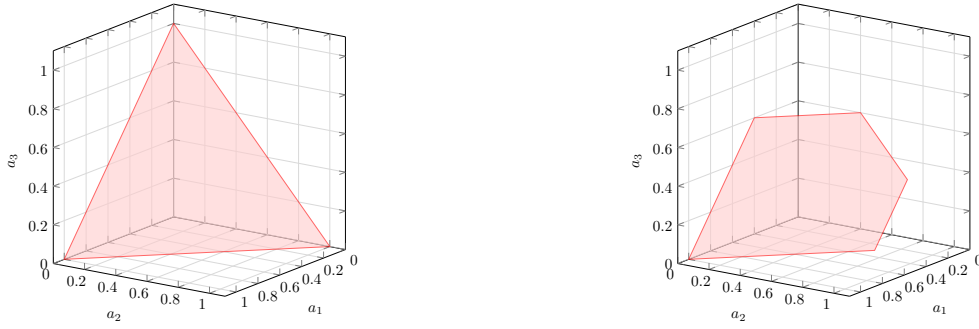
3.3 Continuous Resource Allocation Tasks

So far, we focused on discrete resource allocation problems. However, many allocation tasks require a continuous allocation. This includes numerous applications including power distribution, computational load balancing, security screening, and finance [19, 159, 5, 132, 172, 102, 103, 143, 46].

Constraints on the possible allocation play a crucial role in continuous allocation tasks [48, 118, 107, 9, 2, 19, 132]. For example, an investor may face several restrictions on possible allocations due to regulations or client preferences, such as limits of how much of the portfolio can be invested into assets with a certain rating [48, 118, 107, 9, 2].

In this section, we will first define continuous allocation tasks and allocation constraints. Then we will give an overview of important continuous allocation problems and focus especially on portfolio optimization. Finally, we will discuss existing methods to solve constrained continuous allocation tasks using reinforcement learning.

The goal in continuous allocation tasks is to find an allocation $a = \{a_1, \dots, a_n\}$ over a set of n entities $E = \{e_1, \dots, e_n\}$. Here, $a_i \in [0, 1]$ denotes the allocation to entity e_i . We also require that an allocation is completely allocated. In other words, allocations must



(a) Unconstrained standard simplex without additional allocation constraints.

(b) Constrained simplex with $a_3 \leq 0.6$ and $a_2 \leq 0.7$

Figure 3.4: Examples of 3-dimensional allocation action spaces (a) unconstrained and (b) constrained (valid solutions as red area) using our approach subject of Appendix G. Each dimension represents the allocation to one entity.

satisfy the simplex constraint, i.e., $\sum_{i=1}^n a_i = 1$, and cannot be negative $a_i \geq 0$. Note that even with those two constraints, this setting is often referred to as unconstrained.

Continuous resource allocation tasks often come with hard allocation constraints that restrict the action space. More formally, allocation constraints are a specific class of linear constraints that are defined as follows

$$\sum_{i=1}^n c_i a_i \leq b \quad (3.1)$$

Here, $c_i \in \mathbb{R}$ denotes the weighting of the allocation variable a_i to entity e_i and $b \in \mathbb{R}$ the corresponding constraint limit.

We only define \leq constraints since \geq can be transformed into \leq constraints by multiplying with -1 . Additionally, equality constraints $a = b$ can be expressed as two inequality constraints: $a \leq b$ and $-a \leq -b$.

Geometrically, the action space without allocation constraints forms an n -dimensional standard simplex. Given, some allocation constraints, the action space forms a convex polytope. This is illustrated in Figure 3.4

3.3.1 Overview of Continuous Resource Allocation Tasks

In this subsection, we give an overview of the continuous resource allocation tasks and their corresponding allocation constraints used in this thesis. Portfolio optimization is among the most important allocation tasks, and thus we use it extensively to evaluate the performance of our methods. However, there are many different definitions and variations of portfolio optimization. Therefore, we provide a detailed definition of portfolio

optimization and the variations used in this thesis. At the end of this subsection, we will briefly discuss other continuous allocation problems.

Portfolio Optimization

Portfolio optimization is one of the most prominent examples of continuous resource allocation [102, 103, 143, 46, 172, 49]. In this thesis, we focus on portfolio optimization tasks, where an investor needs to allocate its wealth over a set of assets in each time-step over a fixed investment horizon. Formally, we have a set of N assets $I = \{1, \dots, N\}$, or in the framework of resource allocation, entities. At each decision point, we need to decide the proportion of our wealth which we invest in each asset. Following the framework of continuous resource collection, we define the action space as all possible allocations:

$$A = \{a \in \mathbb{R}_{0,+}^N : \sum_{i=1}^N a_i = 1\} \quad (3.2)$$

Here, a_i denotes the allocation to the i^{th} asset. In this formulation, short selling is not allowed ($a_i \geq 0$) and the whole wealth has to be invested ($\sum_{i=1}^N a_i = 1$). Despite these basic constraints, we refer to this setting as unconstrained.

Allocation constraints are frequently used to restrict or require a minimum amount of allocation into a specific group of assets due to regulatory requirements or client preferences [48, 118, 107, 9, 2, 23, 62, 49]. For example, many investment funds are subject to restrictions that do not allow to invest more than a certain percentage of their portfolio into a single asset or a sector [62, 49]. Such constraints can be modeled by setting c_i to either 0 or 1 depending on whether the asset a_i belongs to the sector that should be restricted. The value of $b \in [0, 1]$ corresponds to the maximum investment allowed in the sector. Since c_i only takes binary values, these constraints are also referred to as binary constraints. Another example is investing considering ESG (Environmental, Social und Governance) criteria. In this case, the investor is required to adhere to certain environmental or sustainability criteria, such as only investing a maximum percentage in CO₂ heavy industries.

Additionally, more complex allocation constraints can be used to integrate criteria for assets like (estimated) dividend yield or other portfolio measures (like the weighted average cost of capital, the estimated return on equity, or ESG criteria). Constraints can than be used that ensure for example a minimum dividend yield of the portfolio or ESG related metrics [122, 160]. In this case, we create a constraint where each c_i corresponds to the dividend yield associated with the i^{th} asset and b represents the desired minimum dividend yield (note that c_i and b would be multiplied by -1 to obtain the \leq constraint). In contrast to binary constraints, the value of c_i can be any real number.

Short selling, or shorting, is a strategy that enables investors to generate profits when asset prices decline [126, 47, 147]. A common way to build a short position is to borrow the asset and immediately sell it [126, 47, 147]. At a later point in time the investor is obliged to repurchase the asset and return it to the lender. In case the value of the asset has fallen, the short seller makes a profit equal to the difference. However, when the price of the asset has increased, the short seller suffers a loss due to the obligation to repurchase the asset to return it to the lender. The act of repurchasing the assets that were sold short is referred to as "covering the short". Additionally, the short seller must typically pay a fee to borrow the asset. Short selling can be a risky strategy because, theoretically, there is no limit on how much the price of a stock can increase and, thus, the potential loss for a short seller is also unlimited [126, 47, 147].

Short selling can be used to construct a 130-30 portfolio. In this popular investment strategy, the investor first short sells stocks worth 30% of the capital they believe will underperform and use the resulting 130% of available capital to invest into stocks they expect to generate the highest returns [91, 53, 87]. This setting can be formulated as a constrained allocation problem. In this formulation, the subset of assets that will be shorted are assigned negative allocation weights. Additionally, we require the assets that are shorted to be mutually exclusive to the assets for which we build long position. Formally, this can be defined as follows:

$$\{a_i \in \mathbb{R}^N : \sum_{i=1}^N a_i = 1, \sum_{j \in V_{short}} a_j = c_{short}, a_j \leq 0 \forall j \in V_{short}, a_k \geq 0 \forall k \in V_{long}\} \quad (3.3)$$

Here, $I = \{1, \dots, N\}$ represents the set of all assets, $V_{short} \in I$ are the assets that must be shorted. Therefore, the allocations to those assets are negative and must add up to c_{short} , which denotes the amount of shortage. The set of long positions is denoted as $V_{long} \in I$. Allocations to long positions are positive and must add up to $1 + |c_{short}|$. Note that V_{short} and V_{long} are mutually exclusive. Overall, all allocations add up to 1, which allows the investor to invest $1 + |c_{short}|$.

Other Continuous Allocation Tasks

While portfolio optimization is a very prominent allocation task, allocating compute jobs to servers in a data center is another important resource allocation task [5, 156]. We follow the definition of this task as outlined by [5]. In this scenario, we have a set of n servers (entities) and we need to decide the fraction of each job to be computed on each server. When a job has been completely computed within its predetermined maximum allowed time, a reward is given. Each server has different computational

capabilities and an individual queue of jobs to be still computed which the agent needs to take into account when scheduling a job. Each job is defined by a payload size, i.e., the amount of data that needs to be transferred to the server, the number of CPU cycles required to process the job, and the maximum time allowed until the job needs to be fully computed. Jobs are generated by a specified number of users following a Poisson process. Other continuous resource allocation tasks include the allocation of software-testing resources [114, 88], energy allocation in satellites [101], blockchain applications [50], or dynamically allocating power in multi-battery systems [159].

3.3.2 Solving Continuous Allocation Tasks

After discussing a variety of different allocation tasks, we now continue by giving an overview over existing solutions to solve allocation tasks with allocation constraints. Let us note that we specifically focus on reinforcement learning to solve allocation tasks.

Reinforcement learning is a widely used method for solving continuous allocation tasks [172, 5, 159, 19, 132, 175]. Many of the most commonly used reinforcement learning algorithms are policy gradient-based and parameterize a probability distribution over the action space [139, 65, 12, 166, 94]. However, even in the absence of additional allocation constraints, the probability distribution still needs to fulfill the simplex constraints in continuous allocation tasks. The Dirichlet distribution is particularly well-suited because, unlike most other distributions, it ensures that the simplex constraint is always satisfied. Additionally, it has been shown that using a Dirichlet policy in allocation tasks often yields better results compared to alternative policies, such as a Gaussian-softmax policy [159, 5, 172, 175]. Therefore, the Dirichlet policy is a popular choice for many allocation tasks [159, 175, 172].

Continuous allocation tasks often come with allocation constraints that restrict the action space [19, 97, 132]. Geometrically, this action space forms a convex polytope. However, directly parameterizing a probability distribution over such a constrained action space is a challenging task. In fact, even uniformly sampling from a convex polytope can be a surprisingly difficult problem [41, 35].

One way to solve constrained allocation tasks is to transform the problem into a CMDP and applying standard safe reinforcement learning techniques [43]. To this end, cost functions are used that measure the violations of each constraint [43]. More specifically, the cost function measures how much the constraint is violated. Safe reinforcement learning typically focuses on implicit soft cumulated constraints, where the expected cumulated costs should be smaller than some specified limit ϵ_i . In this setting, the cost function is usually not explicitly defined (analogous to the reward function) and a strict adherence to the constraints, especially during training, cannot be guaranteed [43, 99, 64].

Many safe reinforcement learning approaches are based on Lagrangian relaxation [8, 99, 157, 28]. For example, in [29, 116] a primal-dual update mechanism is used. Other methods introduce penalty terms on the rewards to integrate the constraints [181, 157, 98]. Specifically, in [98] (IPO) the authors utilize a logarithmic barrier function, while in [181] (P30), the authors use a penalty function to convert the constrained problem into an equivalent unconstrained problem. The authors of [30, 31] construct Lyapunov functions to satisfy the constraints by projecting them back on to the set of feasible solutions induced by the Lyapunov functions. In [36] (SafetyLayer), the authors correct unsafe actions through projections based on a linear approximation of a safety signal. In [97], the authors combine IPO and SafetyLayer for constrained allocation tasks. The second-order optimization algorithm CPO [3] extends TRPO [138] and integrates the constraints into the trust region. In PCPO [178], the authors use a two-step trust region-based approach that also projects actions back onto the constraint set using a cost critic. Other two-step approaches, like FOCOPS [183] or CUP [176], are also popular approaches that use first-order optimization. However, these approaches cannot guarantee constraint satisfaction at all times, especially during training [43, 99, 64].

The majority of literature in constraint reinforcement learning has traditionally focused on implicit soft constraints. However, several methods for handling explicit hard constraints have been proposed [95, 19, 120, 43, 132]. In this setting, the constraints are explicitly given and usually strict adherence is required at all times. Most of these approaches are projection-based methods that correct infeasible actions, i.e., actions that do not satisfy the constraints, into feasible actions [95, 19, 120, 43, 132]. Let us note that the majority of these methods address linear allocation constraints, i.e., tasks where the action space forms a convex polytope [19, 120, 132]. A notable example is OptLayer [120], a projection-based approach based on OptNet [10]. OptLayer projects infeasible actions to the closest feasible action on the polytope by solving a quadratic problem in a differentiable way that allows easy integration into existing reinforcement learning algorithms. In [19], the authors focus on resource allocation tasks with hierarchical allocation constraints by proposing another projection based approach. The authors of [95] use a projection method based on Franke-Wolf policy optimization. The authors of [43] propose a projection based on generalized reduced gradients. In [132], the authors use an alpha-projection, which is a more efficient way to project a point outside to the surface of the polytope than quadratic programming. However, it has been shown that projection-based methods can lead to a biased policy gradient and thus may impact learning [28, 52, 99]. Additionally, many projection-based methods suffer from the zero-gradient issue, where small changes in the policy parameters do not lead to any change in the final output due to the projection mechanism [95].

Chapter 4

Overview of Contributions

In this chapter, we provide an overview of the publications included in this thesis and highlight the most important contributions. We focus on different areas of resource allocation. Our publications deal with challenges in these areas as well as those that arise from specific applications. The publications can be found in the Appendix. This also includes supplementary materials and a description of the author contributions in each work. This dissertation consists of seven publications that we will now detail [150, 148, 149, 129, 171, 173, 174].

4.1 Spatial-Aware Deep Reinforcement Learning for the Traveling Officer Problem

Solving complex spatial optimization problems like stochastic resource collection and the traveling officer problem with reinforcement learning requires specialized solutions and neural network architecture to achieve a good performance [137]. Nevertheless, current methods have several limitations. They do not fully account for the spatial relationship between the parking locations and the officer's current position, have difficulties scaling to realistically-sized real-world settings, and struggle in predicting the future implications of their actions due to the dynamic nature of the traveling officer problem.

In our work [149] (which can be found in Appendix A), we introduce a novel architecture for solving the traveling officer problem with reinforcement learning. We follow the formulation of the traveling officer problem as a stochastic resource collection task with temporally extended actions as proposed by [137]. Our first contribution is a novel pathing module that is capable of more effectively encoding a temporally extended action, particularly the spatial relationships between the officer's current location, parking spots, and each action. Our second contribution is a future positioning module that uti-

lizes message passing to encode the interactions of each action and potential the future actions, thereby enabling a better estimate of how well the agent is positioned after each action.

Our approach is able to consistently outperform the current state of the art by up to 22% in terms of fined parking violations using real-world parking data from Melbourne replayed in a simulator.

4.2 Reinforcement Learning for Multi-Agent Stochastic Resource Collection

In resource collection tasks, like the traveling officer problem, it is common that multiple agents work in parallel [123]. In this work [150] (see Appendix B), our first contribution is to formalize multi-agent stochastic resource collection as a Semi-Markov game. Our second contribution is a novel architecture that includes an intent combination module to better assess the intents of other agents. We train the policy of each agent using independent Q-learning. Even though the task is cooperative, we show that using selfish rewards, i.e., rewarding each agent only for the resource collected by itself, results in a more effective overall policy. We argue that this is because the price of anarchy, i.e., the impact of acting selfish over cooperating [33], is low in multi-agent stochastic resource collection tasks. This is because agents naturally tend to spread out and avoid direct competition, a behavior that one would also expect when the agents cooperate. Therefore, the selfish rewards, which avoid the credit assignment problem, outweigh the potential benefits of a cooperative joint reward. We evaluate this effect and also compare our approach by performing extensive experiments using a simulator that replays real-world parking data from the city of Melbourne. We demonstrate that our approach is able to significantly outperform existing methods. Additionally, we perform several ablations to examine the impact of individual parts of our architecture.

4.3 A Comparison of Ambulance Redeployment Systems on Real-World Data

It is challenging to make direct comparisons between existing ambulance redeployment methods. This is due to the fact that existing researchers frequently use simulators and datasets that are not publicly available. Moreover, a considerable number of existing research does not employ real-world data, instead relying on synthetic benchmarks. A major limitation of many synthetic datasets is the lack of temporal variability. In other words, the rate of incident occurrence remains constant across different time scales. For

example, the analysis of real-world datasets shows that fewer incidents occur at night compared to daytime [148]. This can lead to unrealistic scenarios and the development of redeployment methods that perform well on synthetic data but poorly in the real world. In light of the ethical considerations involved, it is crucial to examine the impact of an ambulance redeployment policy on the EMS through the use of a simulator prior to its implementation in the real world. In this context, accurately replaying real-world data is important. In this work [148] (found in Appendix C), our first main contribution is the introduction of an event-based simulator that can replay publicly available real-world data from the city of San Francisco. One of our primary focus points was developing a simulator that is well-suited for reinforcement learning to facilitate research in this direction. Our second contribution is the implementation of several existing redeployment approaches. Finally, our third contribution is the evaluation of existing redeployment strategies using our high-performance simulation in a close to real-world setting, revealing that very simple baselines like redeploying to the nearest station oftentimes work surprisingly well and is able to outperform complex state-of-the-art approaches. This includes the reinforcement learning-based approach of [78], which struggles to surpass existing methods, including simple heuristics in many scenarios. We hope to introduce a more standardized and realistic benchmark for ambulance redeployment and facilitate future research in this area with our openly available simulator.

4.4 DEAR: Dynamic Electric Ambulance Redeployment

The transition to electric vehicles not only affects individual transportation and public transport but also impacts emergency services. Since ambulances are often modified versions of standard vehicles, it is likely that they too will be based on electric vehicles in the future. In fact, this transition is already happening. Electric ambulances have already been developed and pilot-tested in real-world operations [1]. Furthermore, operators in the UK [67] and the US [27] are planning to purchase electric ambulances. However, the transition to electric ambulances requires significant changes in operational procedures, as the battery levels and the associated charging requirements of the ambulances must be taken into account. Therefore, we introduce and formalize the new task of dynamic electric ambulance redeployment (presented in Appendix D). In our work [129], we are the first to evaluate the impact of electric ambulances on the EMS systems by developing a simulator that closely mirrors real-world conditions and is able to replay actual incident data from the city of San Francisco. We use our simulator to evaluate how well existing redeployment strategies developed for ICE ambulances perform with electric ambulances. Additionally, we introduce minimizing energy deficits (MED), a novel redeployment strategy that considers the charging requirements and battery lev-

els of electric ambulances. The core idea of MED is to minimize the energy deficit with each redeployment decision. The energy deficit is calculated using demand estimates and predicted future charging levels at each base station. The results of our evaluation demonstrate the effectiveness of our novel redeployment policy in various settings. Furthermore, we show that redeployment strategies developed for ICE ambulances show a significant drop in performance when applied to electric ambulances. However, with our approach, electric ambulances can achieve a similar service level comparable to ICE ambulances (i.e., a response time of less than 10 minutes for 90% of the incidents) without requiring additional ambulances in a realistic scenario using real-world data from the city of San Francisco. Additionally, we evaluate our approach across a multitude of different settings with varying numbers of fast chargers, different charging speeds, numbers of ambulances, and different years of data. We are consistently able to outperform the baselines in almost all settings. These results further highlight that under reasonable conditions transition to electric ambulances is not only feasible but it is possible to do so, while maintain a comparable service level to ICE ambulances. Additionally, our openly available simulator can be a crucial tool for EMS operators in planning and transitioning to electric ambulances.

4.5 Constrained Portfolio Management Using Action Space Decomposition for Reinforcement Learning

Portfolio optimization is an important resource allocation task that is often accompanied by binary allocation constraints. In this paper [171] (subject of Appendix E), we propose a decomposition of the constraint action space into multiple independent subspaces that can be sampled in parallel. These subspaces can be parameterized with a Dirichlet distribution. Using the Minkowski sum, we can combine the subspaces back into the original action space. We evaluate our approach in two portfolio optimization tasks. In the first setting, we construct a long-only portfolio with ESG constraints, where the investor is required to allocate at least 40% of its portfolio to the stocks of companies with an Environmental Score Metric (ESM) of 80 or higher. The ESM score, which is provided by the financial data provider LSEG, is a rating system that measures a company's environmental sustainability. In the second setting, a 130-30 portfolio is constructed through short-selling. In this case, the investor is required to build a short-position that amounts to 30% of the available capital. The constraints specify the stocks from which the short position can be built as well as the size of the short position. Our approach is able to outperform existing methods, while guaranteeing constraint satisfaction at all times. Our evaluation also includes backtesting, i.e., evaluating the trained model on unseen real-world market data. We are able to outperform the baselines with our approach in

backtesting as well. Furthermore, we also include theoretical proofs that our decomposition indeed parameterizes an action space equivalent to the original action space. Thus, we can guarantee constraint satisfaction.

4.6 Simplex Decomposition for Portfolio Allocation Constraints in Reinforcement Learning

The primary contribution of this paper [173] (see Appendix F) is the extension of our previous approach to allow for two binary allocation constraints. We decompose the simplex action space into several unconstrained sub-action spaces that can be parameterized using a Dirichlet distribution. Importantly, our approach ensures that the constraints are always satisfied. We also provide a formal proof that our approach always guarantees constraint satisfaction. We conduct an extensive experimental evaluation focused on the task of portfolio optimization. As in our previous paper, this also includes backtesting. The results demonstrate that our approach can consistently outperform the baselines. Moreover, we also present a proof that our decomposition can always guarantee constraint satisfaction by showing that the set of actions from our decomposed surrogate action space is equivalent to those of the original action space.

4.7 Autoregressive Policy Optimization for Constrained Allocation Tasks

In our previous papers [171, 173] (see Section 4.5 and Section 4.6), we are limited in the number of constraints and are only able to handle binary constraints. In this paper [174] (which can be found in Appendix G), we present an approach that overcomes these limitations and is able to handle an arbitrary number of linear allocation constraints. We achieve this by defining the policy using an autoregressive decomposition that utilizes linear programming to sample the allocation for each entity in a step-by-step process. An illustration of this process can be found in Figure 4.1 Our policy can be trained using existing policy gradient-based reinforcement learning algorithms, such as PPO [139]. Another key contribution of this paper is the introduction of a novel de-biasing mechanism to ensure that the initial policy samples uniformly across the entire polytope. This is because uniformly sampling each individual allocation from the range of possible allocations in each step results in an exponential decrease of possible allocations for the remaining entities. Our ablations clearly show that without the de-biasing mechanism, the policy is unable to overcome this initial bias and converges early to a sub-optimal policy. Additionally, the de-biasing mechanism makes the policy robust to the order in

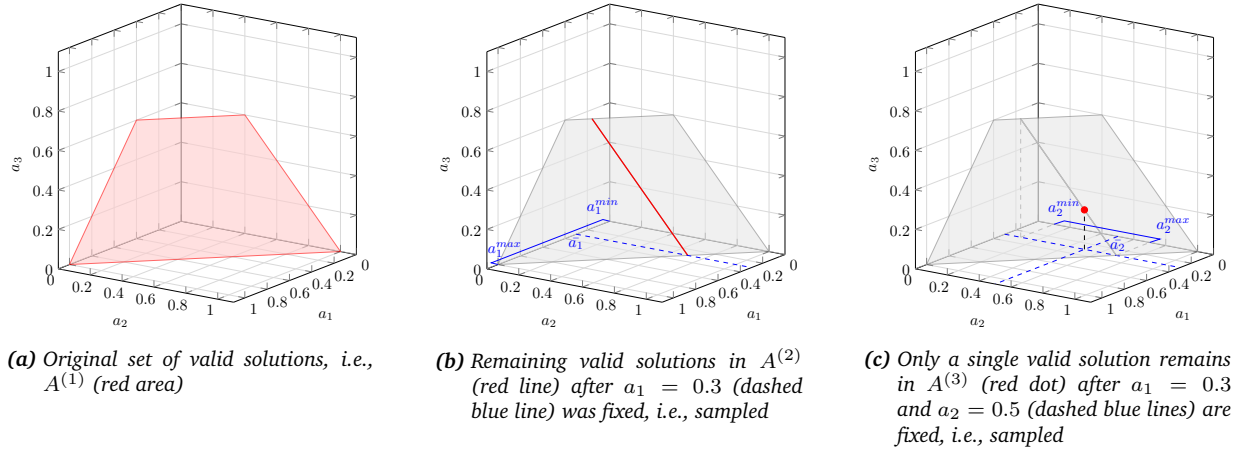


Figure 4.1: Example of sampling process of an action $a = (a_1, a_2, a_3)$ in a 3-dimensional constrained allocation task using the approach from our paper *Autoregressive Policy Optimization for Constrained Allocation Tasks* (see Appendix G). Figure from Appendix [174].

which the entities are allocated. We evaluate our approach in three different resource allocation tasks with a wide range of different allocation constraints: distributing computational workloads, portfolio optimization, and on a novel synthetic benchmark introduced by us. The results demonstrate the effectiveness of our approach, which is able to significantly outperform existing methods while guaranteeing constraint satisfaction at all times.

Chapter 5

Conclusion and Outlook

In this thesis, we presented multiple advances in different areas of resource allocation. In the following, we will briefly summarize our main contributions before we discuss future research opportunities as well as limitations. Resource allocation is a broad topic, and our work focused on a number of specific areas. From an applications perspective, we mainly focused on various spatial tasks and portfolio optimization. Specifically, we improved stochastic resource collection in single [149] as well as multi-agent scenarios [150]. Furthermore, we examined dynamic redeployment of combustion engine ambulances [148] and electric ambulances [129] in this thesis, two examples of discrete resource allocation. Afterward, we looked into continuous resource allocation with allocation constraints, with portfolio optimization as a key application. We developed approaches that can effectively incorporate one [171] and two allocation constraints [173] into standard reinforcement learning algorithms using a decomposition of the simplex action space. Finally, we propose an approach that is able to handle an arbitrary number of allocation constraints by iteratively sampling actions in each dimension autoregressively by utilizing linear programming to compute the range of possible actions [174]. Again, our approach can be trained using standard reinforcement learning algorithms, such as PPO.

5.1 Limitations and Future Work

In general, we envision several main axes for future work:

- Currently, numerous resource allocation tasks have been addressed by using task-specific, specialized approaches [150, 149, 129, 5, 132]. Developing more generic solutions that can be applied across various resource allocation tasks represents a promising research direction. In particular, foundation models have been shown

to cover a wide range of tasks in natural language processing or computer vision [40, 125, 128, 124]. Given the success of foundation models in these domains, researchers also proposed the use of foundation models for decision making [177]. The development of a foundation model for resource allocation tasks represents a promising but challenging area of research.

- While some resource allocation tasks can be effectively solved for a large number of resources or entities, many existing solutions struggle to scale. For instance, approaches in continuous resource allocation, particularly portfolio optimization, are limited to a small number of assets. Therefore, overcoming these limitations yields a promising opportunity for future research.
- While the aforementioned directions for future research described opportunities in the broad field of resource allocation, individual tasks offer a many task-specific research opportunities. For example, in electric ambulance redeployment, managing dispatch and redeployment jointly represents a promising research direction. In continuous resource allocation, the combination of our approaches with safe reinforcement learning methods to address allocation and other safety constraints simultaneously is an exiting research opportunity. Furthermore, it should be investigated, if our approaches can be transferred to certain types of non-linear constraints and discrete action spaces. Finally, there exists a plethora of different resource allocation tasks. Thus, applying our approaches to new use cases is an interesting research direction.

Overall, we have made substantial contributions to both individual resource allocation tasks as well as to the broader field. Nevertheless, as outlined, numerous opportunities for future research remain.

References

- [1] Was 500 electric ambulance - wietmarscher ambulanz- und sonderfahrzeug gmbh. <https://www.was-vehicles.com/en/innovation/was-500-electric-ambulance.html> [Accessed: (2024-08-09)].
- [2] G. Abate, T. Bonafini, and P. Ferrari. Portfolio constraints: An empirical analysis. *International Journal of Financial Studies*, 10(1):9, 2022.
- [3] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.
- [4] R. Alanis, A. Ingolfsson, and B. Kolfal. A markov chain model for an ems system with repositioning. *Production and operations management*, 22(1):216–231, 2013.
- [5] L. Ale, S. A. King, N. Zhang, A. R. Sattar, and J. Skandaraniyam. D3pg: Dirichlet ddpq for task partitioning and offloading with constrained hybrid action space in mobile-edge computing. *IEEE Internet of Things Journal*, 9(19):19260–19272, 2022.
- [6] M. Allen, K. Pearn, and T. Monks. Developing an openai gym-compatible framework and simulation environment for testing deep reinforcement learning agents solving the ambulance location problem. *arXiv preprint arXiv:2101.04434*, 2021.
- [7] A. Alshamsi, S. Abdallah, and I. Rahwan. Multiagent self-organization for a taxi dispatch system. In *8th international conference on autonomous agents and multi-agent systems*, pages 21–28, 2009.
- [8] E. Altman. *Constrained Markov Decision Processes*, volume 7. CRC Press, 1999.
- [9] N. Amenc, F. Goltz, and A. Lioui. Practitioner portfolio construction and performance measurement: Evidence from europe. *Financial Analysts Journal*, 67(3):39–50, 2011.

- [10] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pages 136–145. PMLR, 2017.
- [11] T. Andersson and P. Värbrand. Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society*, 58(2):195–201, 2007.
- [12] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.
- [13] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.
- [14] Y. Bengio, P. Frasconi, and P. Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pages 1183–1188. IEEE, 1993.
- [15] P. L. V. D. Berg and J. T. V. Essen. Comparison of static ambulance location models. *International Journal of Logistics Systems and Management*, 32(3-4):292–321, 2019.
- [16] O. Berman. Dynamic repositioning of indistinguishable service units on transportation networks. *Transportation Science*, 15(2):115–136, 1981.
- [17] M. Bernhard, N. Strauß, and M. Schubert. Mapformer: Boosting change detection by using pre-change information. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16837–16846, 2023.
- [18] D. Bertsimas and Y. Ng. Robust and stochastic formulations for ambulance deployment and dispatch. *European Journal of Operational Research*, 279(2):557–571, 2019.
- [19] A. Bhatia, P. Varakantham, and A. Kumar. Resource constrained deep reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 610–620, 2019.
- [20] G. R. Bitran and A. C. Hax. On the design of hierarchical production planning systems. *Decision Sciences*, 8(1):28–55, 1977.
- [21] G. R. Bitran and A. C. Hax. Disaggregation and resource allocation using convex knapsack problems with bounded variables. *Management Science*, 27(4):431–441, 1981.

- [22] G. Bono, J. S. Dibangoye, O. Simonin, L. Matignon, and F. Pereyron. Solving multi-agent routing problems using deep attention mechanisms. *IEEE Transactions on Intelligent Transportation Systems*, 22(12):7804–7813, 2020.
- [23] A. Braun, H. Schmeiser, and F. Schreiber. Portfolio optimization under solvency ii: Implicit constraints imposed by the market risk standard formula. *Journal of Risk and Insurance*, 84(1):177–207, 2017.
- [24] T. B. Brown. Language models are few-shot learners. *arXiv preprint ArXiv:2005.14165*, 2020.
- [25] G. Cady. 200 city survey. jems 2001 annual report on ems operational & clinical trends in large, urban areas. *JEMS: a journal of emergency medical services*, 27(2):46–65, 2002.
- [26] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.
- [27] E. Chhabra. Ambulance business goes electric with its new fleet, Apr 2022. <https://www.forbes.com/sites/eshachhabra/2022/04/14/ambulance-business-goes-electric-with-its-new-fleet/> [Accessed: (2024-08-09)].
- [28] P.-W. Chou, D. Maturana, and S. Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International conference on machine learning*, pages 834–843. PMLR, 2017.
- [29] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18(167):1–51, 2018.
- [30] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.
- [31] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.
- [32] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

- [33] C. Chung, K. Ligett, K. Pruhs, and A. Roth. The price of stochastic anarchy. In *Algorithmic Game Theory: First International Symposium, SAGT 2008, Paderborn, Germany, April 30-May 2, 2008. Proceedings 1*, pages 303–314. Springer, 2008.
- [34] R. Church and C. R. Velle. The maximal covering location problem. *Papers in regional science*, 32(1):101–118, 1974.
- [35] K. Giomek and M. Kadziński. Polyrun: A java library for sampling from the bounded convex polytopes. *SoftwareX*, 13:100659, 2021.
- [36] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.
- [37] J. M. Danskin. *The theory of max-min and its application to weapons allocation problems*, volume 5. Springer Science & Business Media, 2012.
- [38] M. S. Daskin. A maximum expected covering location model: formulation, properties and heuristic solution. *Transportation science*, 17(1):48–70, 1983.
- [39] D. H. Dayapule, A. Raghavan, P. Tadepalli, and A. Fern. Emergency response optimization using online hybrid planning. In *IJCAI*, pages 4722–4728, 2018.
- [40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [41] P. Diaconis, G. Lebeau, and L. Michel. Gibbs/metropolis algorithms on a convex polytope. *Mathematische Zeitschrift*, 272(1):109–129, 2012.
- [42] N. A. Dieleman and C. J. Jagtenberg. Electric ambulances: will the need for charging affect response times? *Available at SSRN 4874479*, 2024.
- [43] S. Ding, J. Wang, Y. Du, and Y. Shi. Reduced policy optimization for continuous control with hard constraints. *Advances in Neural Information Processing Systems*, 36, 2024.
- [44] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [45] M. Dzator and J. Dzator. An effective heuristic for the p-median problem with application to ambulance location. *Opsearch*, 50:60–74, 2013.

- [46] E. J. Elton, M. J. Gruber, and M. W. Padberg. Simple criteria for optimal portfolio selection. *The Journal of finance*, 31(5):1341–1357, 1976.
- [47] J. E. Engelberg, A. V. Reed, and M. C. Ringgenberg. Short-selling risk. *The Journal of Finance*, 73(2):755–786, 2018.
- [48] M. Escobar-Anel, M. Kschonnek, and R. Zagst. Portfolio optimization with allocation constraints and stochastic factor market dynamics. *arXiv preprint arXiv:2303.09835*, 2023.
- [49] F. J. Fabozzi and H. M. Markowitz. *The theory and practice of investment management: Asset allocation, valuation, portfolio construction, and strategies*, volume 198. John Wiley & Sons, 2011.
- [50] J. Feng, F. R. Yu, Q. Pei, J. Du, and L. Zhu. Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems. *IEEE Transactions on Wireless Communications*, 19(6):4321–4334, 2020.
- [51] M. M. Flood. The traveling-salesman problem. *Operations research*, 4(1):61–75, 1956.
- [52] Y. Fujita and S.-i. Maeda. Clipped action policy gradient. In *International Conference on Machine Learning*, pages 1597–1606. PMLR, 2018.
- [53] G. L. Gastineau. The short side of 130/30 investing for the conservative portfolio manager. *Journal of Portfolio Management*, 34(2):39, 2008.
- [54] M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European journal of operational research*, 88(1):3–12, 1996.
- [55] M. Gendreau, G. Laporte, and F. Semet. Solving an ambulance location model by tabu search. *Location science*, 5(2):75–88, 1997.
- [56] M. Gendreau, G. Laporte, and F. Semet. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel computing*, 27(12):1641–1653, 2001.
- [57] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.
- [58] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

- [59] V. Gligorijević, P. D. Renfrew, T. Kosciolatek, J. K. Leman, D. Berenberg, T. Vatanen, C. Chandler, B. C. Taylor, I. M. Fisk, H. Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):3168, 2021.
- [60] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
- [61] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.
- [62] E. Grizickas Sapkute, M. Sánchez-Granero, L. García, and J. Trinidad Segovia. The impact of regulation-based constraints on portfolio selection: The spanish case. *Humanities and Social Sciences Communications*, 9(1):1–14, 2022.
- [63] S. Gronauer and K. Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022.
- [64] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.
- [65] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.
- [66] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
- [67] C. Hampel. Nhs develops electric ambulances with ford, Jan 2024. <https://www.electrive.com/2021/08/09/nhs-procures-fleet-of-electric-ambulances/> [Accessed: (2024-08-09)].
- [68] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [69] R. He, X. Xiao, Y. Kang, H. Zhao, and W. Shao. Heterogeneous pointer network for travelling officer problem. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.
- [70] S. G. Henderson, A. J. Mason, et al. Bartsim: a tool for analysing and improving ambulance performance in auckland, new zealand. In *Proceedings of the 35th*

- annual conference of the operational research society of New Zealand, Wellington, New Zealand*, pages 57–64, 2000.
- [71] H. Hewamalage, C. Bergmeir, and K. Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.
- [72] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.
- [73] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [74] J. Hu and M. P. Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.
- [75] T. Ibaraki and N. Katoh. *Resource allocation problems: algorithmic approaches*. MIT press, 1988.
- [76] C. J. Jagtenberg, S. Bhulai, and R. D. van der Mei. An efficient heuristic for real-time ambulance redeployment. *Operations Research for Health Care*, 4:27–35, 2015.
- [77] C. J. Jagtenberg, S. Bhulai, and R. D. van der Mei. Dynamic ambulance dispatching: is the closest-idle policy always optimal? *Health care management science*, 20:517–531, 2017.
- [78] S. Ji, Y. Zheng, Z. Wang, and T. Li. A deep reinforcement learning-enabled dynamic redeployment system for mobile ambulances. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(1):1–20, 2019.
- [79] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.
- [80] N. Katoh and T. Ibaraki. Resource allocation problems. *Handbook of Combinatorial Optimization: Volume 1–3*, pages 905–1006, 1998.
- [81] J. Kim and K. Kim. Optimizing large-scale fleet management on a road network using multi-agent deep reinforcement learning with graph neural network. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 990–995. IEEE, 2021.

- [82] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [83] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [84] W. Kool, H. van Hoof, and M. Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.
- [85] B. O. Koopman. The optimum distribution of effort. *Journal of the Operations Research Society of America*, 1(2):52–63, 1953.
- [86] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- [87] C. Krusen, F. Weber, and R. A. Weigand. 130/30 funds. *The Journal of Retirement*, 2008(1):176–185, 2008.
- [88] P. Kubat and H. S. Koch. Managing test-procedures to achieve reliable software. *IEEE Transactions on Reliability*, 32(3):299–303, 1983.
- [89] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.
- [90] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [91] M. L. Leibowitz, S. Emrich, and A. Bova. *Modern portfolio management: active long/short 130/30 equity strategies*, volume 488. John Wiley & Sons, 2009.
- [92] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.
- [93] W. Li, H. Luo, Z. Lin, C. Zhang, Z. Lu, and D. Ye. A survey on transformers in reinforcement learning. *arXiv preprint arXiv:2301.03044*, 2023.
- [94] Y. Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

- [95] J.-L. Lin, W. Hung, S.-H. Yang, P.-C. Hsieh, and X. Liu. Escaping from zero gradient: Revisiting action-constrained reinforcement learning via frank-wolfe policy optimization. In *Uncertainty in Artificial Intelligence*, pages 397–407. PMLR, 2021.
- [96] K. Lin, R. Zhao, Z. Xu, and J. Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1774–1783, 2018.
- [97] Y. Liu, J. Ding, and X. Liu. A constrained reinforcement learning based approach for network slicing. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2020.
- [98] Y. Liu, J. Ding, and X. Liu. Ipo: Interior-point policy optimization under constraints. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4940–4947, 2020.
- [99] Y. Liu, A. Halev, and X. Liu. Policy learning with constraints in model-free reinforcement learning: A survey. In *The 30th international joint conference on artificial intelligence (ijcai)*, 2021.
- [100] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.
- [101] J. J. G. Luis, M. Guerster, I. del Portillo, E. Crawley, and B. Cameron. Deep reinforcement learning for continuous power allocation in flexible high throughput satellites. In *2019 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAaw)*, pages 1–4. IEEE, 2019.
- [102] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.
- [103] H. Markowitz. *Portfolio Selection: Efficient Diversification of Investments*. Cowles Foundation for Research in Economics at Yale University. Monograph 16. "The method of analysis presented in this monograph was originally developed for ... the author's doctoral dissertation. University of Chicago." Includes bibliography. Wiley, 1959.
- [104] M. S. Maxwell, S. G. Henderson, and H. Topaloglu. Tuning approximate dynamic programming policies for ambulance redeployment via direct search. *Stochastic Systems*, 3(2):322–361, 2013.

- [105] M. S. Maxwell, M. Restrepo, S. G. Henderson, and H. Topaloglu. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing*, 22(2):266–281, 2010.
- [106] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021.
- [107] R. O. Michaud and T. Ma. Efficient asset management: a practical guide to stock portfolio optimization and asset allocation., 2001.
- [108] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [109] Y. Mo, Y. Wu, X. Yang, F. Liu, and Y. Liao. Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing*, 493:626–646, 2022.
- [110] M. Mohammed, M. B. Khan, and E. B. M. Bashier. *Machine learning: algorithms and applications*. Crc Press, 2016.
- [111] A. A. Nasrollahzadeh, A. Khademi, and M. E. Mayorga. Real-time ambulance dispatching and relocation. *Manufacturing & Service Operations Management*, 20(3):467–480, 2018.
- [112] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.
- [113] D. Neira-Rodado, J. W. Escobar-Velasquez, and S. McClean. Ambulances deployment problems: categorization, evolution and dynamic problems review. *ISPRS International Journal of Geo-Information*, 11(2):109, 2022.
- [114] H. Ohtera and S. Yamada. Optimal allocation and control problems for software-testing resources. *IEEE Transactions on Reliability*, 39(2):171–176, 1990.
- [115] J. Ong, D. Kulpanowski, Y. Xie, E. Nikolova, and N. M. Tran. Openems: an open-source package for two-stage stochastic and robust optimization for ambulance location and routing with applications to austin-travis county ems data. *arXiv preprint arXiv:2201.11208*, 2022.

- [116] S. Paternain, M. Calvo-Fullana, L. F. Chamon, and A. Ribeiro. Safe policies for reinforcement learning via primal-dual methods. *IEEE Transactions on Automatic Control*, 68(3):1321–1336, 2022.
- [117] M. Patriksson. A survey on the continuous nonlinear resource allocation problem. *European Journal of Operational Research*, 185(1):1–46, 2008.
- [118] S. Perrin and T. Roncalli. Machine learning optimization algorithms & portfolio allocation. *Machine Learning for Asset Management: New Developments and Financial Applications*, pages 261–328, 2020.
- [119] G. Pettet, A. Mukhopadhyay, M. J. Kochenderfer, and A. Dubey. Hierarchical planning for dynamic resource allocation in smart and connected communities. *ACM Transactions on Cyber-Physical Systems*, 6(4):1–26, 2022.
- [120] T.-H. Pham, G. De Magistris, and R. Tachibana. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6236–6243. IEEE, 2018.
- [121] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.
- [122] Y. Qi and X. Li. On imposing esg constraints of portfolio selection for sustainable investment and comparing the efficient frontiers in the weight space. *Sage Open*, 10(4):2158244020975070, 2020.
- [123] K. K. Qin, W. Shao, Y. Ren, J. Chan, and F. D. Salim. Solving multiple travelling of-ficers problem with population-based optimization algorithms. *Neural Computing and Applications*, 32:12033–12059, 2020.
- [124] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [125] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.
- [126] A. V. Reed. Short selling. *Annu. Rev. Financ. Econ.*, 5(1):245–258, 2013.
- [127] E. S. Rigas, A. Billis, and P. D. Bamidis. Can artificial intelligence enable the transition to electric ambulances? In *Challenges of Trustable AI and Added-Value on Health*, pages 73–77. IOS Press, 2022.

- [128] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- [129] L. Rottkamp, N. Strauß, and M. Schubert. Dear: Dynamic electric ambulance redeployment. In *Proceedings of the 18th International Symposium on Spatial and Temporal Data*, pages 11–20, 2023.
- [130] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [131] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.
- [132] S. Sanket, A. Sinha, P. Varakantham, P. Andrew, and M. Tambe. Solving online threat screening games using constrained action space reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2226–2235, 2020.
- [133] A. M. Schäfer, S. Udluft, et al. Solving partially observable reinforcement learning problems with recurrent neural networks. In *Workshop Proc. of the European Conf. on Machine Learning*, pages 71–81, 2005.
- [134] M. E. Schjølberg, N. P. Bekkevold, X. Sánchez-Díaz, and O. J. Mengshoel. Comparing metaheuristic optimization algorithms for ambulance allocation: An experimental simulation study. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1454–1463, 2023.
- [135] V. Schmid. Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming. *European journal of operational research*, 219(3):611–621, 2012.
- [136] S. Schmoll. *Navigation with uncertain spatio-temporal resources*. PhD thesis, lmu, 2021.
- [137] S. Schmoll and M. Schubert. Semi-markov reinforcement learning for stochastic resource collection. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3349–3355, 2021.
- [138] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

- [139] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [140] H. Setzler, C. Saydam, and S. Park. Ems call volume predictions: A comparative study. *Computers & Operations Research*, 36(6):1843–1851, 2009.
- [141] W. Shao, F. D. Salim, J. Chan, S. Morrison, and F. Zambetta. Approximating optimisation solutions for travelling officer problem with customised deep learning network. *arXiv preprint arXiv:1903.03348*, 2019.
- [142] W. Shao, F. D. Salim, T. Gu, N.-T. Dinh, and J. Chan. Traveling officer problem: Managing car parking violations efficiently using sensor data. *IEEE Internet of Things Journal*, 5(2):802–810, 2017.
- [143] W. F. Sharpe. A simplified model for portfolio analysis. *Management science*, 9(2):277–293, 1963.
- [144] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- [145] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.
- [146] A. Sivagnanam, A. Pettet, H. Lee, A. Mukhopadhyay, A. Dubey, and A. Laszka. Multi-agent reinforcement learning with hierarchical coordination for emergency responder stationing. In *Forty-first International Conference on Machine Learning*, 2024.
- [147] K. F. Staley. *The art of short selling*, volume 4. John Wiley & Sons, 1996.
- [148] N. Strauß, M. Berrendorf, T. Haider, and M. Schubert. A comparison of ambulance redeployment systems on real-world data. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1–8. IEEE, 2022.
- [149] N. Strauß and M. Schubert. Spatial-aware deep reinforcement learning for the traveling officer problem. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pages 869–877. SIAM, 2024.
- [150] N. Strauss, D. Winkel, M. Berrendorf, and M. Schubert. Reinforcement learning for multi-agent stochastic resource collection. In *Joint European Conference on*

- Machine Learning and Knowledge Discovery in Databases*, pages 200–215. Springer, 2022.
- [151] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [152] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.
- [153] X. Tang, Z. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1780–1790, 2019.
- [154] K. M. Tarwani and S. Edem. Survey on recurrent neural network in natural language processing. *Int. J. Eng. Trends Technol*, 48(6):301–304, 2017.
- [155] J. Tassone and S. Choudhury. A comprehensive survey on the ambulance routing and location problems. *arXiv preprint arXiv:2001.05288*, 2020.
- [156] G. Tesauro et al. Online resource allocation using decompositional reinforcement learning. In *AAAI*, volume 5, pages 886–891, 2005.
- [157] C. Tessler, D. J. Mankowitz, and S. Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2019.
- [158] P. T. Thach and T. Thang. Problems with resource allocation constraints and optimization over the efficient set. *Journal of Global Optimization*, 58(3):481–495, 2014.
- [159] Y. Tian, M. Han, C. Kulkarni, and O. Fink. A prescriptive dirichlet power allocation policy with deep reinforcement learning. *Reliability Engineering & System Safety*, 224:108529, 2022.
- [160] C. Torricelli, B. Bertelli, et al. Esg compliant optimal portfolios: The impact of esg constraints on portfolio optimization in a sample of european stocks. *CEFIN WORKING PAPERS*, 2022.
- [161] P. Toth and D. Vigo. *The vehicle routing problem*. SIAM, 2002.
- [162] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

- [163] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [164] S. M. Vazirizade, A. Mukhopadhyay, G. Pettet, S. El Said, H. Baroud, and A. Dubey. Learning incident prediction models over large geographical areas for emergency response. In *2021 IEEE International Conference on Smart Computing (SMART-COMP)*, pages 424–429. IEEE, 2021.
- [165] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.
- [166] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078, 2022.
- [167] Z. Wang, T. Xia, R. Jiang, X. Liu, K.-S. Kim, X. Song, and R. Shibasaki. Forecasting ambulance demand with profiled human mobility via heterogeneous multi-graph neural networks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1751–1762. IEEE, 2021.
- [168] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.
- [169] D. M. Williams. Jems 2008 200 city survey: the future is your choice. *JEMS: a journal of emergency medical services*, 34(2):36–51, 2009.
- [170] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.
- [171] D. Winkel, N. Strauß, M. Schubert, Y. Ma, and T. Seidl. Constrained portfolio management using action space decomposition for reinforcement learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 373–385. Springer, 2023.
- [172] D. Winkel, N. Strauß, M. Schubert, and T. Seidl. Risk-aware reinforcement learning for multi-period portfolio selection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 185–200. Springer, 2022.
- [173] D. Winkel, N. Strauß, M. Schubert, and T. Seidl. Simplex decomposition for portfolio allocation constraints in reinforcement learning. In *ECAI*, 2023.
- [174] D. Winkel, N. A. Strauß, M. Bernhard, Z. Li, T. Seidl, and M. Schubert. Autoregressive policy optimization for constrained allocation tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

- [175] H. Yang, H. Park, and K. Lee. A selective portfolio management algorithm with off-policy reinforcement learning using dirichlet distribution. *Axioms*, 11(12):664, 2022.
- [176] L. Yang, J. Ji, J. Dai, L. Zhang, B. Zhou, P. Li, Y. Yang, and G. Pan. Constrained update projection approach to safe policy optimization. *Advances in Neural Information Processing Systems*, 35:9111–9124, 2022.
- [177] S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, and D. Schuurmans. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023.
- [178] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge. Projection-based constrained policy optimization. In *International Conference on Learning Representations*, 2020.
- [179] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.
- [180] Y. Yue, L. Marla, and R. Krishnan. An efficient simulation-based approach to ambulance fleet allocation and dynamic redeployment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 398–405, 2012.
- [181] L. Zhang, L. Shen, L. Yang, S. Chen, B. Yuan, X. Wang, and D. Tao. Penalized proximal policy optimization for safe reinforcement learning. *arXiv preprint arXiv:2205.11814*, 2022.
- [182] R. Zhang, C. Yang, and X. Peng. Dynamic graph attention network for traveling officer problem. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2022.
- [183] Y. Zhang, Q. Vuong, and K. Ross. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 33:15338–15349, 2020.
- [184] L. Zhen, K. Wang, H. Hu, and D. Chang. A simulation optimization framework for ambulance deployment and relocation problems. *Computers & Industrial Engineering*, 72:12–23, 2014.
- [185] Z. Zhou. Predicting ambulance demand: Challenges and methods. *arXiv preprint arXiv:1606.05363*, 2016.

- [186] Z. Zhou and D. S. Matteson. Predicting melbourne ambulance demand using kernel warping. 2016.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Matthias Schubert, for all the guidance, constant support, and invaluable feedback throughout my PhD. Thank you for giving me the opportunity to obtain my PhD under your supervision.

I am also extremely grateful to my reviewers, Goce Trajcevski and Sebastian Tschitschek, for their willingness to dedicate their valuable time to reviewing this thesis.

I would also like to express my gratitude to my colleagues, co-authors, and friends in our chair who have supported and assisted me in numerous ways over the past years. Specifically, I want to thank David, Lukas, Maxi, Sandra, Max, Collin, Zongyue, Joao, Franz, Thomas, and Susanne. It was a great pleasure to work with you.

Furthermore, I am very grateful to my family and friends for their unwavering support during my PhD. Thank you!

Appendix

A Spatial-Aware Deep Reinforcement Learning for the Traveling Officer Problem

Venue 2024 SIAM International Conference on Data Mining

DOI <https://epubs.siam.org/doi/10.1137/1.9781611978032.99>

Declaration of authorships The research idea was proposed, developed, and conceptualized by Niklas Strauss and discussed with Matthias Schubert. The implementations and the experiments were done by Niklas Strauss. The manuscript was written by Niklas Strauss and improved by Matthias Schubert.

Publication

Spatial-Aware Deep Reinforcement Learning for the Traveling Officer Problem

Niklas Strauß*

Matthias Schubert*

Abstract

The traveling officer problem (TOP) is a challenging stochastic optimization task. In this problem, a parking officer is guided through a city equipped with parking sensors to fine as many parking offenders as possible. A major challenge in TOP is the dynamic nature of parking offenses, which randomly appear and disappear after some time, regardless of whether they have been fined. Thus, solutions need to dynamically adjust to currently fineable parking offenses while also planning ahead to increase the likelihood that the officer arrives during the offense taking place. Though various solutions exist, these methods often struggle to take the implications of actions on the ability to fine future parking violations into account. This paper proposes SATOP, a novel spatial-aware deep reinforcement learning approach for TOP. Our novel state encoder creates a representation of each action, leveraging the spatial relationships between parking spots, the agent, and the action. Furthermore, we propose a novel message-passing module for learning future inter-action correlations in the given environment. Thus, the agent can estimate the potential to fine further parking violations after executing an action. We evaluate our method using an environment based on real-world data from Melbourne. Our results show that SATOP consistently outperforms state-of-the-art TOP agents and is able to fine up to 22% more parking offenses.

Keywords: Reinforcement Learning, Deep Learning, Spatial Optimization, Traveling Officer Problem

1 Introduction

In recent years, reinforcement learning (RL) has been successfully applied to tackle complex optimization problems [2, 10]. One of these optimization tasks that has attracted significant attention is the traveling officer problem (TOP) [16, 15, 13, 8, 22] where a parking officer navigates through a sensor-equipped road network and tries to fine as many parking offenders as possible during a shift. Though the locations of current parking offenses are known to the officer due to real-time data from the sensor network, the agent has to travel to the location before writing out a ticket. Thus, the offender might leave before the agent arrives, and new offenses might occur in the meantime.

As pointed out in previous works [13, 17], TOP yields considerably different challenges from related routing-based problems such as the vehicle routing problem (VRP) or the taxi dispatch problem (TDP). For instance, in TOP, violations might yield rewards

only for a very short time, and offenders do not wait for the agent to arrive. In contrast, VRP and TDP settings usually guarantee a certain waiting time, and the cost of not visiting a location is usually considerably higher than in TOP. In TOP, the agent is not considered to require a significant time to write a ticket and can continue its task from the same location. In contrast, in VRP and TDP, agents often must spend time handling customers and might have to change location within this time, e.g., in case of a taxi trip.

Several approaches have been proposed to solve TOP. These include heuristics [16], ant colony optimization [16], imitation learning [15], and RL [13, 22, 8]. However, existing approaches have limitations as they may not fully exploit the spatial relationships between parking locations and the officer’s current location, struggle to scale to realistically sized settings, and have difficulty assessing possible future implications of actions due to the dynamic nature of TOP.

In this paper, we propose SATOP, a RL-based agent that leverages a novel spatial-aware neural network architecture for TOP. Our agent considers all edges containing parking locations as actions and travels on the shortest path to this destination as proposed in [13]. This implies that our agent works on a Semi Markov Decision Process (SMDP), as different actions take different times to execute. To capture the impact of these temporally extended actions, our novel architecture employs a pathing module evaluating the path the agent will take to reach the target location of an action. In addition, we propose a future positioning module, which encodes the correlations between actions and possible future actions using a spatial-aware message-passing mechanism. This way, our new architecture can more easily adapt to the dynamics of TOP and learn the potential of fining future offenses. In contrast to existing RL-based approaches for TOP, our method is able to consistently outperform classical optimization approaches such as ant colony optimization by a large margin.

In line with previous research, we evaluate SATOP on a simulation environment that replays real-world parking data from Melbourne. We compare our approach with state-of-the-art methods for TOP. Our experimental evaluation shows that our method can

*Munich Center for Machine Learning, LMU Munich {strauss,schubert}@dbs.ifi.lmu.de

A Spatial-Aware Deep Reinforcement Learning for the Traveling Officer Problem 63

consistently outperform state-of-the-art approaches, including existing RL-based approaches. Furthermore, we investigate the impact of various architectural choices in an ablation study. To summarize, our main contributions are as follows:

- A pathing module to encode the impact of an action in a more comprehensive way.
- A novel message passing-based future positioning module encoding inter-action correlations and, thus, the potential of fining future offenses.
- A joint architecture that outperforms state-of-the-art methods for TOP by a significant margin of up to 22%.

The remainder of this paper is organized as follows: Section 2 provides a comprehensive review of related work, and places TOP in the context of similar optimization problems. In Section 3 and Section 4, we formally define the task and present the architecture of SATOP, including both new modules. Section 5 describes our experimental setup and the results of our evaluation. Finally, Section 6 concludes the paper with a summary of our contributions and ideas for future work.

2 Related Work

2.1 TOP and Related Spatial Optimization Problems In this section, we compare TOP to related problems, namely the VRP, the traveling salesman problem (TSP), and the TDP. We also highlight the unique challenges and differences inherent to TOP compared to these related tasks.

The VRP has been extensively studied by AI researchers [5, 2, 3, 10, 11]. It involves one or more vehicles that must visit a given set of customers in minimal time. Though violations and customers represent locations that provide a reward when visited, customers in the VRP need to be visited exactly once. In TOP, parking spots can dynamically change their status and offenders do not wait for the agent to fine them. Thus, in contrast to VRP, agents must reach violations before the offenders leave on their own. While various VRP variants exist [3, 10, 5], they do not fully include the dynamic and uncertain nature of TOP. For instance, some variations consider the appearance of new customers during the day, but they do not include the disappearance of customers after an unknown time interval [5, 3].

The well-known TSP can be viewed as a special case of VRP. Although one might attempt to simplify TOP to a variant of the TSP by planning a path through the current violations [22, 16], this approach overlooks a critical aspect: the changing states of parking spots

over time. Ignoring this aspect can lead to sub-optimal strategies. Moreover, in TOP, the number of parking spots can be significantly higher than in traditional TSP instances, making it difficult to solve the TSP. In recent years, RL-based methods to address the TSP [2, 10] emerged, but the dynamic nature of TOP necessitates novel approaches tailored to its unique demands.

The most closely related task to TOP is TDP, where dispatchers distribute taxis throughout a city to facilitate quick customer pick-ups [19, 9]. However, significant differences exist between TOP and TDP despite their similarities. In TDP, a dispatcher assigns each customer a nearby taxi. In general, the customer is expected to wait for the taxi to arrive, and the dispatcher only assigns taxis that can reach the customer within a maximum waiting time. In TOP, the agent has to manage the risk that a violation might disappear before its arrival. Another difference between TDP and TOP is that after picking up a customer, a taxi has to drive to a location determined by the customer. Thus, information about other available customers is usually outdated at the time of the drop-off. Finally, most TDP approaches employ grid abstractions rather than working directly on the road network [9], making them inadequate for handling the fine-grained spatial requirements of TOP. Notably, TDP typically involves managing a large fleet of taxis, unlike TOP tasks that are often executed by a single agent [19, 9].

2.2 Methods for Solving TOP In recent years, various solutions for TOP were proposed [16, 13, 22, 8] approaching the problem from different angles.

In [16], the authors introduce TOP and propose a simple yet effective greedy heuristic that employs a probabilistic model to determine the next parking spot to visit. Additionally, they transform TOP into a time-varying TSP, which is solved using ant colony optimization.

Several works have utilized RL to tackle TOP [13, 22, 8]. In [8], the authors introduced a RL-based solution for TOP using pointer networks inspired by [2]. Furthermore, TOP has been approached by using attention-based architectures and RL in an attempt to leverage the advantages of attention mechanisms in related tasks like the TSP. The authors of [22] adopted the well-known attention-based architecture from [10] to solve TOP using RL. The authors of [13] take a distinctive approach by formalizing TOP as a SMDP with temporally extended actions. They introduce a state encoder combined with RL to address TOP. Their approach yields comparable performance to previous works in most settings and is only able to outperform them in scenarios with a large number of parking spots.

While we follow their formalization with temporally extended actions, we introduce a novel architecture, enabling our approach to consistently outperform all existing baselines in all settings.

While this paper focuses on the single-agent TOP, the multiple traveling officer problem (MTO), where multiple officers are employed to fine parking violations, has also received attention. A particular challenge in MTO is the coordination among multiple officers. Thus, researchers employed genetic algorithms [12] or RL techniques [17] to tackle the coordination among agents. In [17], the authors utilized RL to solve the MTO. They adapt the state encoder of [13] and introduce several extensions to enable effective officer coordination.

3 The Traveling Officer Problem

In TOP, an officer tries to fine as many parking offenses as possible by traversing a road network $G = (V, E, C)$, where V is a set of vertices (intersections), E is a set of edges (road segments), and $C : E \rightarrow \mathbb{R}^+$ denotes the travel time of the officer for an edge. Each parking spot $p \in P$ is mapped to an edge $e \in E$ of the road network. We refer to the set of parking spots located on edge e as $PE(e)$. The officer observes the state of each parking spot at the current time t . The status indicates whether a parking spot is *free*, *occupied*, in *violation*, or already *fined*. Whenever the officer passes by a parking spot in violation, the officer writes a ticket, and the spot's status is set to *fined*.

We model TOP as a fixed horizon SMDP where the episode length t_{end} corresponds to the duration of the shift of a parking officer. In a SMDP, temporally extended actions are introduced, enabling the agent to take actions that span multiple time steps. This allows treating road segments with varying travel times as actions. In addition, we allow the officer to perform extended actions that follow a path from the officer's current location to any edge hosting a parking spot as in [13].

Formally, a SMDP is a five-tuple (S, A, T, R, γ) , where S is the set of states, A is the set of actions, T is a probabilistic transition function, R is the reward function, and γ denotes the discount factor. Unlike an ordinary Markov Decision Process (MDP), the transition function T also comprises the duration of a state transition τ .

We specify the underlying discrete-time SMDP for TOP as follows:

S represents the set of possible states. Each $s \in S$ contains information about the current daytime, the officer's location loc_o , and the state of each parking spot $p \in P$.

A denotes the set of actions. Following the formulation of [13], the action space corresponds to the subset of all edges E hosting parking spots. Thus, an action $a \in A$ corresponds to traveling from the officer's current position to the end of the associated edge $e_a \in E$. The agent's path can be computed with a dedicated policy. Here, we compute the shortest path w.r.t. the agent's travel time and compute paths with Dijkstra's algorithm. We denote the set of parking spots along the route from the officer's current position to the target of action a as PR_a ¹.

$\mathbf{T}(s_{t+1}, \tau | s_t, a_t) : (S \times \mathbb{R}^+ \times A \times S) \rightarrow [0, 1]$ denotes the state transition probabilities. While the agent's position change is deterministic, the state of parking spots is non-deterministic. A parking spot's state might change due to the officer fining an offense or an unknown stochastic process modeling parking occupancy. The transition function also includes the duration of the transition τ , given by the travel time along the path representing action a .

$R : (S \times A \times S) \rightarrow \mathbb{R}$ denotes the reward function. The agent receives a time-discounted reward $\zeta_t = \sum_{j=0}^{\tau-1} \gamma^j r_{t+j+1}$, where r_{t+j} denotes the reward received at time step $t+j$ during the temporally extended action a with duration τ time-steps. The officer gets a reward of +1 for any parking violation fined along the path corresponding to action a .

The objective of an agent is to maximize the expected time-discounted reward $\mathbb{E} \left[\sum_{t=0}^{t_{end}} \gamma^t r_{t+1} \right]$.

4 A Spatial-Aware TOP Agent

A good policy for TOP requires the agent to consider two aspects: During the current action, the agent has to anticipate the likelihood that parking spots are in violation upon arrival. In addition, the agent has to assess the potential to fine further parking violations during future actions. Our novel architecture effectively covers both aspects by incorporating the spatial relationships between parking locations and the agent's location with a novel pathing module and a future positioning module to assess the potential to fine parking violations within future actions. A schematic overview of our novel architecture, named SATOP, can be found in Figure 1. For readability, we detail the exact parameters of our architecture in the Appendix.

4.1 Input Features Even though the agent observes the current status of the parking spots and its own position, this observation does not satisfy the Markov property. For instance, information about the current park-

¹For the sake of readability, we omit the current location officer loc_o from our notation.

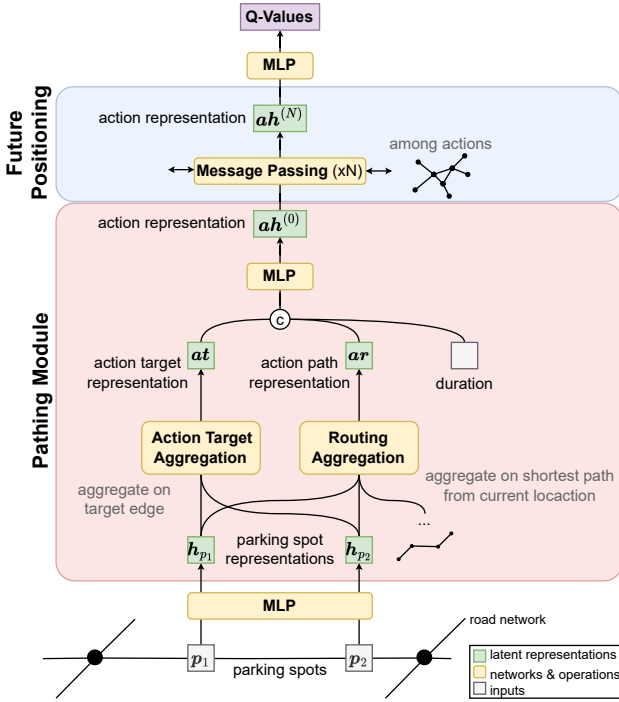


Figure 1: A schematic overview of our novel architecture consisting of the pathing module (red) and the future positioning module (blue). Best viewed in color.

ing duration and the maximum allowed parking duration yield essential information for learning transition probabilities. Therefore, as in [13], we enrich the observation with additional features summarizing the relevant history. Let us note that we explored training LSTM and Transformer-based encoders to learn those features. However, using the handcrafted features generated comparable results while being significantly more computationally efficient. The input to our agent consists of a feature vector for all parking spots $p_i \in P$ describing the state of p_i relative to the agent’s position loc_o and the current time of the day. The input vector for a parking spot p_i at time t contains its current status, either *free*, *occupied*, *violation*, or *fined*, as a one-hot encoding. In addition, we add x and y coordinates of the spot’s location being normalized to the operation area’s bounding rectangle, which helps identify spatially related parking opportunities. Furthermore, we add a normalized timestamp to allow the agent to differentiate between daytimes. To estimate the likelihood of a status change, we provide an additional feature indicating the remaining parking duration for occupied spots. The same feature encodes the time interval a parking spot yields a violation if its status is *violation*. In particu-

lar, a value of -1 to 0 indicates the remaining allowed parking time, while a positive indicator up to a value of 2 indicates how long the parking spot is already in violation. In addition, we provide a feature indicating whether an occupied parking spot might exceed the maximum allowed parking duration within the time the officer would need to reach this parking spot. This way, the agent can learn to travel toward parking spots that are not in violation yet but might be soon. Finally, we add features for the officer’s normalized distance, travel time, and arrival time to the parking spot p_i .

4.2 Parking Spot Representations First, we employ an MLP to generate a latent d_h -dimensional representation h_{p_i} for each parking spot $p_i \in P$ based on its current observation and a d_{le} -dimensional learnable embedding vector. While the weights of the MLP are shared across all parking spots, the embedding vector is not shared to allow the network to learn parking spot-specific representations.

4.3 Pathing Module In our formulation of TOP, each action a corresponds to traveling to a specific action target e_a . Since the agent’s current position varies, the parking spots the agent will pass by when executing an action also vary. Therefore, we compute an encoding for each action a , which captures the information of parking spots along the corresponding path to the action target e_a . All modules in our architecture described in the following either aggregate information to a per-action level or work on per-action representations. The weights of the modules are shared across all actions. To enhance readability, we describe our architecture on a per-action level. In practice, implementations perform the computations for all actions simultaneously using matrix operations. Our novel pathing module consists of several components described in the following.

Action Target Encoder To describe an action a in our setting, we first create a representation of each action target e_a by building a weighted aggregation over the parking spots $p_i \in PE(e_a)$. The action target e_a yields essential information since it represents the agent’s position after action a is performed. Thus, it plays an important role for the future positioning module described in Section 4.4. In particular, we combine the latent representation h_{p_i} of the parking spots $p_i \in PE(e_a)$ on each action’s a target edge e_a into a vector at_a in the following way:

$$(4.1) \quad at_a = \sigma\left(\left(\sum_{p_i \in PE(e_a)} W^{at} h_{p_i}\right) + b^{at}\right),$$

where W^{at} ($d_h \times d_{at}$) and b^{at} denote the parameters of the layer, which are shared among all actions, and $\sigma(\cdot)$

is the activation function.

Route Aggregation Module In addition, we introduce the route aggregation module, where we aggregate the latent representations of parking spots along the path from the officer’s current location loc_o to the action target e_a . In this aggregation, we weight the latent parking spot representations \mathbf{h}_{p_i} by the normalized travel time $\hat{\phi}_a(loc_o, p_i)$ from the officer’s current location loc_o to the parking spot $p_i \in PR_a$ along the route to the action target e_a . Additionally, we scale the normalized travel times using a learnable parameter $\theta \in \mathbb{R}$ that is shared among all actions. This results in a d_h -dimensional path representation \mathbf{ar}_a for each action a that is computed as follows:

$$(4.2) \quad \mathbf{ar}_a = \sum_{p_i \in PR_a} \theta \cdot \hat{\phi}_a(loc_o, p_i) \cdot \mathbf{h}_{p_i}$$

Action Representations Next, we concatenate the path representation \mathbf{ar}_a with the corresponding action target representation \mathbf{at}_a and the expected travel time to the action target $duration_a$ ² for each action a .

We pass this combined information through an MLP, resulting in d_{ah} -dimensional action representations $\mathbf{ah}_a^{(0)}$ which combines all three inputs:

$$(4.3) \quad \mathbf{ah}_a^{(0)} = \text{MLP}_{ah}([\mathbf{ar}_a, \mathbf{at}_a, duration_a])$$

Here, $[\cdot, \cdot, \cdot]$ is the action-wise concatenation operator, and the MLP parameters are shared between all actions.

4.4 Future Positioning Module To estimate how well the agent is positioned after each action, we perform message passing between actions and possible future actions. In this process, the agent is able to aggregate information from potential future actions into each action’s representation utilizing the spatial relationship between them. Therefore, we create a graph structure where nodes contain the latent representations of actions and edges link to possible future actions.

We already created rich representations $\mathbf{ah}_a^{(0)}$ for each action (“node”) in the pathing module. To compute information about the links between actions, we define the edge information between action a and a possible future action a' as $\delta_{a,a'}$. In particular, we employ the travel time and the number of parking spots along the path between e_a and $e_{a'}$ as $\delta_{a,a'}$. Then, we transform each edge information $\delta_{a,a'}$ into an importance factor $\hat{\delta}_{a,a'}^{(l)}$ by passing the information through an MLP

with an output size of 1, followed by a tanh activation to constrain its range:

$$(4.4) \quad \hat{\delta}_{a,a'}^{(l)} = \tanh(\text{MLP}_{\delta}^{(l)}(\delta_{a,a'}))$$

The parameters of this MLP are shared across all links.

In the next step, we use this importance factor to combine the information $\mathbf{ah}_{a'}^{(l-1)}$ from all possible future actions $a' \in A$ for each action a :

$$(4.5) \quad \mathbf{ah}_a^{(l)} = \sigma\left(\sum_{a' \in A} \hat{\delta}_{a,a'}^{(l)} W^{ah,(l)} \mathbf{ah}_{a'}^{(l-1)} + \mathbf{b}^{ah,(l)}\right)$$

$$(4.6) \quad \mathbf{ah}_a^{(l)} = \text{LN}^{(l)}(\mathbf{ah}_a^{(l)} + \mathbf{ah}_a^{(0)})$$

Here, $l \in \{1, \dots, N\}$ denotes the layer index. We share the $(d_{ah} \times d_{ah})$ weight matrix $W^{ah,(l)}$ and $\mathbf{b}^{ah,(l)}$ between all actions and possible future actions but not between layers. For each layer, we apply layer normalization (LN) [1] and incorporate residual connections [7]. We choose ELU [4] as the activation function $\sigma(\cdot)$.

We use two future positioning module layers to enable efficient message passing and parking spot information aggregation across possible future actions. The process ensures that the neural network captures significant spatial and temporal dependencies between possible future actions and parking spots.

4.5 Q-Value Estimation Finally, we employ another MLP with an output size of 1 to reduce the action representation $\mathbf{ah}_a^{(N)}$ of each action a into a final Q-Value, which provides estimates of the expected return associated with each action:

$$(4.7) \quad Q_a = \text{MLP}_Q(\mathbf{ah}_a^{(N)})$$

The weights of the network are shared between actions.

Employing our proposed neural network architecture enables the agent to effectively assess parking violations on the route and anticipate its future positioning after executing actions in TOP, thus allowing the agent to learn effective policies.

4.6 Training Given the formulation of TOP with temporally extended actions, we utilize DoubleDQN [20] adapted to the semi-Markov setting to train the agent. While several state-of-the-art RL algorithms are based on policy gradients [18, 14, 6], their application to our architecture is difficult. Many of these algorithms require learning a compatible shared representation that actor and critic utilize [18]. However, this proved challenging with our architecture because we separated the representation of different actions early on. To still explore these approaches, we conducted experiments training our proposed architecture using algorithms like

²Let us note that $duration_a$ is the computed travel time for the shortest path the agents computed, whereas τ describes the required time of an action provided by the environment.

A Spatial-Aware Deep Reinforcement Learning for the Traveling Officer Problem 67

PPO [14] and SAC [6] on a concatenation of all action representations. Unfortunately, these attempts resulted in sub-optimal performance and training instabilities. Hence, we turned our attention to the DoubleDQN algorithm, which has successfully dealt with TOP [13]. We utilize a replay buffer and update the parameters of our network using a batch of transitions. We minimize the following loss function:

$$(4.8) \quad \mathcal{L}(\Theta) = \mathbb{E}_{s_t, a_t, \tau, r_{t:t+\tau-1}} [(y_t - Q(s_t, a_t; \Theta))^2]$$

where $y_t = \sum_{j=0}^{\tau-1} \gamma^j r_{t+j} + \gamma^\tau Q(s_{t+\tau}, a'_{t+\tau}; \Theta')$. Here, $a'_{t+\tau}$ represents the optimal action with respect to Θ , i.e., $a'_{t+\tau} = \operatorname{argmax}_{a_{t+\tau} \in A(s_{t+\tau})} Q(s_{t+\tau}, a_{t+\tau}; \Theta)$. Θ corresponds to the parameters of the behavior Q network and Θ' represents the parameters of the frozen target Q network, which are periodically updated by copying from Θ .

5 Evaluation

In this section, we provide an extensive experimental evaluation of our novel approach, SATOP. We implement an event-based simulator in C++ for TOP with OpenAI Gym-compatible Python bindings. To replicate real-world conditions, we utilize parking data from Melbourne in 2019, sourced from the city’s open data platform³. To create different graph structures and explore the transferability of hyperparameters, the city of Melbourne is divided into three distinct areas, namely Docklands, Queensberry, and Downtown. We obtain the walking graph from OpenStreetMap⁴. The characteristics of these areas are summarized in Table 1 and Figure 2. To ensure an unbiased evaluation, we split the parking event dataset into a training, validation, and test set. Since parking patterns tend to exhibit weekly trends and to avoid biases introduced through week-days, we account for this by partitioning the dataset by the remainder of dividing the day of the year by 13: If the remainder is 0, the day is included in the test set. If the remainder is 1, we add the day to the validation set. The remaining days are assigned to the training set. We consider each day as an episode and to improve diversity, we shuffle the order of the training days. The officer’s workday spans 12 hours each day from 7AM to 7PM. We set the travel speed of the officer to 5km/h. To accelerate the training process, the agent interacts with several environments simultaneously. We implement our approach within the Tianshou framework [21], which provides a reliable and efficient platform for RL. Hyperparameters were tuned using the

Area	Docklands	Queensberry	Downtown
Nodes	1,435	1,711	6,806
Edges	4,307	5,356	21,369
P. Spots	487	639	1,481
Actions	166	177	493

Table 1: Characteristics of the different areas used in our evaluation. Note that the number of actions is much smaller than the number of parking spots or edges.

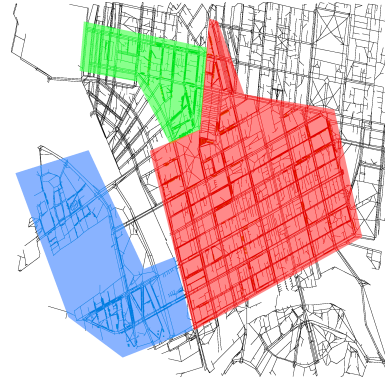


Figure 2: Visualization of the areas used in our evaluation: Queensberry (green), Docklands (blue), and Downtown (red). Best viewed in color.

Docklands area and the resulting hyperparameters were applied to all areas. The test results are generated by using the weights achieving the highest validation results during training. Each training run is executed on a single GPU within a cluster consisting of various GPUs equipped with 24GB to 48GB of GPU memory. The full details regarding the environment parameters, hyperparameters, and training procedures can be found in the Appendix⁵. As a significant part of the research in this domain does not publish their implementation, uses data from different years, and employs varying pre-processing and data splits, our work strives to provide an extensive comparison to related approaches and establish a benchmark in the field. To promote standardization and facilitate future research, we openly publish our framework and baseline implementations⁵.

5.1 Baselines In order to evaluate the performance of our proposed neural network architecture SATOP, we compare it against several state-of-the-art methods commonly used in this domain.

³<https://data.melbourne.vic.gov.au/explore/dataset/on-street-car-parking-sensor-data-2019/information/>

⁴<https://www.openstreetmap.org>

⁵<https://github.com/niklasdbs/satop>

Algorithm	Area		Queensberry		Docklands		Downtown	
	Validation	Test	Validation	Test	Validation	Test	Validation	Test
Greedy [16]	166.59	166.79	223.59	224.96	209.48	208.96		
ACO (0.1s) [16]	192.17	192.36	262.45	261.89	231.86	231.89		
ACO (1.0s) [16]	197.21	196.93	271.24	271.11	261.66	253.57		
SDDQN [13]	185.31	187.07	268.59	267.25	292.41	289.46		
DGAT [22]	159.1	158.64	181.28	176.46	138.59	137.18		
PTR [8]	161.31	159.04	223.55	222.11	150.41	149.39		
SATOP (ours)	209.45	210.89	314.31	310.71	359.0	353.25		

Table 2: Average number of parking violations fined per day across Queensberry, Docklands, and Downtown on the validation and test data.

5.1.1 Greedy The greedy approach, proposed by [16], uses a probability model to determine the next parking spot to visit. It selects the spot with the highest probability of still being in violation upon arrival. The approach assumes that the distribution of parking violations follows an exponential distribution. While this method provides a simple and easy-to-compute solution, it yields surprisingly effective results.

5.1.2 ACO The authors of [16] address TOP by simplifying it to time-varying TSP. By applying ant colony optimization, this approach attempts to find an optimal sequence of visits to different locations. As the ACO approach requires computing the solution during inference, we limit the computation time to 0.1 and 1.0 seconds.

5.1.3 SDDQN [13] developed a state-of-the-art RL approach for TOP. Its performance serves as a benchmark for evaluating the effectiveness of our novel architecture.

5.1.4 PTR The authors of [8] propose simplifying TOP to a variant of the TSP and solve it using pointer networks based on the work of [2].

5.1.5 DGAT In [22], the authors apply the well-known approach from [10] to solve TOP using an attention mechanism. They also simplify TOP to a variant of the TSP.

5.2 Metrics The primary objective of TOP is to fine as many parking violations as possible. Hence, we use the average number of fined violations per day as our primary metric for assessing performance.

5.3 Results In this section, we demonstrate the superior results of SATOP across different areas, namely Queensberry, Docklands, and Downtown. The perfor-

mance of various approaches including Greedy, ACO with different time limits (0.1s and 1.0s), SDDQN, PTR, DGAT, and our proposed approach SATOP are evaluated based on both validation and test sets.

Table 2 displays the average number of fined parking violations per day for each approach in the different areas. The results demonstrate the outstanding performance of our approach, outperforming all other algorithms across all areas on both the validation and test data.

In the Queensberry area, SATOP achieves a significantly better average number of fined violations with a score of 210.89 (test set) and 209.45 (validation set). It outperforms all other methods, with DGAT, PTR, and Greedy trailing far behind with test scores of 158.64, 159.04, and 166.79, respectively. SDDQN achieves a score of 187.07. While ACO with different time limits shows promising results with test scores of 192.36 (0.1s) and 196.93 (1.0s), SATOP is still significantly more effective in this scenario.

In the Docklands area, SATOP continues to show its superiority, achieving a score of 314.31 on the validation set and 310.71 on the test set, easily surpassing all other methods. The closest competitor in this area is the ACO (1.0s), with a score of 271.24. In line with previous research, SDDQN (with a score of 267.25) is unable to outperform the ACO, but it still manages to outperform the Greedy baseline (224.96) significantly.

In the Downtown area, our approach again stands out as the top-performing method, with a score of 353.25 (test set) and 359.0 (validation set). Downtown is the largest and most challenging area. Notably, DGAT and PTR trail far behind in this scenario and exhibit a drop in performance compared to Docklands. A possible explanation for this is that they treat TOP as a variant of the TSP, which is difficult to solve because of the large number of parking spots. ACO (1.0s) also shows a slight decrease in performance (261.66) compared to Docklands (271.24), while SATOP manages to fine

A Spatial-Aware Deep Reinforcement Learning for the Traveling Officer Problem 69

Ablation	Area	Docklands		Downtown	
		Validation	Test	Validation	Test
SATOP (ours)		314.31	310.71	359.0	353.25
No Future Pos. Module		-31.24	-30.07	-32.72	-34.82
Future Pos. Module: Only Travel Time as Link Info (δ)		-3.76	-1.14	-4.76	-7.79
Pathing Module: No Action Target Encoder		-6.10	-5.57	-6.62	-11.36
Pathing Module: No Route Agg. Module		-3.38	-5.21	-7.07	-9.07
No Spot Specific Representations		-5.93	-5.53	-4.38	-7.64

Table 3: Ablations evaluated on the Docklands and Downtown area. We report the reduction in average number of fined violations per day.

significantly more parking violations than in Docklands. SDDQN, our closest competitor, is able to outperform all other baselines with scores of 289.46 (test set) and 292.41 (validation set). Still, our approach is able to fine nearly 64 more parking violations per day on average on the test data, which is an improvement of 22%.

Furthermore, our results reveal that DGAT and PTR perform inferior in all areas. These approaches treat TOP as a variant of the TSP and solve it using RL. In both papers, the authors limited the evaluation to only a small number of parking spots (150 max), which is only 10% of the parking spots in Downtown and around 30% of the parking spots in Docklands (the area with the fewest parking spots). Our results indicate that these approaches have difficulties scaling to real-world scenarios. Furthermore, this underscores the distinct characteristics of TOP and the TSP.

Overall, the results presented in Table 2 provide compelling evidence of the effectiveness of SATOP. Our method consistently outperforms other algorithms across all areas, demonstrating its potential to effectively solve TOP in various settings.

5.4 Ablations We conducted various ablations to assess the influence of the different components within our novel architecture. These ablations were carried out in the Docklands and Downtown area and tested on both the validation and test datasets. To measure the performance, we use the average number of violations fined per day. The results of our ablation study are presented in Table 3. Notably, the outcomes of the ablations are consistent across different areas and both the validation and test datasets.

Removing the future positioning module has a substantial impact on performance. In both Docklands and Downtown, eliminating this module resulted in a sharp reduction of approximately 30 fewer parking violations fined per day. This performance drop was consistent across both the test and validation datasets. In our future positioning module, we make use of

complex edge data. Using only the travel time of the path and omitting the number of parking spots along the path leads to a minor decrease in performance (of around 1 to 8).

Our architecture includes a novel pathing module, which cannot be removed entirely since subsequent modules require a per-action representation. However, we can examine the impact of excluding specific components within this module, namely the action target encoder and the route aggregation module. Our ablation reveals a slight reduction in performance (around 4 to 11 fewer violations fined per day) when leaving out either of these components.

Furthermore, we investigate the impact of removing information that enables the network to differentiate between individual parking spots and learn spot-specific representations. This results in a slight decrease in the average number of parking violations fined (of around 4 to 8).

In summary, our ablation study highlights the importance of the future positioning module in our architecture. Its removal significantly reduces the agent’s ability to fine parking violations effectively. Additionally, while we cannot remove the pathing module in its entirety, its individual components, the action target encoder, and the route aggregation module contribute to the performance, although their exclusion has a comparatively smaller impact. Lastly, maintaining the ability to differentiate between individual parking spots and learn spot-specific representation has a minor performance benefit.

6 Conclusion

In this paper, we presented SATOP, a novel RL-based approach for TOP that incorporates various spatial relationships between parking spots, actions, and the officer’s location. Our novel architecture consists of a pathing module to learn better representations of actions by including parking spots along each action’s path, as well as a future positioning module that allows

the agent to assess its positioning after executing an action by learning inter-action correlations. We evaluated our approach and several state-of-the-art baselines using a simulation environment replaying real-world parking data from Melbourne. Our approach consistently outperformed all competitors regarding fined violations, demonstrating its effectiveness in addressing the challenges of TOP in realistic scenarios.

Our approach opens up several future research directions. First, we plan to extend our spatial-aware architecture to the multi-agent version of TOP, where the coordination between multiple officers poses a significant challenge. Further, we want to apply and extend our architecture to tackle various other spatial tasks beyond TOP. One such example is the VRP, which shares some similarities with TOP but presents its own set of challenges. Finally, we aim to explore scenarios that include dynamically changing routes and travel times due to traffic or other environmental factors.

References

- [1] J. L. BA, J. R. KIROS, AND G. E. HINTON, *Layer normalization*, arXiv preprint arXiv:1607.06450, (2016).
- [2] I. BELLO, H. PHAM, Q. V. LE, M. NOROUZI, AND S. BENGIO, *Neural combinatorial optimization with reinforcement learning*, arXiv preprint arXiv:1611.09940, (2016).
- [3] G. BONO, J. S. DIBANGOYE, O. SIMONIN, L. MATIGNON, AND F. PEREYRON, *Solving multi-agent routing problems using deep attention mechanisms*, IEEE Trans. Intell. Transp. Syst., 22 (2020), pp. 7804–7813.
- [4] D.-A. CLEVERT, T. UNTERTHINER, AND S. HOCHREITER, *Fast and accurate deep network learning by exponential linear units (elus)*, arXiv preprint arXiv:1511.07289, (2015).
- [5] M. GENDREAU, G. LAPORTE, AND R. SÉGUIN, *Stochastic vehicle routing*, Eur. J. Oper. Res., 88 (1996), pp. 3–12.
- [6] T. HAARNOJA, A. ZHOU, P. ABBEEL, AND S. LEVINE, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, in International conference on machine learning, PMLR, 2018, pp. 1861–1870.
- [7] K. HE, X. ZHANG, S. REN, AND J. SUN, *Identity mappings in deep residual networks*, in Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14, Springer, 2016, pp. 630–645.
- [8] R. HE, X. XIAO, Y. KANG, H. ZHAO, AND W. SHAO, *Heterogeneous pointer network for travelling officer problem*, in 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, 2022, pp. 1–8.
- [9] J. KIM AND K. KIM, *Optimizing large-scale fleet management on a road network using multi-agent deep reinforcement learning with graph neural network*, in ITSC, IEEE, 2021, pp. 990–995.
- [10] W. KOOL, H. VAN HOOF, AND M. WELLING, *Attention, learn to solve routing problems!*, arXiv preprint arXiv:1803.08475, (2018).
- [11] M. NAZARI, A. OROOJLOOY, L. SNYDER, AND M. TAKÁC, *Reinforcement learning for solving the vehicle routing problem*, Adv Neural Inf Process Syst, 31 (2018).
- [12] K. K. QIN, W. SHAO, Y. REN, J. CHAN, AND F. D. SALIM, *Solving multiple travelling officers problem with population-based optimization algorithms*, Neural Computing and Applications, 32 (2020), pp. 12033–12059.
- [13] S. SCHMOLL AND M. SCHUBERT, *Semi-markov reinforcement learning for stochastic resource collection*, in Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 3349–3355.
- [14] J. SCHULMAN, F. WOLSKI, P. DHARIWAL, A. RADFORD, AND O. KLIMOV, *Proximal policy optimization algorithms*, arXiv preprint arXiv:1707.06347, (2017).
- [15] W. SHAO, F. D. SALIM, J. CHAN, S. MORRISON, AND F. ZAMBETTA, *Approximating optimisation solutions for travelling officer problem with customised deep learning network*, arXiv preprint arXiv:1903.03348, (2019).
- [16] W. SHAO, F. D. SALIM, T. GU, N.-T. DINH, AND J. CHAN, *Traveling officer problem: Managing car parking violations efficiently using sensor data*, IEEE Internet of Things Journal, 5 (2017), pp. 802–810.
- [17] N. STRAUSS, D. WINKEL, M. BERRENDORF, AND M. SCHUBERT, *Reinforcement learning for multi-agent stochastic resource collection*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2022, pp. 200–215.
- [18] R. S. SUTTON, D. MCALLESTER, S. SINGH, AND Y. MANSOUR, *Policy gradient methods for reinforcement learning with function approximation*, Advances in neural information processing systems, 12 (1999).
- [19] X. TANG, Z. QIN, F. ZHANG, Z. WANG, Z. XU, Y. MA, H. ZHU, AND J. YE, *A deep value-network based approach for multi-driver order dispatching*, in Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 1780–1790.
- [20] H. VAN HASSELT, A. GUEZ, AND D. SILVER, *Deep reinforcement learning with double q-learning*, in Proceedings of the AAAI conference on artificial intelligence, vol. 30, 2016.
- [21] J. WENG, H. CHEN, D. YAN, K. YOU, A. DUBURCQ, M. ZHANG, Y. SU, H. SU, AND J. ZHU, *Tianshou: A highly modularized deep reinforcement learning library*, Journal of Machine Learning Research, 23 (2022), pp. 1–6.
- [22] R. ZHANG, C. YANG, AND X. PENG, *Dynamic graph attention network for traveling officer problem*, in 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, 2022, pp. 1–7.

Appendix

1 Parameters of Environment, Training, and our Architecture

Parameter	Value
Officer Speed	5km/h
Data Year	2019
Working Hour Start	7
Working Hour End	19
γ	0.999
ϵ min	0.01
ϵ -decay	exp
Steps til ϵ -decay start	10000
Steps until min ϵ	5000000
Optimizer	RMSProp
Learning Rate	0.0001
Alpha	0.99
Batch Size	256
Number of Parallel Envs	8
Replay Buffer Size	100000
Reward Transformation	None
Number of Episodes	40 (=8000000 total env steps)
Env Steps per Episode	200000
Start Learning	10000
Train Every Env Steps	32
Target Update Frequency	3125 gradient steps
Parking Spot Encoder (MLP)	256 HDIM, 4 Layers, ELU (no act after last), LN
Parking Spot Rep Dim d_h	256
Parking Spot ID Linear Emb Dim (d_{le})	64
Action Representation Net (MLP_{ah})	1024 HDIM, 4 Layers, ELU, LN, 256 Out Dim
Action Target Hidden Dim (d_{at})	256
Future Pos Hidden Dim d_{ah}	256
Complex Edge Info Net (MLP_{δ})	256 HDIM, 2 Layers, Tanh
Action Target Encoder Activation (σ)	None
Q-Net (MLP_Q)	256 HDIM, 4 Layers, ELU (no act after last), LN
Norm Complex Edge Info	False, True
Future Pos Activation (σ)	ELU
Number of Future Pos Layers (N)	2
Distance Norm Factor	1/3000.0
Duration Norm Factor	1/3000.0
Route Norm Factor (ϕ_a)	1/3000.0
Learnable Parameter Route (θ)	True
Norm Sim Matrix	False

Table 1: Parameters of Environment, Training, and our Architecture

In this section we list the hyperparameters of our approach.

2 Input Features

Feature	Details	Encoding
Current Status	Free, Occupied, In Violation, Fined	One Hot
Optimistic In Violation	-	Boolean
Time of Day	-	0 to 1
Walking Time of Officer	-	Normalized
Arrival Time of Officer	-	Normalized
Distance To Spot	-	Normalized
Occupy/Violation Duration	-	-1 to 2
X and Y coordinates	-	Normalized

Table 2: Parking Spot Features

We briefly give a tabular description of the input features.

B Reinforcement Learning for Multi-Agent Stochastic Resource Collection

Venue 2022 European Conference on Machine Learning and Data Mining

DOI https://doi.org/10.1007/978-3-031-26412-2_13

Declaration of authorships The research idea was proposed, developed, and conceptualized by Niklas Strauss and discussed with all co-authors. The implementations and the experiments were done by Niklas Strauss. The manuscript was written by Niklas Strauss and improved by Max Berrendorf and Matthias Schubert.

Publication

Reinforcement Learning for Multi-Agent Stochastic Resource Collection

Niklas Strauß^[✉]^[0000-0002-8083-7323], David Winkel^[0000-0001-8829-0863], Max
Berrendorf^[0000-0001-9724-4009], and Matthias Schubert^[0000-0002-6566-6343]

LMU Munich

`{strauss,winkel,berrendorf,schubert}@dbs.ifi.lmu.de`

Abstract. Stochastic Resource Collection (SRC) describes tasks where an agent tries to collect a maximal amount of dynamic resources while navigating through a road network. An instance of SRC is the traveling officer problem (TOP), where a parking officer tries to maximize the number of fined parking violations. In contrast to vehicular routing problems, in SRC tasks, resources might appear and disappear by an unknown stochastic process, and thus, the task is inherently more dynamic. In most applications of SRC, such as TOP, covering realistic scenarios requires more than one agent. However, directly applying multi-agent approaches to SRC yields challenges considering temporal abstractions and inter-agent coordination. In this paper, we propose a novel multi-agent reinforcement learning method for the task of Multi-Agent Stochastic Resource Collection (MASRC). To this end, we formalize MASRC as a Semi-Markov Game which allows the use of temporal abstraction and asynchronous actions by various agents. In addition, we propose a novel architecture trained with independent learning, which integrates the information about collaborating agents and allows us to take advantage of temporal abstractions. Our agents are evaluated on the multiple traveling officer problem, an instance of MASRC where multiple officers try to maximize the number of fined parking violations. Our simulation environment is based on real-world sensor data. Results demonstrate that our proposed agent can beat various state-of-the-art approaches.

Keywords: Multi-Agent RL · Navigation · Deep RL

1 Introduction

In many sequential planning tasks, agents travel on a transportation network, like road or public transportation networks, to reach certain points of interest (POIs) to earn rewards. One way to differentiate these tasks is according to the time intervals for which POIs grant rewards and whether these intervals are known to the agents. For example, for the traveling salesman and the basic vehicular routing problem (VRP), reaching POIs grants rewards regardless of the time they are visited. In more sophisticated tasks such as windowed VRPs [11], POIs only grant rewards during given time windows that are known to the agent. In contrast, in applications like taxi dispatching and ride-sharing, the agent does not

2 Strauß et al.

know in advance at which time intervals rewards can be earned. Thus, policies try to guide the agents into areas where collecting rewards is more likely, i.e., passengers might show up.

The task of Stochastic Resource Collection (SRC) [21] assumes that resources have fixed locations and change their availability based on an unknown random process. Thus, the agent observes currently collectible resources and can try to reach these before the resources are not collectible anymore. An instance of the SRC task is the TOP [22] in which a parking officer is guided to fine a maximal amount of parking offenders. The setting is based on the assumption that information about parking sensors is available from sensors registering the duration of parking events. As offenders might leave before the officer arrives, not all resources remain collectible, and thus, agents have to consider the chance of reaching resources in time. [21] model SRCs as Semi-Markov Decision Processes (SMDP) and propose an action space that lets the agent travel to any resource location on a pre-computed shortest path. To find effective policies maximizing the number of collected resources in a given time interval, a reinforcement learning (RL) algorithm based on deep Q-Networks (DQN) is proposed. Though the proposed method learns successful policies for single agents, it often requires more than one agent to handle sufficiently large areas. Thus, [18] propose a multi-agent heuristics for guiding multiple officers in a larger area. As RL methods already showed better performance than known heuristic methods in the single-agent case, it makes sense to examine multi agent reinforcement learning (MARL) methods to improve policies. However, known MARL approaches usually are not designed for Semi-Markov models where agents' actions require varying amounts of time. In addition, they often require mechanisms that counter the problem of the size of the joint action space, which grows exponentially with the number of agents, and the credit assignment problem when using joint rewards. Though there are several methods to counter each of these problems, most of them do not consider the properties of the MASRC environments with asynchronous agent actions in a Semi-Markov environment.

In this paper, we formalize MASRC as a selfish Semi-Markov Game (SMG). We adapt the action space of [21] to let each agent target any resource in the network. Thus, agents generally terminate their actions in varying time steps. We propose a selfish formulation where each agent optimizes its own individual rewards. We argue that a group of independent agents still optimizes the sum of collected resources sufficiently well as the agents learn that evading other agents decreases the chances of another agent collecting close-by resources. We empirically verify our reward design by comparing it to joint rewards. To approximate Q-values, we propose a neural network architecture that processes information about resources, agents, actions, and the relation between them. To combine these types of information, we employ attention and graph neural network mechanisms. This way, our agent can estimate the likelihood of reaching a collectible resource before it becomes uncollectible or another agent reaches the resource first. Furthermore, our resource embedding considers the spatial closeness of additional collectible resources to make actions moving the agent into a

region with multiple collectible resources more attractive. To evaluate our new approach, we developed a multi-agent simulation based on real-world parking data from the city of Melbourne. Our experiments demonstrate superior performance compared to several baselines [18] and (adaptions of) state-of-the-art approaches [21,1,9]. We compare our methods with heuristic methods proposed in [18], an adaption of the single-agent SRC method from [21] and an architecture proposed for dynamic multi-agent VRP [1] which is based on the well-known single-agent architecture [9]. We evaluated the last benchmark to demonstrate that state-of-the-art solutions for the dynamic VRP do not sufficiently cope with the additional stochasticity of MASRC problems. To further justify the design choices in our architecture, we provide ablations studies. To conclude, we summarize the contributions of our paper as:

- A formulation of the MASRC as a Semi-Markov Game building a solid theoretical foundation for the development of MARL approaches
- A novel architecture for learning rich state representations for MARL
- A scalable simulation environment for the multi-agent traveling officer problem (MTOP) problem based on real-world data

2 Related Work

In this section, we review work on related tasks routing an agent through spatial environments to collect rewards. In addition, we will discuss general multi agent reinforcement learning approaches.

2.1 Stochastic Resource Collection

One of the most recognized routing tasks in the AI community is the *vehicular routing problem (VRP)* where a group of agents needs to visit a set of customer locations in an efficient way. There exist various variations of the VRP [4] and some of them include the appearance of new customers during the day [1]. In contrast to SRC, the setting does not include customers disappearing after an unknown time interval. This is a decisive difference as it makes the reward of an action uncertain. In recent years, several approaches have been developed to solve the vehicular routing problem or some of its variations using DRL [1,9,16,17]. MARDAM [1] is an actor-critic RL-agent - based on [9] - designed to solve VRP with multiple agents using attention mechanisms. While state and action spaces of dynamic VRP and MASRC can be considered as very similar, the behavior of the environment is not. To demonstrate these differences, we compare to an agent using the architecture of [1] in our experiments.

There exist various papers on *multi-agent taxi dispatching* [8,12,13,28,32] which can be formulated as a MASRC task. However, in most settings there are significant differences to MASRC as the resources are usually not claimed at arrival. Instead, customers are assigned to close-by taxis the moment the guest publishes a request to the dispatcher. Thus, reaching the guest in time is

4 Strauß et al.

usually not considered. Furthermore, to the best of our knowledge, only a single approach works directly on the road network [8]. All other approaches work on grid abstractions which are too coarse for MASRC. Finally, taxi dispatching tasks usually involve large and time variant sets of agents. To conclude, known solutions to taxi dispatching are not applicable to solve MASRC.

The *traveling officer problem (TOP)*, first described by [22] is an instance of SRC. In [21], the authors propose an Semi-Markov RL-based agent to solve the single-agent TOP task and name other tasks that can be formulated as SRC. Later on, the authors of [18] study the MTOP. They propose a population-based encoding, which can be solved using various heuristics for optimization problems like cuckoo search or genetic algorithms. Additionally, they propose a simple greedy baseline that assigns idling officers to the resource in violation using "first-come-first-serve". Competition between officers is handled by assigning a collectible resource to the officer with the highest probability that the resource is still in violation when the officer arrives.

2.2 Multi-Agent Reinforcement Learning

After reviewing solutions to similar tasks, we will now discuss general multi agent reinforcement learning (MARL) approaches w.r.t. their suitability for training on MASRC environments. In MARL, a group of agents shares the same environment they interact with. There are various challenges in MARL: the non-stationarity of the environment from the perspective of an individual agent, the exponentially increasing joint action space, the coordination between agents, and the credit assignment problem. A plethora of different approaches to tackle these challenges exists [6] and we will give a brief overview of the most important MARL approaches in the following.

Joint action learners reduce the multi-agent problem to a single-agent problem by utilizing a single centralized controller that directly selects a joint action. While joint action learners can naturally handle coordination and avoid the non-stationarity, in practice, these approaches are often infeasible because of the exponential growth of the joint action space w.r.t. the number of agents [7].

On the opposite site, we can use multiple independent learners [26]. The agents interact in parallel in a shared environment using a single-agent RL algorithm. In many cases, it has been shown that independent learners can yield strong performance while allowing for efficient training. However, in some settings, independent learners can suffer from the non-stationarity of the environment induced by simultaneously learning and exploring agents.

In recent years, approaches have been developed that utilize centralized training and decentralized execution (CLDE). In [24], the authors presented VDN that decomposes the joint action-value function as a sum of the individual agents' Q-function values obtained solely from the agents' local observation. The authors of [19] propose QMIX, a method that extends VDN by learning a non-linear monotonic combination of the individual Q-functions, which allows representing a larger class of problems. The authors of [3] propose a counterfactual multi-agent actor-critic method (COMA) that uses a centralized critic that allows

estimating how the action of a single agent affects the global reward in order to address the credit assignment problem.

Another way to tackle the problem of coordination between agents is to facilitate communication between the agents. CommNet [23] is a prominent approach that learns a differentiable communication model between the agents. Both CLDE and communication-based approaches suffer from the credit-assignment problem, which we mitigate through our individual reward design.

A drawback of the named approaches when applied to MASRC is that these algorithms do not consider *temporal abstractions*, i.e., actions with varying duration. The application of temporal abstraction to CLDE requires the modification of the problem in a way that the decision epochs are synchronized or experience needs to be trimmed [27]. This way of training is inefficient as it exponentially increases the number of decision epochs with respect to the number of agents. The authors of [2,14,5,20,27] investigate temporal abstraction in multi-agent settings. The authors of [20] first introduce different termination schemes for actions with different temporal abstractions that are executed in parallel. [14] propose independent learners to efficiently handle the asynchronous termination setting, [27] adapt CommNet and QMIX to a setting with temporal abstraction, while [2] propose a version of COMA in decentralized settings with temporal abstractions. Let us note that some of these approaches, like COMA, QMIX, or CommNet, can be adapted to train our function approximation and thus, can be applied to MASRC. We experimented with these approaches but could not observe any convincing benefit for solving MASRC. In addition, the use of those methods tries to learn complex coordination schemes between agents. However, in MASRC agents basically cannot directly support each other as the only action impacting other agents is collecting resources.

3 Problem Formulation

We consider the problem of MASRC, where n agents try to maximize the collection of resources in a road network $G = (V, E, C)$, where V is a set of nodes, E denotes a set of edges and $C : E \rightarrow \mathbb{R}^+$ are the corresponding travel costs. Each resource $p \in P$ is located on an edge $e \in E$ in the road network. Whether a resource p is collectible can be observed by the agents but might change over time. The state changes of resources follow an unknown stochastic process. Whenever an agent passes a collectible resource, the resource is collected by the agent.

Formally, we model the MASRC problem as a Semi-Markov Game (SMG) $\langle I, S, \mathbf{A}, P, R, \gamma \rangle$, where I is a set of agents indexed by $1, \dots, n$, S is the set of states, \mathbf{A} denotes the joint action space, P is the transition probability functions, R denotes the reward functions of the individual agents, and γ is the discount factor.

Agent: A set of n agents moving in road network and collecting resources.

State: $\mathbf{s}_t \in S$ denotes the global state of the environment at time t . The exact information included in the state depends on the actual instantiation, e.g.,

6 Strauß et al.

TOP. Nonetheless, all MASRC tasks share a common structure that can be decomposed into resources, agents, and environment:

- *Resources* characterized by the current status, e.g., availability and position.
- *Agents* defined by their position and ID.
- *Environment* with features such as the time of the day or an indication of holidays.

Action: $\mathbf{a}_t \in \mathbf{A} = A_1 \times \dots \times A_n$: is the joint action at time t . Following the single-agent formulation of [21], we define the individual action space A_i of an agent to correspond to the set of edges E , i.e., the agent will travel on the shortest path to the corresponding edge. This allows to focus on the MASRC task itself rather than solving the routing problem, where high-performance deterministic algorithms are available. Therefore, the individual actions have varying duration, depending on the agent’s position and target location. As a result, agents may have to asynchronously select actions at different decision times. Between those decision times, agents continue to their target. Formally, we can reduce this to a synchronous setting, and thus the given joint action space, by introducing a special "continue" action, as described in [14].

Reward: Each agent i has an independent reward function $R_i \in R$, where $R : (S \times \mathbf{A}) \rightarrow \mathbb{R}$. Each agent i independently tries to maximize its own expected discounted return $\mathbb{E} \left[\sum_{j=0}^{\infty} \gamma^j r_{i,t+j} \right]$. Each agent’s individual reward function corresponds to the resources collected by the agent itself. The reward is incremented by 1 for each collected resource. A resource is collected when an agent passes a collectible resource.

State Transition Probability: With

$$(\mathbf{s}_{t+1}, \tau \mid \mathbf{s}_t, \mathbf{a}_t) : (S \times \mathbb{R}^+ \times \mathbf{A} \times S) \rightarrow [0, 1] \subset \mathbb{R} \quad (1)$$

we denote the probability of transitioning to the state \mathbf{s}_{t+1} from the current state \mathbf{s}_t by taking the joint action \mathbf{a}_t . Although, some effects of an action are deterministic (e.g., the positions of the agents), the state changes of resources are uncertain and the exact dynamics are unknown. Unlike in a Markov Game, in a SMG, we additionally sample the number of elapsed time-steps τ of the action \mathbf{a}_t . The smallest feasible temporal abstraction is the greatest common divisor of all edge travel times. The duration is determined by the individual action $a_i \in \mathbf{a}$ with the shortest duration, which is a multiple of the smallest feasible duration. As a result, an agent receives a time-discounted reward $\zeta = \sum_{j=0}^{\tau-1} \gamma^j r_{t+j+1}$.

4 Method

In this section, we introduce our novel multi-agent RL agent. At first, we provide some insight into our reward design. Secondly, we present our training procedure that is based on independent DQN [25]. After that, we describe the inputs to our architecture and name the particular features for our evaluation on the MTOP task. Finally, we introduce our novel function approximator for the MASRC problem that facilitates coordination between the agents.

4.1 Individual Rewards

In general, the goal of MASRC is to maximize the expected joint reward $\mathbb{E} [\sum_{i=0}^{|I|} \sum_{j=0}^{\infty} \gamma^j r_{i,t+j}]$. However, we decided to use individual rewards to avert the credit assignment problem, which leads to a Markov Game where agents act selfishly. In literature, the impact of such selfish behavior is commonly denoted as the "price of anarchy" [10]. In the context of MASRC, we argue that the price of anarchy is likely to be very low and outweighed by the benefits of having a reward function that allows the agents to assess the impact of their actions more directly and thus mitigates the credit assignment problem. This is because in MASRC helping other agents directly is not possible. Therefore, coordination boils down to not getting in the way of other agents. There might be cases where a joint reward might lead to policies where particular agents would target far-off resources decreasing their own but increasing the sum of collected resources. However, we observed in our experiments that these cases are rare. We provide an empirical evaluation of our reward design choice compared to joint rewards in Section 6.3.

4.2 Training

Independent DQN [25] combines independent learners [26] and DQN [15]. To speed up learning, we share the network parameters between agents and distinguish them by their IDs [31]. Independent learning provides a natural way to handle settings with asynchronous termination [14]. In independent learning, each agent treats the other agents as part of the environment. However, this may lead to sub-optimal coordination between the agents. To mitigate this problem, we introduce an architecture that allows each agent to efficiently reason about the intents of other agents. We utilize a DoubleDQN [29] adapted to the Semi-Markov setting. We update the network parameters with respect to a batch of transitions collected from all agents by minimizing the following loss function:

$$\mathcal{L}(\Theta) = \mathbb{E}_{\mathbf{s}_t, a_t, \tau, r_{t:t+\tau-1}, \mathbf{s}_{t+\tau}} [\text{loss}(y_t, Q(\mathbf{s}_t, a_t; \Theta))] \quad (2)$$

where $y_t = \sum_{j=0}^{\tau-1} \gamma^j r_{t+j} + \gamma^\tau Q(s_{t+\tau}, a'_{t+\tau}; \Theta')$. The action $a'_{t+\tau}$ is the optimal action w.r.t to Θ , i.e., $a'_{t+\tau} = \operatorname{argmax}_{a_{t+\tau} \in A(s_{t+\tau})} Q(s_{t+\tau}, a_{t+\tau}; \Theta)$. Θ denotes the parameters of the behavior Q network and Θ' denotes the parameters of the frozen target Q network which are periodically copied from Θ . To improve clarity, we omitted the indices indicating the individual agents. We use a smooth L1 loss.¹

4.3 Input Views

In the following, we will briefly describe the inputs to our function approximation and name the particular features for our evaluation on the MTOP task.

¹ cf. <https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html>

8 Strauß et al.

Resource Features We encode each resource from the perspective of each individual agent separately. To this end, we add features describing the relation of the agent to the resource, e.g., the distance or arrival time. This results in n times different views of each resource. The resource view for the MTOP contains a one-hot-encoding of the resource’s current status, i.e., free, occupied, in violation, or fined. Additionally, we provide a flag that indicates whether a parked car would be in violation if it remains parked and the officer would directly go there. Finally, we add the current time of the day, walking time, agent arrival time, and distance to the resource. All these features are normalized. We add a real-valued number between -1 and 2, indicating how long a car is still allowed to occupy the resource and how long it is in violation, respectively. A score greater than zero indicates a violation. Finally, we add the normalized coordinates of the resource’s position.

Agent Features For MTOP, it consists of a one-hot encoding of the agents’ ID, the normalized coordinates of its current position and target, as well as the normalized walking time and distance to its target.

Spatial Relation To capture the spatial interaction between the resources, we create a distance matrix for each agent. There is one row in the matrix for each action consisting of the network distance of the action target to each resource, the distance between the agent, and the action target to each row.

4.4 Architecture

An effective policy in MASRC requires an agent to consider the complex interaction between resources, actions, and other agents to estimate the likelihood of reaching a *collectible* resource. To capture those dependencies, our novel architecture first encodes the action-level intents of each agent using the resources and their spatial relationship solely from the perspective of each individual agent, i.e., ignoring the other agents. We call this module the *Shared Action Encoder*. After that we continue by combining the perspective of the current agent with the action-level intents of the other agents using multi-head attention in the *Intent Combination Module*. This allows an agent to assess the likelihood that another agent catches collectible resources first. While the inputs are different, the parameters of all networks are shared between all agents.

Shared Action Encoder In the context of SRC, the value of an action, i.e., the likelihood of reaching resources in time, depends largely on the state of resources near the target [21]. We argue that in a multi-agent setting, the simple distance weighting from [21] is not expressive enough to capture the complex dependency of an action’s value on, e.g., the uncertainty of reaching the resource in time. Thus, we propose an extended *Shared Action Encoder* to calculate agent-specific action embeddings, based upon the agent’s features, and resources’ features, as well as the distances. We provide the pseudo-code roughly following PyTorch style in Fig. 2, and show an overview in Fig. 1. We begin by transforming the agent’s features with an MLP (cf. line 3). Next, we calculate unnormalized agent-specific resource to action relevance scores combining information from the agent

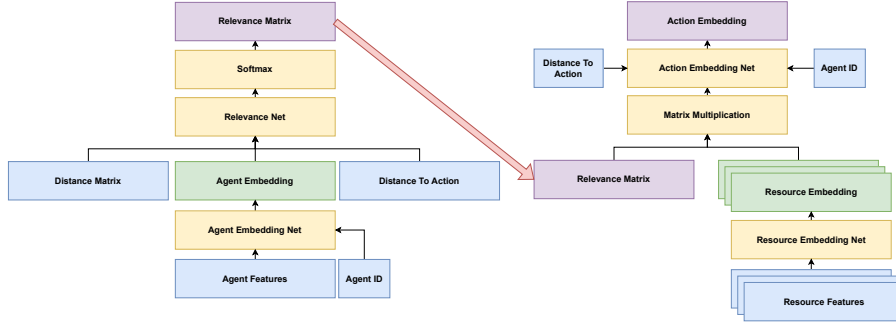


Fig. 1: Conceptual overview of the *Shared Action Encoder*. This module creates a rich representation for each action based on the resource states from the perspective of an individual agent. The module captures the spatial relationship of resources around each action’s target using a graph neural network mechanism. Networks and operations are colored yellow, the output of the module and crucial intermediate representations are purple, while blue denotes input features.

```

def sea(feat_ag, feat_res, i_ag, dist, dist_ag2ac): 1
    """Shared action encoder for a single agent.""" 2
    x_ag = mlp1(feat_ag) 3
    # shape: (dim_ag,) 4
    rel_act_res = mlp2(cat(broadcast([ 5
        x_ag[None], 6
        dist_ag2ac[None], 7
        dist, 8
    ]), dim=-1)) 9
    # shape: (n_action, n_res) 10
    rel_act_res = softmax(rel_act_res, dim=-1) 11
    # shape: (n_action, n_res) 12
    x_res = mlp3(feat_res) 13
    # shape: (n_res, dim_res) 14
    x_act = rel_act_res @ x_res 15
    # shape: (n_act, dim_res) 16
    return mlp4(cat(x_act, i_ag, dist_ag2ac)) 17

```

Fig. 2: Pseudocode for the *Shared Action Encoder* following PyTorch style. `feat_ag` denotes the agent’s features, `feat_res` the (agent-specific) resource features, `i_ag` the agent’s ID, and `dist` the action-resource distance matrix, and `dist_ag2ac` the distance from the current agent to all actions (which are target edges). `mlp1` to `mlp4` are separate MLPs.

10 Strauß et al.

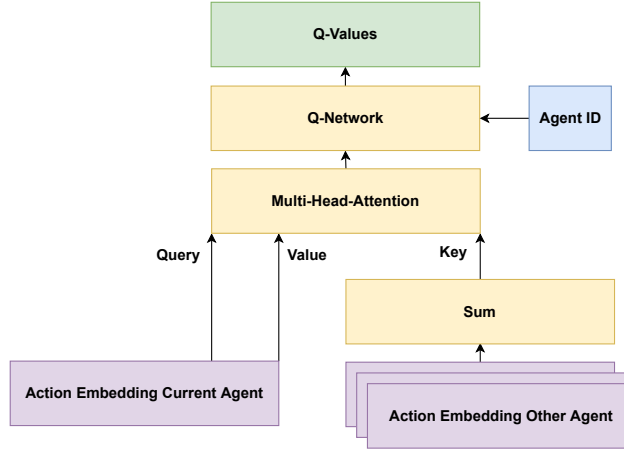


Fig. 3: In the *Intent Combination Module*, we enrich the action embedding of a single agent with information about the other agents’ actions using multi-head attention. Afterwards, we reduce the enriched action representations to a Q -value for every action with an MLP. Networks and operations are colored yellow, inputs coming from the *Shared Action Encoder* are purple, and blue denotes input features.

representation, the distance from the agent to the action (i.e., edge), and the action-to-resource distance matrix using another MLP (cf. lines 5-9). These relevance scores are subsequently normalized using the softmax operator (cf. line 11). The resource features are first transformed by an MLP, before we use the previously computed relevance scores for aggregating them per action (cf. line 13-15). A final MLP combines this information with distance to the agent as well as the agent ID (cf. line 17). In the following, we denote the result of this component as \mathbf{E} .

Intent Combination Module Information about the other agents’ intents is crucial in multi-agent settings - thus, we propose an attention-based mechanism to update an agent’s action representations by considering the ones of other agents. Let \mathbf{E}_i denote the output of the shared action encoder for agent i . For scalability, we first aggregate the latent actions of all other agents $\bar{\mathbf{E}}_i := \sum_{j \in I \setminus \{i\}} \mathbf{E}_j$. Next, apply a multi-head attention mechanism [30] with \mathbf{E}_i as query and value and $\bar{\mathbf{E}}_i$ as key. Finally, we reduce every row of the result, corresponding to the co-agents-aware action representation, to a single Q -value using an MLP. This MLP also receives the agent ID as an additional input to allow diversification of the agents.

Area	Nodes	Edges	Resources	Edges with Resources
Docklands	1,435	4,307	487	166
Queensberry	1,711	5,356	639	177
Downtown	6,806	21,369	1,481	493

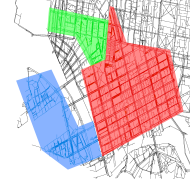


Fig. 4: Description and illustration of the different areas used in our evaluation: Docklands (blue), Downtown (red), and Queensberry (green). Notice that typically only a small fraction of edges contains resources and there can be more than one resource per edge.

5 Simulator Design

To the best of our knowledge, there is no publicly available simulation for MTOP. To enable effective training of reinforcement learning agents, we implement a simulator that can replay real-world sensor data and parking restrictions, which allows us to simulate as close as possible to the real world. The walking graph, i.e., road network, is extracted from OpenStreetMap². We assign parking spots to the closest edge in the graph. When an agent passes a resource in violation, it will be fined. The time for fining a violation is set to zero in our simulation. The agent collects a reward of +1 for every fined resource. All agents start at the same place every day. They work for 12 hours from 7 am to 7 pm. Each agent has a walking speed of 5km/h.

For our evaluation, we use openly available on-street parking sensor data and parking restrictions from the city of Melbourne in 2019³. We divide Melbourne into three areas to study different graph structures and hyperparameter transferability. Details regarding the areas can be found in Fig. 4. Each run was trained using a single GPU on a cluster consisting of RTX A6000 (48GB) and A100 (40GB) GPUs. The code of our simulation and agents is publicly available.⁴

6 Experimental Evaluation

We split the parking event dataset into a training, validation, and test set. Parking follows weekly patterns. To avoid biases introduced through weekdays, we split the dataset as follows: If the remainder of the day in the year divided by 13 is 0, we add the day to the test set. In case the remainder is 1, we add the day to the validation set. The remaining 308 days are added to the training set. An episode is equivalent to a working day. The order of the training days is shuffled. To speed up training, the agent interacts with eight environments in parallel.

The transferability of the hyperparameters across different regions and numbers of agents is important. We tuned the hyperparameters in a single area

² <https://www.openstreetmap.org>

³ <https://data.melbourne.vic.gov.au/browse?tags=parking>

⁴ <https://github.com/niklasdbs/masrc>

12 Strauß et al.

(Docklands) with two agents. Agents were trained using early stopping. The test results reported are with respect to the best validation results. The full hyperparameter setting can be found in the supplement.

6.1 Baselines

Greedy We modify the greedy baseline from [18] for better performance: Instead of assigning agents to the resource with the earliest violation time, we directly use the catching probability from the tie-breaking mechanism.

LERK The authors of [18] propose to solve the MTOP by representing it using leader-based-random-key encoding (LERK) and then solve it using various classical heuristic solvers developed for combinatorial issues. One of these heuristics that yielded the best performance was the genetic algorithm, which we have implemented.

MARDAM [1] is an actor-critic RL-agent - based on [9] - designed to solve dynamic-VRP with multiple agents using attention mechanisms. While they propose a method to transform the underlying Markov game into a sequential MDP, this transformation is not possible for MASRC tasks. Therefore, we train their architecture using independent actor-critic. Due to the dynamic state of resources in MASRC tasks, we need to calculate the customer-embeddings (i.e., resource), in every step using a Transformer which is computationally expensive and memory intense. As a result, we are not able to train the agent on full episodes and need to rely on bootstrapping.

SASRC We train the architecture of [21] that has been proposed for the SASRC using independent learning with shared independent learners. We add the agent-id to the final network so that agents can differentiate their behavior. Additionally, we add information about the targets of all agents to the resources, which allows the agent to incorporate information about other agents and thus benefits learning [31].

6.2 Results

As the evaluation metric, we use the average number of violations fined per day. We evaluate in three different areas using two, four, and eight agents. The results in Table 1 show that our proposed approach can surpass the other approaches and baselines in various regions and across different numbers of agents. We can beat MARDAM, a state-of-the-art algorithm designed for multi-agent dynamic-VRP, by a large margin. This underlines that approaches for SRC tasks need to be able to handle the increased stochasticity. Moreover, MARDAM requires a massive amount of GPU memory due to the use of the transformer encoder in large settings like Downtown with eight agents. For this setting, our approach

Table 1: Average number of violations fined per day in Docklands, Queensberry, and Downtown for 2, 4, and 8 agents on the validation and test set.

Area	Algorithm	2 Agents		4 Agents		8 Agents	
		Validation	Test	Validation	Test	Validation	Test
Docklands	Greedy	186.93	192.67	304.32	300.22	442.57	439.33
	LERK	244.78	245.11	328.61	330.56	424.32	418.19
	MARDAM	339.79	336.37	418.96	416.44	482.57	479.78
	SASRC	343.82	304.19	476.07	465.52	551.86	544.63
	OURS	388.32	379.59	527.21	518.52	588.04	580.00
Queensberry	Greedy	180.46	189.15	240.5	250.07	277.54	286.56
	LERK	192.18	198.78	233.86	245.07	260.79	271.67
	MARDAM	225.18	229.85	247.29	255.41	257.04	267.81
	SASRC	222.61	231.41	257.11	266.00	263.79	273.15
	OURS	244.43	255.41	271.39	281.59	284.75	294.37
Downtown	Greedy	138.79	144.63	256.14	257.33	429.93	435.22
	LERK	213.57	219.19	298.32	305.40	430.18	428.19
	MARDAM	340.54	342.37	469.18	471.26	255.82	260.93
	SASRC	425.68	418.48	657.61	658.04	815.68	815.41
	OURS	495.07	494.70	710.75	713.3	866.04	867.93

uses approximately 16 times less GPU memory during training. As a result, the batch size needs to be reduced for those settings, which may impact performance. Additionally, the episode length in MASRC tasks is much longer than in a typical VRP, which makes learning on whole episodes impossible. Furthermore, the experiments show that our approach yields considerably better results than existing heuristic solvers designed for the MTOP, such as LERK, which require intensive computational resources at inference time. While our approach requires several days of training it only needs a few milliseconds at inference time. The authors of LERK [18] state a runtime of 4.67 minutes for making a single decision with seven officers using their fastest approach. This makes the application of their algorithm in real-world settings infeasible.

6.3 Ablation Studies

To assess the individual components' impact on the final performance, we provide several ablation studies. We conduct the ablations with two agents on the validation set in the Docklands area and report the average number of violations fined per day. The results of the ablations can be found in Table 2, where they are sorted decreasingly by performance, i.e., the highest impact is on the right.

We observe that not using an *action embedding network* has the strongest impact on the performance resulting in an 8.3% reduction in performance. Since the reduction is less severe when removing inputs of this network, the effect can be primarily attributed to the additional non-linear transformation after resource aggregation. Switching from individual to joint rewards is next in terms of relevance. We observe that using *joint rewards* performs considerably worse,

Table 2: Ablations performed in the area of Docklands with two agents. We report the average number of caught violations per day. The second row shows the relative performance compared to the base configuration. The values are sorted decreasingly, i.e., the highest impact is on the right.

	OURS	With Other Agents' Target	Without Agent ID	Without Resource Position	Without Agent Embedding Network	Without Distance To Action	Joint Reward	Without Action Embedding Network
absolute	388.32	378.54	375.39	370.36	365.25	360.96	359.71	356.18
relative	100.0%	97.5%	96.7%	95.4%	94.1%	93.0%	92.6%	91.7%

leading to a 7.4% reduction in performance,⁵ which we attribute to the credit assignment problem. Ignoring the *distance to the action* leads to a reduction of 7.0%. Without this information the agent lacks input to assess the inherent reward uncertainty in far actions. The *agent embedding network* is the next crucial component, with a reduction of 5.9%. Without it, the model cannot utilize the agent features, such as its position. Not having access to the *agent ID* aggravates diversification of agent policies and leads to a performance decrease of 3.3%. Finally, *adding other agents' target information* to the agent-specific views of the resources leads to slightly worse performance of around 2.5%, despite yielding improvements in the SASRC baseline. This indicates that our architecture can already sufficiently incorporate the intents of other agents for effective coordination.

7 Conclusion

In this work, we have formalized Multi-Agent Stochastic Resource Collection (MASRC) as a Semi-Markov Game, providing a solid theoretical framework for the development of new approaches. We further proposed a novel architecture to solve MASRC tasks featuring an innovative intent combination model which permits re-assessment of action representations based on the other agents' action representations. To enable evaluation, we introduced an efficient agent-based simulation for the MTOP task, for which we publish the source code to support the community in future research. Using the simulation, we could demonstrate that our approach is able to beat existing heuristic baselines, adaptations of state-of-the-art single-agent SRC solutions, and approaches for the multi-agent dynamic-VRP in terms of fined violations. On a more fundamental level, our results indicate that existing approaches for multi-agent dynamic-VRP struggle to handle the increased dynamics in MASRC tasks, and thus MASRC requires specialized solutions. In future work, we want to include dynamic travel times.

⁵ Notice though that even with joint rewards, our approach is able to beat baselines trained with individual rewards.

Furthermore, we want to investigate the transfer of trained policies between different areas and numbers of agents. Finally, we will research further scaling our approach to very large graphs.

8 Acknowledgments

We thank the City of Melbourne, Australia, for providing the parking datasets used in this paper and Oliver Schrüfer for contributing the implementation of LERK. This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

References

1. Bono, G., Dibangoye, J.S., Simonin, O., Matignon, L., Pereyron, F.: Solving multi-agent routing problems using deep attention mechanisms. *IEEE Trans. Intell. Transp. Syst.* **22**(12), 7804–7813 (2020)
2. Chakravorty, J., Ward, N., Roy, J., Chevalier-Boisvert, M., Basu, S., Lupu, A., Precup, D.: Option-critic in cooperative multi-agent systems. *arXiv preprint arXiv:1911.12825* (2019)
3. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: *AAAI*. vol. 32 (2018)
4. Gendreau, M., Laporte, G., Séguin, R.: Stochastic vehicle routing. *Eur. J. Oper. Res.* **88**(1), 3–12 (1996)
5. Han, D., Boehmer, W., Wooldridge, M., Rogers, A.: Multi-agent hierarchical reinforcement learning with dynamic termination. In: *Pacific Rim Int’l Conf on Artificial Intelligence*. pp. 80–92. Springer (2019)
6. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A survey and critique of multiagent deep reinforcement learning. *Auton Agent Multi Agent Syst* **33**(6), 750–797 (2019)
7. Hu, J., Wellman, M.P., et al.: Multiagent reinforcement learning: theoretical framework and an algorithm. In: *ICML*. vol. 98, pp. 242–250. Citeseer (1998)
8. Kim, J., Kim, K.: Optimizing large-scale fleet management on a road network using multi-agent deep reinforcement learning with graph neural network. In: *ITSC*. pp. 990–995. IEEE (2021)
9. Kool, W., Van Hoof, H., Welling, M.: Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475* (2018)
10. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: *Annual symposium on theoretical aspects of computer science*. pp. 404–413. Springer (1999)
11. Kumar, S.N., Panneerselvam, R.: A survey on the vehicle routing problem and its variants (2012)
12. Li, M., Qin, Z., Jiao, Y., Yang, Y., Wang, J., Wang, C., Wu, G., Ye, J.: Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In: *The world wide web conference*. pp. 983–994 (2019)
13. Liu, Z., Li, J., Wu, K.: Context-aware taxi dispatching at city-scale using deep reinforcement learning. *IEEE Trans. Intell. Transp. Syst.* (2020)
14. Makar, R., Mahadevan, S., Ghavamzadeh, M.: Hierarchical multi-agent reinforcement learning. In: *Proc. of the fifth Int’l Conf on Autonomous agents*. pp. 246–253 (2001)

16 Strauß et al.

15. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
16. Nazari, M., Oroojlooy, A., Snyder, L., Takác, M.: Reinforcement learning for solving the vehicle routing problem. *Adv Neural Inf Process Syst* **31** (2018)
17. Peng, B., Wang, J., Zhang, Z.: A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems. In: *International Symposium on Intelligence Computation and Applications*. pp. 636–650. Springer (2019)
18. Qin, K.K., Shao, W., Ren, Y., Chan, J., Salim, F.D.: Solving multiple travelling officers problem with population-based optimization algorithms. *Neural Computing and Applications* **32**(16), 12033–12059 (2020)
19. Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., Whiteson, S.: Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: *ICML*. pp. 4295–4304. PMLR (2018)
20. Rohanimanesh, K., Mahadevan, S.: Learning to take concurrent actions. *Adv Neural Inf Process Syst* **15** (2002)
21. Schmoll, S., Schubert, M.: Semi-markov reinforcement learning for stochastic resource collection. In: *IJCAI*. pp. 3349–3355 (2021)
22. Shao, W., Salim, F.D., Gu, T., Dinh, N.T., Chan, J.: Traveling officer problem: Managing car parking violations efficiently using sensor data. *IEEE Internet of Things Journal* **5**(2), 802–810 (2017)
23. Sukhbaatar, S., Fergus, R., et al.: Learning multiagent communication with back-propagation. *Adv Neural Inf Process Syst* **29** (2016)
24. Sunehag, P., Lever, G., Gruslyns, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., et al.: Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:1706.05296 (2017)
25. Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., Vicente, R.: Multiagent cooperation and competition with deep reinforcement learning. *PloS one* **12**(4), e0172395 (2017)
26. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: *ICML*. pp. 330–337 (1993)
27. Tang, H., Hao, J., Lv, T., Chen, Y., Zhang, Z., Jia, H., Ren, C., Zheng, Y., Meng, Z., Fan, C., et al.: Hierarchical deep multiagent reinforcement learning with temporal abstraction. arXiv preprint arXiv:1809.09332 (2018)
28. Tang, X., Qin, Z., Zhang, F., Wang, Z., Xu, Z., Ma, Y., Zhu, H., Ye, J.: A deep value-network based approach for multi-driver order dispatching. In: *Proc. of the 25th ACM SIGKDD*. pp. 1780–1790 (2019)
29. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: *AAAI*. vol. 30 (2016)
30. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. *Adv Neural Inf Process Syst* **30** (2017)
31. Zheng, L., Yang, J., Cai, H., Zhou, M., Zhang, W., Wang, J., Yu, Y.: Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In: *AAAI*. vol. 32 (2018)
32. Zhou, M., Jin, J., Zhang, W., Qin, Z., Jiao, Y., Wang, C., Wu, G., Yu, Y., Ye, J.: Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching. In: *Proc. of the 28th ACM Int'l Conf on Information and Knowledge Management*. pp. 2645–2653 (2019)

Supplementary Material

Niklas Strauß^[0000-0002-8083-7323], David Winkel^[0000-0001-8829-0863], Max Berrendorf^[0000-0001-9724-4009], and Matthias Schubert^[0000-0002-6566-6343]

LMU Munich

{strauss,winkel,berrendorf,schubert}@dbs.ifi.lmu.de

1 Hyperparameters

Table 1: Parameters of our architecture and environment

Parameter	Value
γ	0.999
ϵ min	0.01
ϵ -decay	exp
Steps until min ϵ	1500000
Start Learning	5000
Optimizer	RMSProp
Learning Rate	0.0001
Alpha	0.99
Target Update Frequency	2500/5000 gradient steps
Batch Size	256 (128 Downtown 8 Agents)
Train Every	32
Replay Size	250000
Number Of Parallel Envs	8
Reward Transformation	tanh
Number of Env Steps	25/50 * 10e6
Prioritized Replay	False
Early Stopping	True
Resource Encoder (MLP3)	512 HDIM, 2 Layers, ReLU
Agent Encoder (MLP1)	256 HDIM, 2 Layers, ReLU, Output 512
Relevance Network (MLP2)	512 HDIM, 4 Layers, ReLU
Action Encoder (MLP4)	512 HDIM, 4 Layers, ReLU
Q-Network	512 HDIM, 6 Layers, ReLU no act after last
Number Of Attention Heads	8
Relevance Normalization	Softmax
Gradient Clipping	False/2.0
Other Agent Reduction	sum
Walking Speed	5km/h

2 Input Features

2 Strauß et al.

Feature	Details	Encoding
Status	Free, Occupied, In Violation, Fined	One Hot
Optimistic In Violation	-	Boolean
Current Time Of Day	-	0 to 1
Walking Time of Agent	-	Normalized
Arrival Time of Agent	-	Normalized
Distance to Resource	-	Normalized
Occupy/Violation Duration	-	-1 to 2
X and Y Coordinates	-	Normalized

Table 2: Resources Features from the perspective of an agent

Feature	Details	Encoding
Agent ID	-	One Hot
Current Position	Coordinates	Normalized
Target Position	Coordinates	Normalized
Distance to Target	-	Normalized
Walking Time to Target	-	Normalized

Table 3: Agent Features

C A Comparison of Ambulance Redeployment Systems on Real-World Data

Venue 2022 IEEE International Conference on Data Mining Workshops (ICDMW)

DOI <https://doi.org/10.1109/ICDMW58026.2022.00010>

Declaration of authorships The research idea was proposed, developed, and conceptualized by Niklas Strauss and discussed with all co-authors. Niklas Strauss did the main part of the implementation, and the students implemented some of the baselines. The manuscript was written by Niklas Strauss and improved by the co-authors.

Publication

A Comparison of Ambulance Redeployment Systems on Real-World Data

Niklas Strauß Max Berrendorf Tom Haider Matthias Schubert
MCML, LMU Munich *MCML, LMU Munich* *Fraunhofer-Institut für Kognitive Systeme IKS* *MCML, LMU Munich*
 strauss@dbs.ifi.lmu.de max.berrendorf@gmail.com tom.haider@iks.fraunhofer.de schubert@dbs.ifi.lmu.de

Abstract—Modern Emergency Medical Services (EMS) benefit from real-time sensor information in various ways as they provide up-to-date location information and help assess current local emergency risks. A critical part of EMS is dynamic ambulance redeployment, i.e., the task of assigning idle ambulances to base stations throughout a community. Although there has been a considerable effort on methods to optimize emergency response systems, a comparison of proposed methods is generally difficult as reported results are mostly based on artificial and proprietary test beds. In this paper, we present a benchmark simulation environment for dynamic ambulance redeployment based on real emergency data from the city of San Francisco. Our proposed simulation environment is highly scalable and is compatible with modern reinforcement learning frameworks. We provide a comparative study of several state-of-the-art methods for various metrics. Results indicate that even simple baseline algorithms can perform considerably well in close-to-realistic settings. The code of our simulator is openly available at <https://github.com/niklasdbs/ambusim>.

Index Terms—Urban Simulation, Public Health, Dynamic Ambulance Redeployment, Reinforcement Learning

I. INTRODUCTION

In recent years, the amount of real-time sensor data has increased dramatically, providing more accurate insights into complex urban systems. These data sources offer unprecedented opportunities to improve mobility-centered systems and processes for smarter cities. One particular example are emergency response systems, where sensor data can help to assess emergency risks, estimate hospital occupation rates, or provide up-to-date location data of incidents and ambulances [1].

For many life-threatening conditions, such as cardiac arrest or stroke, the likelihood of survival depends critically on an ambulance arriving in time. Sensor data can prove particularly valuable in emergency response systems as it allows optimization of these systems based on real-time information. To ensure a timely treatment of patients, a plethora of research has been conducted to help Emergency Medical Service (EMS) providers optimize their infrastructure and operations and thus, reduce response time, i.e., the time an ambulance needs to arrive at an incident [2]. In most communities, ambulances are stationed at fixed locations, so-called base stations, from which they are dispatched to incidents by the EMS operator. After providing pre-hospital care at an incident, the ambulance either directly returns to a base station or transports the patient to a hospital before going to a base station. One problem of particular interest is dynamic ambulance redeploy-

ment (DAR) [3]–[5] which refers to the task of dynamically redistributing the ambulances to different base stations after completely handling an incident. Thus, DAR controls the number of available ambulances at each base station while utilizing real-time information. Although efficient policies for DAR have been studied in literature, an objective comparison of the proposed methods is difficult due to the use of proprietary datasets. Moreover, these datasets are often artificial, e.g., making explicit assumptions about the distributions of incidents, without a direct link to real-world data. In this paper, we provide a novel benchmark simulation based on real-world data and provide a comprehensive evaluation of the various DAR methods. In particular, our contributions are as follows:

- We propose a new benchmark simulation environment for DAR based on openly accessible data across 20 years of real emergency data from the city of San Francisco. The event-based simulation is highly-scalable and written in C++, but also offers Python bindings following the OpenAI Gym API [6], an established standard in the field of reinforcement learning.
- We provide implementations of various state-of-the-art DAR methods, as well as several competitive baselines.
- We conduct a benchmark study of these methods using various metrics. We provide evidence that simple baselines perform surprisingly well in close-to-realistic settings.

Our paper is structured as follows: In Section II, we introduce the problem setting before reviewing existing work in Section III. We continue to describe our simulation environment in Section IV, and the dataset in Section V. In Section VI we present our comparative study before we summarize our work in Section VII.

II. PROBLEM SETTING

In this section, we describe the process of how the EMS operates. A visual outline of this process can be found in Figure 1. When the EMS operator receives a call, an idle ambulance is dispatched from a base station and travels to the incident site. Usually, the closest available ambulance will be dispatched. In case there is no ambulance available, the incident will be handled as soon as an ambulance becomes available again. Once the ambulance arrives at the emergency site, it provides pre-hospital care to the patient. If necessary, the patient will subsequently be transported to a hospital.

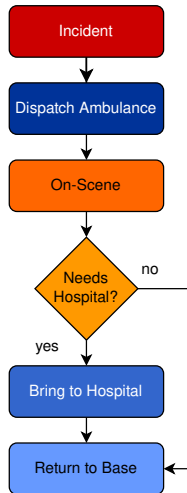


Fig. 1. Simplified schematic overview of the modeled EMS process. Blue boxes denote decision points handled by the dispatch, hospital and redeploy policy.

Otherwise, the ambulance will be directly redeployed to a base station. When the ambulance arrives at the base station, it becomes idle and is ready to be dispatched again.

The state of this simulation contains information about ambulances, base stations, hospitals, and incidents, as well as global information like the current time of the environment. In the following, we provide a detailed description:

a) Ambulances: The state contains information about the activity (e.g., idle), current location, destination, and estimated arrival time at the destination for each ambulance.

b) Base Stations: Base stations have a fixed location where ambulances wait for incidents. We do not limit the number of ambulances per station.

c) Hospitals: Each hospital has a specific location and an unlimited capacity of patients it can handle.

d) Incidents: Each incident contains the following information: time of call, location of the incident, whether the patient has been transported to a hospital, the time spent at the incident, and the time spent at the hospital. Whenever an incident occurs and no ambulance is available, the incident gets queued and will be rescheduled when an ambulance becomes available again.

During this process, the EMS operator employs three policies to make decisions:

- *Dispatch policy:* Deciding from which base station an ambulance should be dispatched to an incident is denoted as the ambulance dispatch problem (ADP). In practice, dispatching the closest available ambulance is prevalent [2].
- *Hospital policy:* This task is about selecting a hospital to transport a patient to if deemed necessary [2]. Selecting the closest hospital is the default policy here as well.
- *Redeploy policy:* Select the base station to redeploy an ambulance after completely handling an incident [1].

In this paper, we focus on the redeploy policy as it is the only task that is not time critical. For example, dispatching an ambulance from a base station further away directly affects the time until pre-hospital care is provided to patient, while redeploying is not time critical. Thus, ambulance redeployment yields sufficient room for optimization. Redeployment can be solved by assigning a fixed base station for each ambulance. In this case, the problem is referred to as ambulance location problem (ALP) [7]. In contrast, there is the DAR [3] task, which facilitates real-time information to select base stations fitting to the current situation. An overview of existing methods for both approaches is given in the next section. During simulation, the policies are employed to select actions that result in new events, which are added to the priority queue of the simulation.

III. RELATED WORK

Motivated by the vital importance of emergency services, optimizing their operations has been a research focus for many years. While the focus of this paper is the DAR, there exists a number of approaches that optimize other parts of the EMS process, e.g., ambulance dispatching. For an overview of these related tasks, we refer the reader to the recent survey of [2]. In the following, we review related work in the field of (dynamic) ambulance redeployment and EMS simulation.

A. Ambulance Redeployment

There is a considerable amount of literature on the DAR. We continue by giving an overview of the most important approaches.

1) Static ALP: Early work in ALP focused primarily on static redeployment models, i.e., each ambulance is assigned to a fixed home base to which it is deployed to whenever it becomes idle [8]–[11]. These static models can be used to determine where to locate base stations, as well as the number of ambulances for each base station [3].

In a recent study on static redeployment models [7], the authors compared several static ALP models and found that the maximum expected covering location problem (MEXCLP) [9] and expected response time model (ERTM) [7] perform best with respect to commonly considered criteria like response times. Both models search for a static policy to distribute a limited number of ambulances over a set of possible base stations. Ambulances are modeled to be busy with a pre-determined probability (busy fraction). The resulting optimization problem can be solved using integer linear programming.

2) DAR: A disadvantage of static models is that they do not make use of real-time information like the current availability of ambulances or time-dependent shifts in incident frequencies at different locations, e.g., due to daily or seasonal patterns. As a result, recent research focuses on dynamic redeployment models. Some of the approaches proposed to make use of lookup tables [12], i.e., an optimal configuration is pre-computed for each possible system state and a dispatcher tries to redeploy the ambulances such that they match the configuration suggested by the lookup table. A major downside of these

approaches is the workload (ambulance movements) required to ensure that the system is in compliance with the lookup table. Another class of approaches models the randomness in the system explicitly, either through a dynamic programming formulation [13], [14] or heuristic approaches [3]. However, an exact dynamic programming formulation is intractable. As a result, approximate dynamic programming approaches have been proposed [15], [16]. Yet, the authors of [3] note that these approaches are still impractical due to the need for an expert to implement these. Additionally, the performance highly depends on the base functions used, and thus the approach is unlikely to work well in general settings [3]. In [3], the authors propose dynamic MEXCLP, an efficient dynamic heuristic based on MEXCLP. To the best of our knowledge, [5] is the only approach published that uses (deep) reinforcement learning (RL) to solve the DAR. It utilizes a demand forecast based on historical averages as well as the current distribution of ambulances over base stations.

B. Simulation

As direct evaluation of different policies is difficult, e.g., due to ethical concerns, simulators have been widely used [3], [15], [17]–[22]. However, existing simulators have either important drawbacks, are not publicly available [15], [17], [21]¹, do not publish the dataset used [15], [18], [21], or are not able to replay real-world incident events [3], [18], [19], [22].

For example, the simulator of [23] does not support dynamic redeployment policies and is not suited for RL, while the simulator of [22] is unable to replay real-world incidents. The simulator of [24] is one of the very few simulators that are openly available, can replay real-world incidents, and support dynamic redeploy policies. However, a major drawback of this simulator is that it is not well suited for (deep) RL because it does not have the structure necessary to train RL agents, which require making observations, deciding on an action thereupon and receiving information about the next state and (immediate) reward.

A further problem that hinders research in the area is the lack of openly available real-world datasets [2]. As a result, most simulators use synthetic or closed-source data.

IV. SIMULATION ENVIRONMENT

In this section, we introduce the architecture of our high-performance simulator. The simulator allows to examine a variety of scenarios, and thus gain valuable insights about various policies. This includes varying the number of ambulances, changing the location and number of base stations, or even removing certain hospitals.

A. General Architecture

The simulation implements the general problem setting described in Section II. It is implemented as a discrete event-based simulation, i.e., around a priority queue that stores events with their time. The queue is initialized by adding all incidents from the real-world data as an event. Subsequently,

the simulator handles the events in chronological order. Certain events require making dynamic decisions and thus add new events to the queue. For example, when an incident occurs, the dispatch policy needs to be called and events for traveling to the incident need to be created as well as events for detailing the pre-hospital care. Since the ambulances operate in a road network, routing and traveling in this network plays a vital role in modeling the EMS process. In a pre-processing step, we map the locations of base stations, hospitals, and incidents to the nearest node in a road network graph obtained from OSM². Since we are replaying real-world events, all incident locations are known in advance. This enables us to accelerate the simulation by pre-computing the travel times between: base stations and incidents, incidents and stations, incidents and hospitals, and hospitals to base stations.

B. Policies

Since the simulation is built to support research in the field of improved policies for EMS, each of the three policies for ambulance dispatching, selecting a hospital, and redeploying an ambulance is separated into extensible modules with a clear API, which allows exchanging policies easily. For our comparative study, we focus on DAR and thus provide multiple implementations of redeployment policies covering simple heuristics as well as state-of-the-art methods. For the other two tasks, we implemented common baseline policies.

1) *Dispatch Policies*: The first module is the dispatch policy, i.e., this module decides by which ambulance an incident should be handled. We implement the prevalent nearest station policy, which dispatches an available ambulance from the station with the shortest travel time to the incident [2].

2) *Hospital Policies*: In some (but not all) cases, a patient needs to be transported to a hospital after the on-site care is finished. In our simulation, the decision of whether a patient is transported to a hospital is based on real-world data. The decision to which hospital a patient is transported is decided by the hospital policy module. In practice, a patient is often taken to the closest hospital [2], which we also implemented as the default policy.

3) *Redeployment Policies*: The main focus of our study is on dynamic redeployment policies. The redeployment policy decides on the base station the ambulance is sent to whenever an ambulance finishes handling an incident. We implement various policies that are commonly used for (dynamic) ambulance redeployment. These include simple heuristics, commonly used SOTA static and dynamic redeployment models, as well as a RL based approach.

- *Random Baseline*: This very simple baseline redeploys the ambulance to a random station. Thus, it provides insight into whether an intelligent policy is necessary in the first place.
- *Least Ambulances*: This baseline deploys the ambulance to the station that currently has the least ambulances [5]. Ambulances already heading towards the station are also

¹ [21] was initially public, however all links are dead.

²<https://www.openstreetmap.org/>

considered when determining the station with the least ambulances.

- *Nearest Base Station*: This baseline deploys the ambulance to the station that is closest to its current position [5].
- *MEXCLP* is a very popular static ambulance location model by [9] that tries to maximize a coverage criteria by solving an integer linear program. It is considered to be one of the best performing static models [3], [7].
- *ERTM* [7], the expected response time model, is a SOTA static ambulance redeployment model. In contrast to *MEXCLP*, it does not optimize a coverage criteria but instead minimizes the expected response time.
- *DMEXCLP* is an efficient dynamic redeployment heuristic that utilizes real-time information [3]. *DMEXCLP* sends the ambulance to the base station maximizing the marginal coverage according to the *MEXCLP* model.
- *DRLSN* is a RL-agent for the DAR task. It utilizes a demand forecast based on historical averages as well as the current distribution of ambulances over base stations [5].

C. Redeployment Agents

The simulation has a particular focus on efficiently supporting RL. In RL, an agent learns a policy by interacting with the environment, i.e., the simulator, in discrete time steps. The interaction between the agent and the environment follows a certain schema: At each time step, the agent receives an observation and selects an action that is passed back to the environment. The simulator then transitions to a new state and returns an observation and a reward signal to the agent. Since the predominant language for (deep) RL research is Python, we implement the redeployment agents in Python and implement the field standard for interfacing simulation and agents, the OpenAI Gym API [6]. Thereby, we combine the benefits of using C++ for efficiently running a complex simulation with the benefits of using Python for deep learning and RL. We utilize `pybind11` [25] to create Python binding of our C++ simulator.

D. Observation Encoders

Different agents base their redeployment decision on different information and require to encode this information in a special format, the so-called observation. Hence, our simulator specifies a flexible and efficient API to create an observation based on the state of the simulator (e.g., location and status of ambulances, current time, travel times, ...) and additional data sources such as IoT data or historical demand averages. For example, a RL agent may require a complex encoding of the state as a floating point tensor, while some heuristics only require the number of ambulances in each station. We encapsulate the complete state of the environment in an object that is passed to the observation creator to build the observation. A benefit of this structure is that it allows for the easy integration of various data sources like urban IoT data.

V. DATASET

A major problem that hinders research is the lack of an openly available dataset containing real-world incidents [2]. Using real-world data instead of sampled synthetic incidents is crucial to realistically evaluate different approaches without biases introduced by making simplifying assumptions. Furthermore, this also has the potential to learn better policies using deep RL models that can exploit statistical patterns that are not present in the simplified synthetic datasets.

In San Francisco, the fire department handles medical 911 calls and the operation of the EMS service. The city publishes a dataset containing these non-police 911 calls³. To the best of our knowledge, this dataset has not been used for the DAR task. This dataset includes, among other things, the date and time when the call is received, when a unit arrives on scene, the time when a unit begins the transport to the hospital, if applicable, as well as the arrival time at the hospital and the time it becomes available again. From these data, we are able to infer whether a patient has been transported to a hospital, the time spent on-scene providing pre-hospital care, as well as the duration the ambulance spent at the hospital. The location of the incident is obfuscated either to the midblock, intersection, or call box to protect the privacy of the caller.

A preprocessing step is required to infer some of the aforementioned data, filter out non-ambulance entries, and a very small amount of invalid entries, e.g., when the order of the process times are invalid. Additionally, we restrict the incidents to the area depicted in Figure 2 to exclude incidents at the airport. We utilize 20 years of incidents from 2001 until 2021. In this time period, a total of 1,616,559 incidents occurred. The number of incidents in each year is detailed in Figure 4. In order to accurately assess the performance of different models, we split the dataset into training, validation, and testing as follows: We assign the first 18 years of data from 2001 to 2019 as the training set, and use 2020 and 2021 for validation and testing, respectively. For our simulation, additional preprocessing is required. Among other things, this includes converting time formats and pre-computing travel times. Some decision policies also require additional information like historical demand averages or demand weights and points. We define the demand locations to be a Voronoi decomposition around the base stations. Following [7], we use the average number of calls per year as the demand weights. The validation set is used to calculate these demand weights.

The road network graph and travel times are obtained from open street maps. The road network spans over an area of around 184 square kilometers and contains 9,586 nodes and 26,762 edges. All incidents, stations, and hospitals are mapped to the closest node in the network.

In order to realistically simulate an EMS system, the location of the base stations and hospitals are required. Unfortunately, these are not published on the open data portal. We will publish the location of stations and hospitals, as well as

³<https://data.sfgov.org/Public-Safety/Fire-Department-Calls-for-Service/nuek-vuh3>

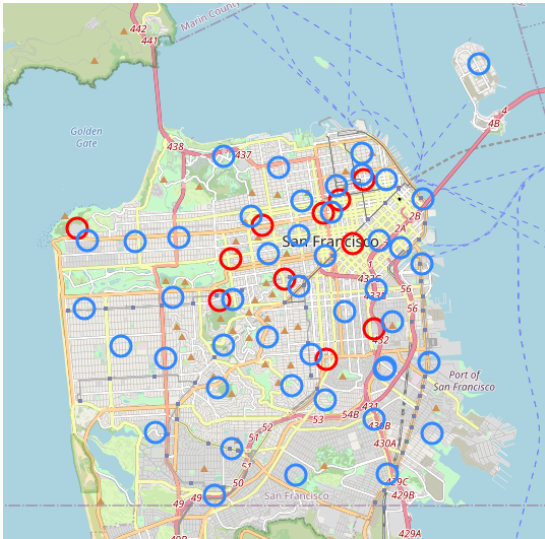


Fig. 2. Locations of base stations (blue) and hospitals (red).

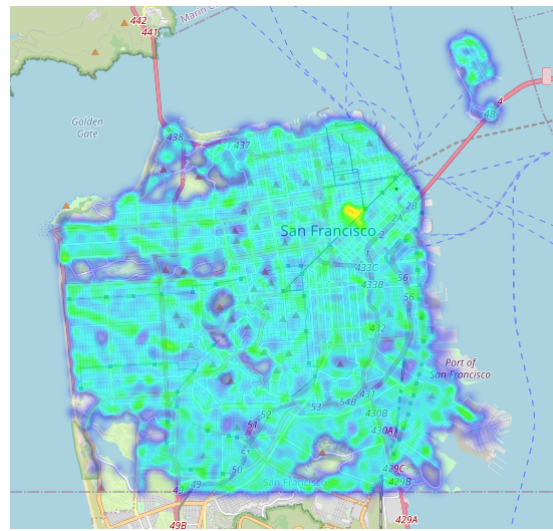


Fig. 3. Spatial distribution of incidents in San Francisco displayed as a heatmap. Lower numbers of incidents are more blueish while higher numbers are more yellowish. Best viewed in color.

the preprocessed dataset. In San Francisco, ambulances are based in 45 fire stations. The location of the fire stations have been extracted from the website of the San Francisco fire department. According to *San Francisco Emergency Medical Response*⁴, the EMS operators transport patients to 11 hospitals (within the city of San Francisco) whose locations have been manually obtained. The locations of stations and hospitals are visualized in Figure 2.

In the following, we outline the most important characteristics of the dataset. The spatial distribution of incidents is visualized in Figure 3. Figure 4 details the number of incidents per year, ranging from around 64,000 calls in 2012 up to 97,000 in 2019. Finally, Figure 5 shows the incidents per month without a clear pattern. Additionally, the dataset contains different temporal patterns. The distribution of incidents by day of the week is illustrated in Figure 6 showing more frequent incidents at the weekend, while Figure 7 shows the varying frequency of incidents during the day, with a peak frequency in the evening and the lowest frequency in the late night.

VI. EVALUATION

In this section, we first briefly detail the performance of our simulator before evaluating the implemented baselines on the San Francisco dataset. In order to ensure a fair comparison across all algorithms, at the start of the simulation all ambulances are uniformly distributed over the base station. Hyperparameters are selected on the validation set with respect to the average response time.

A. Simulator Performance

In this section, we evaluate the speed of our simulator. This is particularly important for training RL agents, as they typically require many interactions with the simulator, and hence

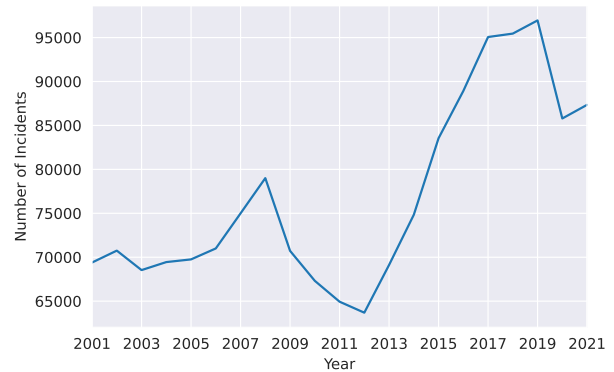


Fig. 4. The number of incidents in each year.

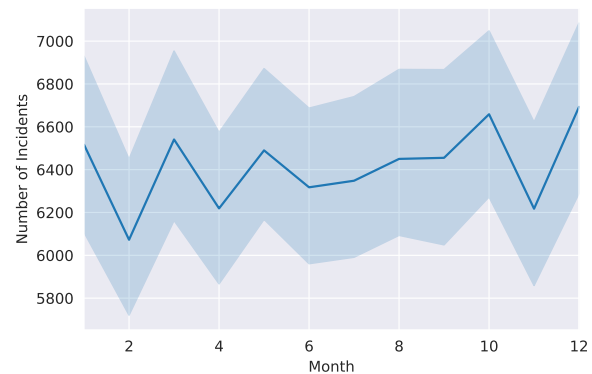


Fig. 5. Number of incidents in each month. The shaded area shows the 95% percentile obtained by bootstrapping.

⁴<http://sfemergencymedicalresponse.weebly.com>

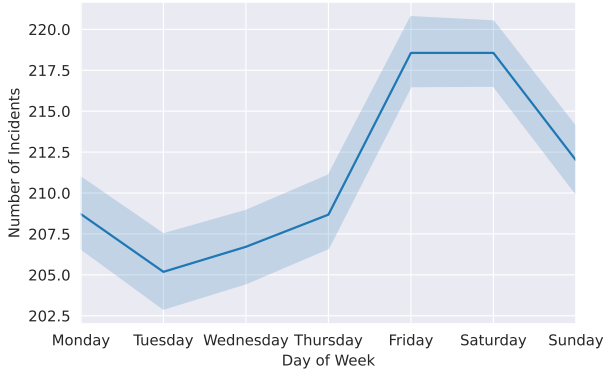


Fig. 6. Number of incidents per weekday. The shaded area shows the 95% percentile obtained by bootstrapping.

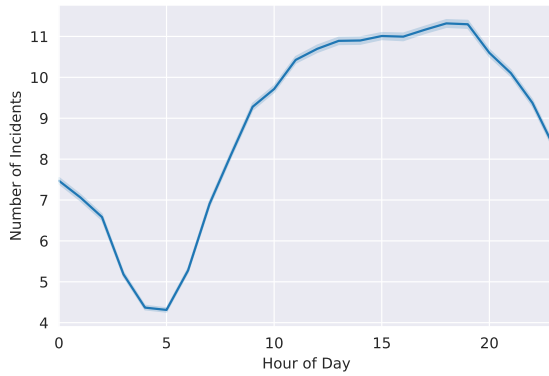


Fig. 7. The hourly number of incidents over the course of the day. The shaded area shows the 95% percentile obtained by bootstrapping.

a high-performance simulator can dramatically reduce training time. Note that comparing absolute runtime is difficult [26], and thus, the following comparison should be taken with a grain of salt.

According to [24] their simulator is one of the fastest EMS simulators available. In their paper, they report a runtime of 3.0295 seconds for simulating one year for a smaller setting with only 4 hospitals, 14 stations, and 16 ambulances on an Windows 10 computer with a 2.4 GHZ Intel i7-5500U CPU and 8GB RAM.

After warmup and with 20 ambulances, our simulator is able to simulate a whole year with around 90,000 incidents in approx 0.72 seconds using the nearest base station agent. We measured the time on a system with an AMD Ryzen 7 2700X Eight-Core Processor at 3.7GHz and 64 GB RAM on Ubuntu 20.04.2 LTS with a 5.4 Linux kernel and Python 3.8.10. Notice that the simulation runs single-threaded, i.e., does not benefit from the multi-core processor. Even though the hardware is different, this indicates that our simulator might be faster, and in particular, is well-suited for a RL application.

B. Metrics

There are various ways to evaluate the performance of an EMS system. The most common metrics, however, are the *average response time* and the *x minute pick-up ratio*. The average response time is the time it takes on average from when a call happens until the ambulance arrives at the incident location. The *x minute pick-up ratio* is the relative frequency of an ambulance arriving at latest after *x* minutes. This is also in line with regulatory requirements in many communities. For example, in San Francisco, an ambulance should arrive within 10 minutes at an incident. The EMS operators target is to handle more than 90% of incidents within that quota. Therefore, our simulator collects these two metrics, which are commonly used in practice and literature [7].

C. Comparison of Redeployment Algorithms

We compare several redeployment algorithms on the real-world dataset from the city of San Francisco. For this comparison, we set the number of ambulances to 15. We evaluate the approaches with respect to the average response time and the ratio of pickups within 10 minutes. The comparison results are detailed in Table I and Table II respectively.

Interestingly, the nearest base station model performs best with respect to the pickup ratio within 10 minutes. Its pickup ratio is 96% on the test set. The MEXCLP, DMEXCLP, ERTM, and DRLSN models are able to achieve marginally worse results ranging from 93% to 95%. The random agent performs worst with 85% followed by the least ambulances baseline with 89%. Noticeably the random agent performance is still relatively high, indicating that there is a sufficient number of ambulances and the incidents do not follow a strongly imbalanced pattern.

The ERTM model achieves the best average response times, around 234 seconds on the test set. The nearest base station baseline performs surprisingly well with 237 seconds. Moreover, the RL agent is not able to achieve the best performance, however, the gap is not very large: Its performance is around 12 seconds worse than the ERTM model. Another interesting finding is that the MEXCLP agent is able to achieve a slightly better average response time than its dynamic heuristic (DMEXCLP). However, when comparing the pickup ratios, DMEXCLP offers marginally better performance.

In line with previous research [7], we can confirm the strong performance of ERTM. Surprisingly, the very simple nearest base station is able to achieve very good results and can beat all baselines with respect to the 10 minute pickup ratio. Furthermore, its average response times are only slightly worse than ERTM. Overall, we are surprised that the DRLSN is not able to beat the other approaches. This is in contrast to previous results reported on other (not publicly available) datasets.

An important question for the EMS operator is to select the right number of ambulances. Using too few ambulances leads to high response times and risks the patient's life. On the other hand, budget constraints only allow operating a limited number of ambulances.

TABLE I
AVERAGE RESPONSE TIME IN SECONDS FOR DIFFERENT POLICIES AND 15 AMBULANCES.

Policy	Validation	Testing
Random	380.41	409.55
Least Ambulances	307.12	336.91
Nearest Base Station	228.05	236.84
MEXCLP	281.62	301.12
DMEXCLP	296.69	305.68
ERTM	218.16	234.15
DRLSN	221.42	246.07

TABLE II
RESULTS 15 AMBULANCES PICKUP WITHIN 10 MINUTES RATIO FOR DIFFERENT POLICIES AND 15 AMBULANCES.

Policy	Validation	Testing
Random	0.86	0.85
Least Ambulances	0.91	0.89
Nearest Base Station	0.96	0.96
MEXCLP	0.94	0.93
DMEXCLP	0.95	0.94
ERTM	0.95	0.94
DRLSN	0.94	0.93

In Figure 8 and Figure 9, we compare the influence of the number of ambulances on the average response time and 10 minute pickup ratio on the test set. For this what-if scenario, we pick different numbers of ambulances between 10 to 32. With regard to the 10 minute pickup ratio, we can observe that 10 ambulances are not enough to ensure efficient operation. The best approach is only able to meet that target for less than 50% of the incidents. Adding four more ambulances has a huge impact on the 10 minute pickup ratio, with an increase to 0.92. However, increasing the number of ambulances even further has only marginal benefits, 18 ambulances are enough to reach a pickup ratio of 0.99. In Figure 8, we can observe a similar trend for the average response time. The Nearest base station approach works surprisingly well in scenarios where the EMS system is overwhelmed and incidents start queuing up because it minimizes the time until an ambulance can handle the next incident.

VII. CONCLUSION

In this paper, we introduced a high-performance, easily extendable, open-source simulator for the DAR problem. Additionally, we presented an openly available real-world dataset that, to the best of our knowledge, has not been used for the DAR task. We also compared the performance of several SOTA approaches and baselines on this dataset, which led to interesting findings. Surprisingly, the performance of the simple nearest base station heuristic worked surprisingly well and was able to beat more sophisticated approaches with respect to the pickup time within 10 minutes. Additionally, the average response time of this baseline is only marginally worse than the ERTM model, the best performing model with respect to this metric.

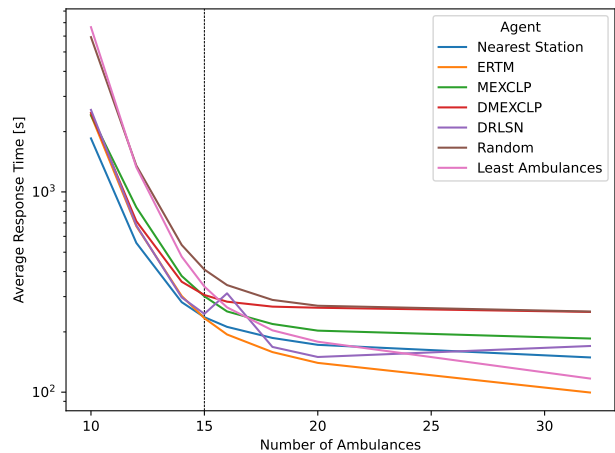


Fig. 8. Comparison of the average response time in seconds on the test set on a logarithmic scale for number of ambulances between 10 and 32.

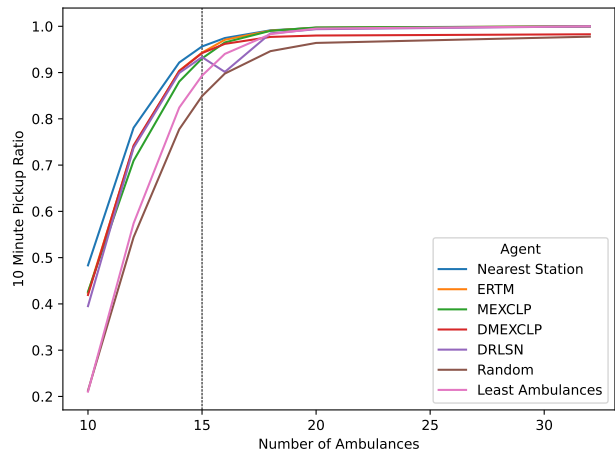


Fig. 9. Comparison of the 10 minute pickup ratio on the test set for number of ambulances between 10 and 32.

Overall, our paper represents a significant step towards comparable research in the area of DAR. Furthermore, the availability of an open-source simulator and dataset considerably reduces the barrier to do research in the field and thus encourages the development of new approaches. In contrast to most existing simulators, our simulator is well suited for training RL agents, which have been able to achieve SOTA performance in many applications. Moreover, RL is well suited to integrate data from smart cities and thus has the potential to improve the operation of EMS systems. Therefore, we are convinced that our simulator enables a promising field of research.

In recent years, an increasing amount of real-time sensor data has become available in smart cities and allows precise insights into complex urban systems. In a future line of work, we want to integrate such data sources and utilize their potential to improve EMS operations. For example, data from

public transportation systems could be considered to detect gatherings of people and improve the estimate of how many people are currently in certain parts of the city, and thus the probability of a medical emergency [27]. Furthermore, integrating dynamic travel times, i.e., real-world traffic information, has the potential to make the simulation more accurate as well as developing policies that adapt to the dynamic traffic conditions [28], [29].

Our experiments have shown that existing RL approaches cannot surpass simple heuristics. Since RL agents are theoretically able to learn complex strategies and have shown impressive performance on similar tasks [30]–[32], it is promising to further investigate developing improved RL methods.

In a future line of work, we want to allow utilizing severity information about emergencies and permit ambulances to directly serve new incidents without returning to a base station beforehand. Another major resource direction is the integration of valuable urban IoT data, such as real-time traffic information, location and health data from mobile devices, or micro-weather data.

ACKNOWLEDGEMENTS

This work was funded by the Bavarian Ministry for Economic Affairs, Regional Development and Energy as part of a project to support the thematic development of the Institute for Cognitive Systems. We would like to thank Hyeri An, Kerui Zhang, and Jingcheng Wu for their helpful contributions regarding the implementation of the simulator and baselines.

REFERENCES

- [1] D. Degel, L. Wiesche, and B. Werners, “Data driven ambulance optimization considering dynamic and economic aspects,” in *Operations Research Proceedings 2013*, D. Huisman, I. Louwerse, and A. P. Wagelmans, Eds. Cham: Springer International Publishing, 2014, pp. 105–111.
- [2] D. Neira-Rodado, J. W. Escobar-Velasquez, and S. McClean, “Ambulances deployment problems: Categorization, evolution and dynamic problems review,” *ISPRS International Journal of Geo-Information*, vol. 11, no. 2, p. 109, 2022.
- [3] C. J. Jagtenberg, S. Bhulai, and R. D. van der Mei, “An efficient heuristic for real-time ambulance redeployment,” *Operations Research for Health Care*, vol. 4, pp. 27–35, 2015.
- [4] M. Gendreau, G. Laporte, and F. Semet, “A dynamic model and parallel tabu search heuristic for real-time ambulance relocation,” *Parallel computing*, vol. 27, no. 12, pp. 1641–1653, 2001.
- [5] S. Ji, Y. Zheng, Z. Wang, and T. Li, “A deep reinforcement learning-enabled dynamic redeployment system for mobile ambulances,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 1, pp. 1–20, 2019.
- [6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.
- [7] P. L. V. D. Berg and J. T. V. Essen, “Comparison of static ambulance location models,” *International Journal of Logistics Systems and Management*, vol. 32, no. 3-4, pp. 292–321, 2019.
- [8] R. Church and C. ReVelle, “The maximal covering location problem,” in *Papers of the regional science association*, vol. 32, no. 1. Springer-Verlag, 1974, pp. 101–118.
- [9] M. S. Daskin, “A maximum expected covering location model: formulation, properties and heuristic solution,” *Transportation science*, vol. 17, no. 1, pp. 48–70, 1983.
- [10] C. ReVelle and K. Hogan, “The maximum availability location problem,” *Transportation science*, vol. 23, no. 3, pp. 192–200, 1989.
- [11] C. Toregas, R. Swain, C. ReVelle, and L. Bergman, “The location of emergency service facilities,” *Operations research*, vol. 19, no. 6, pp. 1363–1373, 1971.
- [12] R. Alanis, A. Ingolfsson, and B. Kolfal, “A markov chain model for an ems system with repositioning,” *Production and operations management*, vol. 22, no. 1, pp. 216–231, 2013.
- [13] O. Berman, “Dynamic repositioning of indistinguishable service units on transportation networks,” *Transportation Science*, vol. 15, no. 2, pp. 115–136, 1981.
- [14] T. Andersson and P. Värbrand, “Decision support tools for ambulance dispatch and relocation,” in *Operational Research for Emergency Planning in Healthcare: Volume 1*. Springer, 2016, pp. 36–51.
- [15] M. S. Maxwell, M. Restrepo, S. G. Henderson, and H. Topaloglu, “Approximate dynamic programming for ambulance redeployment,” *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 266–281, 2010.
- [16] M. S. Maxwell, S. G. Henderson, and H. Topaloglu, “Tuning approximate dynamic programming policies for ambulance redeployment via direct search,” *Stochastic Systems*, vol. 3, no. 2, pp. 322–361, 2013.
- [17] V. Schmid, “Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming,” *European Journal of Operational Research*, vol. 219, no. 3, pp. 611–621, 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.ejor.2011.10.043>
- [18] Y. Yue, L. Marla, and R. Krishnan, “An efficient simulation-based approach to ambulance fleet allocation and dynamic redeployment,” *Proceedings of the National Conference on Artificial Intelligence*, vol. 1, pp. 398–405, 2012.
- [19] L. Zhen, K. Wang, H. Hu, and D. Chang, “A simulation optimization framework for ambulance deployment and relocation problems,” *Computers & Industrial Engineering*, vol. 72, pp. 12–23, 2014.
- [20] D. Bertsimas and Y. Ng, “Robust and stochastic formulations for ambulance deployment and dispatch,” *European Journal of Operational Research*, vol. 279, no. 2, pp. 557–571, 2019.
- [21] S. G. Henderson, A. J. Mason *et al.*, “Bartsim: a tool for analysing and improving ambulance performance in auckland, new zealand,” in *Proceedings of the 35th annual conference of the operational research society of New Zealand, Wellington, New Zealand*. Citeseer, 2000, pp. 57–64.
- [22] M. Allen, K. Pearn, and T. Monks, “Developing an OpenAI Gym-compatible framework and simulation environment for testing Deep Reinforcement Learning agents solving the Ambulance Location Problem,” *arXiv preprint arXiv:2101.04434*, 2021.
- [23] J. Ong, D. Kulpanowski, Y. Xie, E. Nikolova, and N. M. Tran, “Open-EMS: an open-source Package for Two-Stage Stochastic and Robust Optimization for Ambulance Location and Routing with Applications to Austin-Travis County EMS Data,” *arXiv preprint arXiv:2201.11208*, 2022.
- [24] S. Ridler, A. J. Mason, and A. Raith, “A simulation and optimisation package for emergency medical services,” *European Journal of Operational Research*, vol. 298, no. 3, pp. 1101–1113, 2022.
- [25] W. Jakob, J. Rhineland, and D. Moldovan, “pybind11 – seamless operability between c++11 and python,” 2017, <https://github.com/pybind/pybind11>.
- [26] H. Kriegel, E. Schubert, and A. Zimek, “The (black) art of runtime evaluation: Are we comparing algorithms or implementations?” *Knowl. Inf. Syst.*, vol. 52, no. 2, pp. 341–378, 2017. [Online]. Available: <https://doi.org/10.1007/s10115-016-1004-2>
- [27] H. Zou, Y. Zhou, J. Yang, and C. J. Spanos, “Device-free occupancy detection and crowd counting in smart buildings with wifi-enabled iot,” *Energy and Buildings*, vol. 174, pp. 309–322, 2018.
- [28] H. Taghipour, A. B. Parsa, and A. Mohammadian, “A dynamic approach to predict travel time in real time using data driven techniques and comprehensive data sources,” *Transportation Engineering*, vol. 2, p. 100025, 12 2020.
- [29] M. Sarrab, S. Pulparambil, and M. Awadalla, “Development of an iot based real-time traffic monitoring system for city governance,” *Global Transitions*, vol. 2, pp. 230–245, 2020.
- [30] K. Lin, R. Zhao, Z. Xu, and J. Zhou, “Efficient large-scale fleet management via multi-agent deep reinforcement learning,” in *Proc. of the 24th ACM SIGKDD*, 2018, pp. 1774–1783.
- [31] S. Schmolle and M. Schubert, “Semi-markov reinforcement learning for stochastic resource collection,” in *IJCAI*, 2021, pp. 3349–3355.
- [32] W. Kool, H. van Hoof, and M. Welling, “Attention, learn to solve routing problems!” in *International Conference on Learning Representations*, 2018.

D DEAR: Dynamic Electric Ambulance Redeployment

Venue 18th International Symposium on Spatial and Temporal Data

DOI <https://doi.org/10.1145/3609956.3609959>

Declaration of authorships The research idea was developed and conceptualized by Lukas Rottkamp and Niklas Strauss and discussed with Matthias Schubert and Niklas Strauss. Lukas Rottkamp implemented the approach and extended the simulation written by Niklas Strauss to electric ambulances. Niklas Strauss and Lukas Rottkamp wrote the manuscript which was improved by all co-authors.

Publication

DEAR: Dynamic Electric Ambulance Redeployment

Lukas Rottkamp*
MCML, LMU Munich
Munich, Germany
rothkamp@cip.ifi.lmu.de

Niklas Strauß*
MCML, LMU Munich
Munich, Germany
strauss@dbs.ifi.lmu.de

Matthias Schubert
MCML, LMU Munich
Munich, Germany
schubert@dbs.ifi.lmu.de

ABSTRACT

Dynamic Ambulance Redeployment (DAR) is the task of dynamically assigning ambulances after incidents to base stations to minimize future response times. Though DAR has attracted considerable attention from the research community, existing solutions do not consider using electric ambulances despite the global shift towards electric mobility. In this paper, we are the first to examine the impact of electric ambulances and their required downtime for recharging to DAR and demonstrate that using policies for conventional vehicles can lead to a significant increase in either the number of required ambulances or in the response time to emergencies. Therefore, we propose a new redeployment policy that considers the remaining energy levels, the recharging stations' locations, and the required recharging time. Our new method is based on minimizing energy deficits (MED) and can provide well-performing redeployment decisions in the novel Dynamic Electric Ambulance Redeployment problem (DEAR). We evaluate MED on a simulation using real-world emergency data from the city of San Francisco and show that MED can provide the required service level without additional ambulances in most cases. For DEAR, MED outperforms various established state-of-the-art solutions for conventional DAR and straightforward solutions to this setting.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**; • **Computing methodologies** → **Simulation environments**.

KEYWORDS

Ambulance Redeployment, Optimization, Spatio-Temporal Data

ACM Reference Format:

Lukas Rottkamp, Niklas Strauß, and Matthias Schubert. 2023. DEAR: Dynamic Electric Ambulance Redeployment. In *Symposium on Spatial and Temporal Data (SSTD '23)*, August 23–25, 2023, Calgary, AB, Canada. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3609956.3609959>

1 INTRODUCTION

The Emergency Medical Service (EMS) is a critical part of health infrastructure all over the world [15]. Paramedics are often the first professional aid in health emergencies and are responsible for safe

*Both authors contributed equally to this research.



This work is licensed under a Creative Commons Attribution International 4.0 License.

SSTD '23, August 23–25, 2023, Calgary, AB, Canada
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0899-2/23/08.
<https://doi.org/10.1145/3609956.3609959>

and quick transport to a secondary care unit such as a hospital. A low response time to emergency calls has increased survival and recovery rates in life-threatening health conditions such as cardiac arrest [5, 18]. Ambulance response times to emergencies depends on various factors, such as the emergency call itself, the processing time needed for dispatch, the readiness of a qualified paramedic team, and its travel time to the incident location. Travel time is a substantial factor. While it can be accelerated by using high-powered vehicles and specialized training for driving in emergency conditions, the initial distance of the ambulance to the incident site is the most prominent factor, with various approaches trying to minimize this distance by proper ambulance placement.

Today, most ambulances are outfitted with internal combustion engines (ICE) using fossil fuels. However, the growing public demand for less air pollution and less release of greenhouse gases promotes the transition towards electric vehicles (EV). Electric ambulances further come with additional benefits, such as a smoother acceleration improving in-ambulance care. Thus, a first generation of electric ambulances is already commercially available.

Ambulances are usually positioned at base stations strategically placed over a city or coverage area to minimize incident response times. Incoming emergency calls are assigned to an ambulance, which drives to the incident location. Some incidents can be resolved on-site, while in other cases, patients need to be transported to a hospital. After completing their assignment, ambulances return to a base station. While ambulances could return to their origin station, it is often advisable to select another base station based on the actual ambulance distribution at this time. This selection of base stations is known as the Dynamic Ambulance Redeployment (DAR) problem in literature [13, 16, 23].

In this paper, we show that existing approaches do not perform well when confronted with electric ambulances. First, we present a formal definition of the Dynamic Electric Ambulance Redeployment Problem (DEAR), which extends existing DAR formalizations by battery levels, range restrictions, charging stations, and recharging. Based on this extension, we can examine the performance of established state-of-the-art methods for dynamic ambulance redeployment, which do not consider these aspects. Afterwards, we present the minimizing energy deficits (MED) approach, designed to avoid these shortcomings and provide state-of-the-art ambulance redeployment for E-Ambulances. Our method is based on matching the predicted future demand in the area of each base station to the joint energy level of the ambulances. The energy level of vehicles at a base station is extrapolated for the same time frame as the future demand and considers any recharging activity increasing the energy level. Based on both estimations on future development, MED assigns ambulances to those base stations where the deficits between the energy level and the demand are expected to be the largest. We compare MED to various state-of-the-art conventional

ambulance redeployment methods on an extended environment of [23]. Our results demonstrate that the conventional DAR methods suffer significant performance decreases in various settings. In contrast, MED can cope well with the requirements of E-Ambulances, often compensating for their drawbacks against using conventional ICE ambulances.

To summarize, our contributions are as follows:

- We formalize DEAR, an extension of the DAR problem considering electric ambulances.
- We extended a DAR simulation environment based on real-world data to consider the DEAR setting and examine the performance of conventional DAR methods.
- We propose MED and present experimental results showing that it copes well with DEAR compared to existing DAR methods and basic DEAR approaches.

The remainder of this paper is structured as follows: Related work is presented in Section 2. We then formulate the Dynamic Electric Ambulance Redeployment Problem (DEAR) in Section 3 and propose MED in Section 4. We evaluate established DAR approaches and MED for DEAR using a simulation based on real-world incident data from San Francisco in Section 5 and summarize our work in Section 6.

2 RELATED WORK

The ambulance location problem (ALP) is an established research topic. Existing approaches can be classified into static and dynamic methods: In static methods, ambulances are stationed at fixed base stations and always return to the same base station after an incident has been handled [7, 8, 19]. One way to obtain a static assignment is to solve the *Maximum Expected Covering Location Problem* (MEXCLP) [8, 13]. Its solution maximizes the expected coverage of incident locations. In contrast to the *Maximum Coverage Location Problem* [7] it is based on, the underlying model assumes an ambulance to be *busy* with a certain probability. In this way, ambulances that are unavailable due to being on a mission, are not included in the coverage calculation. This reasonable modification has been proven to be advantageous compared to earlier methods [12, 13]. *Expected Response Time Model* (ERTM) [3] is another static approach that has shown excellent performance due to its direct minimization of the expected response time [3, 23].

Current state-of-the-art ALP solutions use a dynamic assignment due to the volatility of the problem [13]. The dynamic assignment of ambulances is also called *Real-Time Ambulance Redeployment Problem* or *Dynamic Ambulance Redeployment Problem* (DAR). Dynamic redeployment leads to better response times than static return policies because the stochastic nature of incoming emergency calls can lead to imbalances in ambulance distribution which are ignored by static approaches [10, 11]. The redeployment decision is primarily based on the locations of ambulances and base stations but may also take other factors, such as demand distributions, into account. The *DMEXCLP* approach by [13] is a dynamic variation of MEXCLP. At each redeployment step, it selects the base station providing the largest coverage increase in the respective situation according to the MEXCLP strategy. This way, DMEXCLP takes the actual distribution of ambulances into account. A reinforcement-learning based

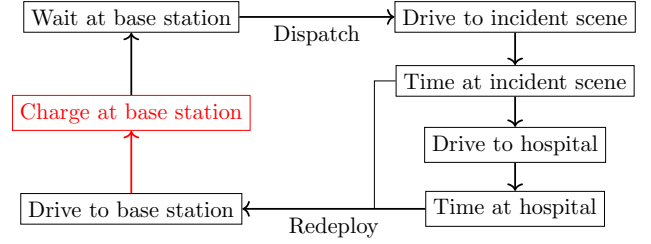


Figure 1: Simplified schematic overview of the modeled EMS process. Specifics for electric ambulances are shown in red.

approach “Reinforcement Learning Deep Score Network” (DRLSN) is presented by [14].

A vision paper by [20] highlights the growing importance of electric ambulances and the associated challenge of keeping a fleet of ambulances charged. It suggests a high-level framework for ambulance scheduling concerning the optimal use of renewable energy sources, including predictive components for patient demand and energy production and use. Though this work is related, it does neither propose a formalization of DEAR nor does it provide a method for the redeployment problem for electric ambulances.

3 PROBLEM DEFINITION

In this section, we will provide a formal definition of the Dynamic Electric Ambulance Redeployment (DEAR) problem, outlining the operational process of the Emergency Medical Services (EMS) provider and considering the specifics of electric ambulances. Figure 1 provides a visual representation of the EMS process. When an incident occurs, the EMS operator receives a call, and an available ambulance is dispatched from a base station to the incident location. In our scenario, the ambulance closest in driving time is dispatched to ensure a prompt response. If no ambulance is available, the incident is handled as soon as an ambulance becomes available again. Upon arrival at the incident, on-site care is provided to the patient. Depending on the patient’s condition, subsequent transport to a hospital may be necessary. Otherwise, the ambulance is redeployed from the incident site to a base station. Once the ambulance arrives at a base station, it becomes idle and available for dispatch. Considering electric vehicles introduces unique challenges compared to Internal Combustion Engine (ICE) vehicles. The downtime for refueling ICE vehicles is typically not a significant concern due to their long ranges and fast refueling times. However, electric vehicles have shorter ranges and require substantial charging time. Therefore, factors such as charging downtime, battery levels, and the availability of fast chargers at base stations need to be considered in the EMS process. It is crucial only to dispatch an electric ambulance if its battery is sufficiently charged to not run out of energy while handling the incident. Therefore, we define a minimum dispatch range τ_{MDR} (measured in time units) as the worst-case trip, starting from the current base station to any incident location, followed by transportation to any hospital, and finally redeployment to a base station.

Electric ambulances can be charged at regular AC outlets (we refer to them as slow chargers), which are already available in large

numbers at base stations. However, slow chargers have limited power output, resulting in extended charging times and longer downtimes of ambulances. Charging times can be significantly reduced by installing high-voltage DC chargers (fast chargers specifically installed for electric vehicles) at base stations. However, their number is limited because installation presents a significant cost factor and constraints caused by the capabilities of the energy grid.

Assigning chargers to ambulances at a base station requires a charging policy when the number of ambulances exceeds the number of fast chargers. The objective is to charge ambulances in a manner that allows them to reach the minimum dispatch range τ_{MDR} as quickly as possible, thereby maximizing the number of available ambulances. It is also important to avoid an unreasonably high number of re-plugging actions by staff. To achieve these goals, we implement the following approach: Ambulances below τ_{MDR} are categorized as high-priority and are charged first. If there are more high-priority ambulances than available chargers or fast chargers, the ambulance with the shortest time required to reach τ_{MDR} is prioritized for charging. This ensures that ambulances are prepared for service at the earliest possible time. Once an ambulance reaches τ_{MDR} , it becomes a low-priority ambulance. For charging low-priority ambulances, we prioritize ambulances with the lowest battery level to minimize the number of re-plugging actions. Re-plugging can occur when an ambulance at the station is sufficiently charged to provide the minimum dispatch range, is fully charged, arrives, or is dispatched.

Now, we present a formal definition of the novel DEAR problem, considering the aforementioned characteristics. In this task, an operator needs to dynamically select a base station to redeploy an ambulance to after the ambulance finishes handling an incident, either from the incident site or the hospital.

The road network is represented as a graph $G = (V, E)$, where V is the set of nodes representing locations in the road network, and E is the set of directed edges representing road segments connecting the nodes.

Incidents are emergencies requiring medical attention by an ambulance and are denoted as I . Each incident is mapped to the nearest node in the graph.

Base Stations Let W be the set of base stations available within the road network, where ambulances are stationed and dispatched to incidents. Each base station is mapped to the closest node in the road network. Base stations are equipped with charging infrastructure to support the operation of electric ambulances. They possess an unlimited number of slow chargers (regular AC outlets) and have varying numbers of fast chargers. Not all base stations are guaranteed to have fast chargers available.

Hospitals The set H represents the hospitals. Similar to base stations, hospitals are mapped to the closest node in the graph.

Ambulances are electric vehicles, introducing specific characteristics that affect their operational constraints. Key properties include battery level and capacity, energy use per time, and charging characteristics. The charging rate of an ambulance depends on various factors, including its current battery level and the power output of the charger. A linear charging function is utilized, although other charging functions may also be employed. We assume that all ambulances are the same type, i.e., their key properties are equal. Let us note that our method can easily be adapted to

more specific settings if required. Ambulances are initially assigned to base stations and can be dynamically redeployed to other base stations depending on incident demand. We allow an ambulance to be redeployed only after finishing handling an incident.

Travel Times In our setting, the travel times $\tau(i, j)$ between two nodes $i, j \in V$ are assumed to be deterministic and do not vary with traffic conditions. When responding to an incident or transporting a patient, ambulances use lights and sirens to alert other drivers, enabling them to travel at fast speeds [4]. We denote the travel time with lights and sirens activated as $\hat{\tau}(i, j)$.

4 MED: MINIMIZE ENERGY DEFICIT

In this section, we introduce our approach Minimize Energy Deficit (MED) for the DEAR problem.

While approaches for solving the DAR problem can be applied, they do not take the additional complexity of electric ambulances into account. Our evaluation demonstrates that this leads to drastically degraded response times or requires multiple additional ambulances to maintain EMS service levels compared to combustion engines.

Thus, it is crucial for a redeployment policy to take battery levels and charging into account. MED is based on the concept of matching the anticipated energy demand at different stations with the expected energy supply at those stations. While the energy demand depends on the incidents and, consequently, the amount of energy needed to handle all incidents. On the other hand, the expected energy supply depends mainly on the distribution of ambulances across the base stations, which is influenced by redeployment decisions. Whenever a redeployment decision needs to be made, our approach deploys the ambulance to the station, which minimizes the energy deficit.

Our proposed method consists of three steps described in the remainder of this section:

- (1) Determine the expected energy demand.
- (2) Determine the expected energy supply.
- (3) Calculate and minimize the energy deficit.

4.1 Expected Energy Demand

We introduce the concept of energy demand θ_w , which refers to the expected energy required to handle incoming incidents within the lookahead duration Δt at base station w . It is determined based on the expected number of incidents in the vicinity of the base station $d_w(t_{\text{now}}, \Delta t)$ during the lookahead duration and an expected energy use per incident ρ_w . The expected energy demand can be expressed as the product of these:

$$\theta_w = d_w(t_{\text{now}}, \Delta t) \rho_w \quad (1)$$

We define the demand forecast $d_w(t_{\text{now}}, \Delta t)$ as a function that estimates the expected number of incidents in the vicinity of the station w from the current time t_{now} until $t_{\text{now}} + \Delta t$. Numerous approaches have been proposed in the literature for predicting ambulance demand [21, 22, 25, 26]. These methods include but are not limited to machine learning techniques, time series analysis, and statistical models. In this paper, we compute an hourly historical average for demand prediction. [6] shows that this method yields a strong baseline for predicting ambulance demand. Let us note that

our approach does not depend on a specific forecasting method and likely benefits from more accurate predictions. We leave the exploration of more sophisticated demand models to future research.

The vicinity V_w of a base station w is defined to be the incident locations $i \in V$ where the travel time $\tau(w, i)$ is shorter than from any other station. Mathematically, this can be expressed as follows:

$$V_w = \{i \in V | \tau(w, i) \leq \min_{w' \in W} \tau(w', i)\} \quad (2)$$

Using historical incident data, we calculate the average number of incidents per hour $\kappa_w(h)$ in the vicinity of each base station w and each hour of day $h \in \{0, \dots, 23\}$. Let $\beta_h \in [0, 1]$ represent the fraction of hour h in the time interval $[t_{\text{now}}, t_{\text{now}} + \Delta t]$. The demand forecast is then given by:

$$d_w(t_{\text{now}}, \Delta t) = \sum_{h \in \{0, \dots, 23\}} \beta_h \kappa_w(h) \quad (3)$$

Determining the expected energy per incident within the proximity of each station holds significant importance. This necessitates evaluating the energy expenditure for traveling from a base station to the incident location, potentially to a hospital and returning to a station. A simplistic approach would assume that incidents solely occur at the centers of each demand area (i.e., the base stations), then travel to the nearest hospital, and finally return to the closest station. However, such an approach lacks accuracy. Therefore, we assume that the locations of incidents are uniformly spatially distributed across all possible incident locations $i \in V_w$ in the vicinity of w . We consider the probability of requiring transportation to a hospital, as well as accounting for the distribution of patients transported to different hospitals and the expected energy for redeployment to a station. The hospital distribution and the probability of requiring hospital transportation are derived from historical data.

We denote the proportion of incidents requiring hospital transportation as α , while α_h is the fraction of these incidents handled by hospital h . To calculate the expected energy use per incident ρ_w in the vicinity of station w , we first determine the expected driving time for fully handling an incident and redeployment to a station. Subsequently, we estimate the energy usage by multiplying the resulting driving times with the parameter P_{driving} , which approximates the energy consumed per unit of time:

$$\mathbb{E}(\rho_{\text{hospital}}(i)) = \sum_{h \in H} \alpha_h \frac{1}{|W|} \sum_{w' \in W} (\hat{\tau}(i, h) + \tau(h, w')) \quad (4a)$$

$$\mathbb{E}(\rho_{\text{base}}(i)) = \frac{1}{|W|} \sum_{w' \in W} \tau(i, w') \quad (4b)$$

$$\rho_w = P_{\text{driving}} \frac{1}{|V_w|} \sum_{i \in V_w} \hat{\tau}(w, i) + \alpha \mathbb{E}(\rho_{\text{hospital}}(i)) + (1 - \alpha) \mathbb{E}(\rho_{\text{base}}(i)) \quad (4c)$$

4.2 Expected Energy Supply

This section focuses on outlining the methodology for calculating the expected energy supply ϕ_w , at a base station w over a specific time period. The actual energy supply depends on the demand, as ambulances may leave the base station to respond to incidents. While it is theoretically possible to model the distribution of incidents and to sample from an exponentially expanding set of future

scenarios to derive estimates, finding optimal solutions is computationally intractable. Even approximations similar to the hindsight planning approaches in [24] are impracticable due to the inherent complexity and real-time constraints of the DEAR problem. To address this, we propose calculating an optimistic, expected energy supply $\hat{\phi}_w$, assuming that no incidents occur and no ambulances are redeployed during the prediction horizon, effectively disregarding the demand. This simplification allows for a deterministic calculation. However, we account for the probability of ambulances being dispatched and subsequently reducing the energy supply during the lookahead duration Δt . This is achieved by introducing a charging discount factor $\gamma \in [0, 1]$ to adjust the expected energy supply, resulting in $\phi_w = \gamma \hat{\phi}_w$. Note that even with those assumptions, determining the expected energy supply still requires simulating the complex charging logic and considering the arrivals of ambulances en route to the base station.

4.3 Minimize Energy Deficit

After we have defined the expected energy demand and supply, we continue by specifying how to calculate the energy deficits and subsequently dynamically redeploy ambulances. To define the energy deficit δ_w at a specific base station w , we calculate the difference between the expected energy demand and supply: $\delta_w = \theta_w - \phi_w$. However, simply minimizing this deficit has certain limitations. For instance, if a station already has sufficient supply to meet the demand, adding more supply would be unnecessary, even if it reduces the deficit. Therefore, we introduce a weighted deficit ω_w using a soft plus function [9]. This function assigns lower importance to stations with negative deficits (i.e., surplus supply compared to demand) and prioritizes stations with high deficits. The weighted deficit is calculated as follows:

$$\omega_w = \log(1 + \exp(\frac{1}{100} \delta_w)) \quad (5)$$

In the last step, we describe the methodology for using the weighted energy deficit ω_w to make redeployment decisions. Whenever an ambulance a needs to be redeployed, we simulate sending the ambulance to each base station w to obtain $\omega_w(a)$. This is used to calculate the reduction in the expected weighted energy deficit $\omega_w - \omega_w(a)$ at each station. Subsequently, we redeploy the ambulance to the station that yields the most significant reduction.

4.4 Computational Complexity

Making ambulance redeployment decisions is a time-critical task, and any method should be able to compute a redeployment decision within seconds. Consequently, we designed our approach with this requirement in mind. To make each redeployment decision, we must assess the expected energy demand ρ_w and the expected energy supply ϕ_w at each station w both with and without the ambulance being redeployed. The energy demand consists of two components, the expected number of incidents and energy use per incident. The complexity of the former depends on the demand prediction model used. In this paper, we use the historical average, which can be pre-computed so that a prediction can be made in constant time. The second component, the expected energy use per incident, is a constant factor that can also be pre-computed. Therefore, calculating the expected energy demand has constant

complexity. The primary computational effort lies in determining future energy supplies, which involves simulating the charging logic of each ambulance. As we assume optimistically that ambulances will not be deployed, they will eventually reach full charge, and their energy supply will no longer change. In other words, each ambulance adds a particular constant computational effort to simulate. From a computational point of view, the time complexity of determining future energy supplies is linear in the number of ambulances, resulting in a complexity of $O(|A|)$.

Our approach has an overall worst-case time complexity of $O(|A||W|)$. For each station, we need to calculate the expected energy demand (with constant complexity due to pre-computation) and compute the expected energy supply twice.

We implemented our method in C++ to obtain evaluation results presented in the next section. Executed on a notebook with Intel[®] Core[™] i7-10750H CPU, one redeployment decision is obtained in approximately 0.23 milliseconds during a typical evaluation run with 45 base stations and 25 ambulances. Repeating the measurements with 1,000 ambulances in the environment (an unreasonably high number for benchmark purposes only), one decision is obtained in approx. 0.26 ms. These results satisfy the real-time requirement.

5 EVALUATION

In this section, we evaluate various solutions in a DEAR setting based on real-world emergency data from the city of San Francisco. We will first detail our experimental setup and, afterward, examine the impact of electric ambulances on DAR solutions and the performance of our newly proposed method *MED*.

5.1 Simulation environment

We evaluate various scenarios using an event-based simulator that replays real-world emergency data. This simulator mirrors the DEAR problem defined in Section 3 to simulate the operations of the EMS with electric ambulances. The foundation of our simulator is an openly accessible simulation environment for dynamic ambulance redeployment developed by [23]. Since this simulation does not consider electric vehicles, we extended it to include vehicles' battery state, charging, and energy use. Further, base stations were modified to contain a definable number of chargers of specified charging power with the problem definition's charging logic. Note that charging electric vehicles is a complex process influenced by factors such as battery level, battery condition, and ambient temperature. Similarly, energy usage depends on variables like driving profile, traffic conditions, and secondary loads such as heating or equipment required for patient care. Given the complexity of modeling these factors accurately, we simplify our simulation by utilizing constant values for charging power and driving energy usage, respectively.

The simulated EMS system is based on the city of San Francisco, USA. The system contains eleven hospitals and 45 base stations. Their locations are depicted in Figure 2. The road network graph used in the simulation was acquired from OpenStreetMap¹, with intersections representing the graph nodes. Hospitals and base stations were attached to the nearest node in the graph. Driving times

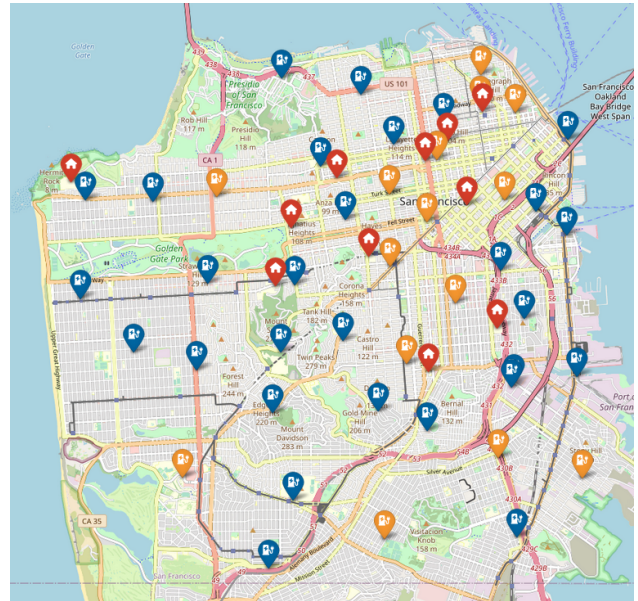


Figure 2: Simulation environment of San Francisco, USA. Locations of base stations are marked in orange if a fast charger is present and otherwise, in blue; Hospitals are marked in red. Note that population density is highest in the downtown area (top right). Map data © OpenStreetMap contributors.

were computed based on the shortest path with respective street limits depending on the road type. To account for traffic and slowing down due to turns and crossings, we calibrated the driving times based on estimates by HERE Traffic² by multiplying a constant factor. Based on this method, the average speed, including traffic congestion, was estimated to be $32 \frac{km}{h}$. Ambulances returning to a base station are assumed to drive at traffic speed. However, when moving toward an incident or hospital, ambulances are granted certain exemptions from traffic regulations, allowing them to drive faster. Nevertheless, traffic congestion and safety considerations still limit realistic driving speed. Thus, we scaled driving times accordingly, resulting in an average emergency speed of $50 \frac{km}{h}$ as suggested by [11].

The city of San Francisco has made real incident data publicly available through their *Fire Department Calls for Service* dataset³. This dataset contains historical records of health emergency calls, including information such as the date, time, and location of each emergency. This enables us to simulate the historical occurrence of incidents with arbitrary configurations of base stations, ambulances, and redeployment methods. However, it is important to note that while the dataset indicates whether a hospital was targeted, specific details such as the hospital's name or location are not disclosed. Selecting a suitable hospital involves a complex decision process, including various factors such as the patient's medical needs, hospital occupancy levels, patient preferences, and the proximity to hospitals [1]. Since this information is unavailable to us,

¹ODbL license <https://www.openstreetmap.org/copyright>. Map data copyrighted by OpenStreetMap contributors and available from <https://www.openstreetmap.org>.

²<https://www.here.com/platform/traffic-solutions/real-time-traffic-information>

³<https://data.sfgov.org/Public-Safety/Fire-Department-Calls-for-Service/nuek-vuh3>

we determine target hospitals by random sampling according to the real-world distribution of patient transports to hospitals between February 2022 and February 2023 published by the Data Working Group (DWG) of the City of San Francisco⁴. Note that this random sampling was done as a preprocessing step, ensuring the hospital transportations are consistent across all experiments. Locations of incidents were mapped to the nearest graph node in our simulation.

5.1.1 Placement and power of fast chargers. As discussed in our introduction, cities will likely outfit only a subset of base stations with fast chargers, primarily due to installation costs. Therefore, we strategically locate fast chargers at the stations with the highest demand, assigning one fast charger per station. Note that according to our problem definition, additional slower chargers are already present at every station. In our evaluation, we considered three different types of fast chargers, each offering different charging powers. The first type is a high-power DC charger delivering 50 kW of charging power. The second type is a cheaper three-phase AC charger delivering 22 kW charging power. Lastly, we considered a more expensive option of 100 kW charging.

5.1.2 Electric Ambulance Models. The battery capacity and average driving energy use (P_{driving}) in our simulation are based on real-world electric ambulances. We base our experiment values on electric ambulance “WAS 500” because it is a suitable replacement for ICE ambulances and technical data is readily available⁵. We set the battery capacity to 87 kWh, based on the specifications provided in the datasheet of the ambulance. To determine P_{driving} , we consider the average speed and the energy usage from the datasheet. We calculate this value as 30 kW.

5.2 Metrics

As motivated in our introduction, minimizing ambulance response times is critical for EMS providers. In an ambulance redeployment context, response times are usually defined as the time between dispatching an ambulance at its base station and its arrival at the incident scene. Aggregated metrics used for evaluating the performance of EMS systems are the average response time (ART) and the fraction of response times within a certain response time threshold (RTT) [17, 23]. RTT values and targeted fractions are set differently by different institutions [17]. San Francisco’s Emergency Medical Services Agency aims to arrive at life-threatening incidents within a 10 minute threshold at least 90% of the time [2, 23]. We use this metric extensively in our evaluation, denoting it as **RTT10**. We occasionally also include RTT fractions for 8 minutes (**RTT8**) and 12 minutes (**RTT12**).

5.3 Baselines

We compare our method **MED** (Minimize Energy Deficit) with several straightforward baselines as well as several state-of-the-art approaches for redeploying combustion engine ambulances. The most simple baseline is **RAND**, which redeploys the ambulance to a random base station. **NEAR** selects the base station which can be reached fastest by the ambulance (i.e. minimizes driving time). **NEARC** and **NEARF** similarly select the nearest station but

Table 1: RTT10 performance of conventional methods in the ICE case compared to the EV case with different charging powers and 24 ambulances.

Scenario	ERTM	DRLSN	MEXCLP	DMEXCLP
ICE	0.88	0.90	0.83	0.89
EV 22 kW	0.47	0.40	0.30	0.20
EV 50 kW	0.72	0.85	0.58	0.57
EV 100 kW	0.76	0.86	0.67	0.56

consider only stations with chargers (**NEARC**) or free, fast chargers (**NEARF**), respectively. Note that this method checks availability at query time. We also include state-of-the-art approaches from the DAR problem discussed in (Section 2) and refer to them as conventional approaches. These approaches consists of static methods, namely **ERTM**[3] and **MEXCLP**[8, 13], a dynamic method called **DMEXCLP**[13], and the reinforcement learning based approach **DRLSN**[14]. Let us note that **DRLSN** is trained in an environment considering **DEAR**, and thus, it can learn the specific behavior of E-Ambulances. However, we did not change the agent itself as a straightforward extension of observation data did not yield improved results.

5.4 Results

In this section, we present the results of our experimental evaluation based on the previously described simulation environment to answer the following research questions:

- (1) How large is the effect of replacing ICE ambulances with EVs using established DAR methods?
- (2) Does our approach **MED** perform better than methods from related work for **DEAR**?
- (3) What is the influence of simulation parameters such as the number of available chargers?
- (4) How sensitive is our approach to variation of its parameters?

For all experiments, methods were evaluated by simulating one year of incidents (*test set*) in our simulation. The resulting response times were then aggregated to obtain RTT10 and ART metrics. The respective previous year (*validation set*) was used to determine the method’s parameters, such as historical demand and selecting hyper-parameters. The best set of hyper-parameters (according to the RTT10 metric) was selected for evaluation on the test set. Experiments were conducted for the years 2015 to 2022. Due to the numerous parameters involved, including different combinations of years, ambulance quantities, charger quantities, charging power, etc., we cannot present all results here. Unless indicated otherwise, the experiments were conducted with incidents from the year 2022, using 15 fast chargers, each providing 50 kW charging power. Additionally, we included variations of these parameters to facilitate a comprehensive comparison of methods under different scenarios.

5.4.1 Effect of switching to electric ambulances. In this section, we analyze the effectiveness of methods for ordinary DAR settings (conventional approaches) when being applied to the **DEAR** problem. We present the results for ICE and EV scenarios containing 24 ambulances in Table 1, as 24 ambulances are required for the

⁴<http://sfemergencymedicalresponse.weebly.com/ambulance-destinations.html>

⁵<https://www.was-vehicles.com/en/innovation/was-500-electric-ambulance.html>

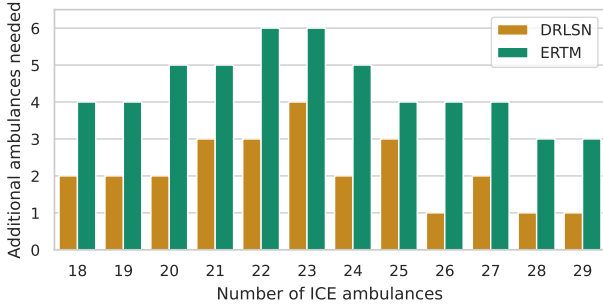


Figure 3: Number of additional ambulances needed to reach same performance (RTT10 metric) as in the non-EV scenario for best methods from related work. 50 kW charging power.

first method to reach the RTT10 target of 90%. We observe a significant decline in the RTT10 metric when introducing energy use and charging, with some cases showing a reduction of more than 50% in performance. Using 22 kW fast chargers results in inferior performance: *ERTM* receives the best 22 kW RTT10 score (0.47), which is not acceptable for an EMS provider, despite its significantly better performance (0.88) in the ICE case. When using 50 kW or 100 kW fast chargers, the decline in performance is less severe but still substantial. *DRLSN* achieves best RTT10 scores in the ICE (0.90), 100 kW (0.86) and 50 kW (0.85) cases. Notably, its reward-based algorithm shows the ability to learn certain characteristics of the EV environment despite not explicitly observing energy-related data. Its poor performance in the 22 kW case may be explained by rewards being too sparse to enable effective training. Like *ERTM*, both *MEXCLP* and *DMEXCLP* show drastic decreases in performance. Although the dynamic method *DMEXCLP* performs better than the static approaches *ERTM* and *MEXCLP* in the ICE case, it experiences substantial difficulties in the EV scenarios, even showing worse results in the 100 kW case compared to 50 kW. Overall, results indicate that using fast chargers with 22 kW charging power will not enable acceptable performance with these methods. Increasing the charging power to 50 kW improved the results, but additional ambulances are still necessary. Installing 100 kW chargers does not appear to improve results substantially. As infrastructure investments generally increase with higher charging power, emergency medical service providers should be aware of this effect when transitioning to electric ambulances.

Figure 3 provides insights into the number of additional ambulances needed when transitioning from ICE to EV ambulances. It depicts the number of additional ambulances required to reach an equal or better RTT10 performance compared to non-EV ambulances for the *ERTM* and *DRLSN*. We use 50 kW chargers in the scenario, as there is a minimal improvement when using 100 kW. *ERTM* requires an additional 3 to 6 ambulances. *DRLSN* requires 2 to 4 additional ambulances when replacing up to 25 ICE ambulances. In settings replacing more than 25 ICE the number of additional ambulances can decrease to 1.

Overall, our results show that employing conventional methods from related work on the DEAR problem requires more ambulances

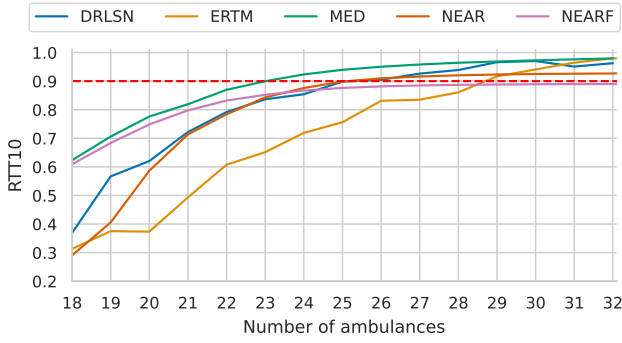
Table 2: Performance of all methods when using 24 ambulances and 50 kW charging power.

Method	RTT8	RTT10	ART
MED	0.87	0.92	4.64
NEAR	0.79	0.88	6.50
NEARF	0.79	0.87	5.38
DRLSN	0.81	0.85	5.90
NEARC	0.75	0.84	5.89
ERTM	0.68	0.72	19.27
MEXCLP	0.50	0.58	32.44
DMEXCLP	0.49	0.57	33.87
RAND	0.01	0.01	153.09

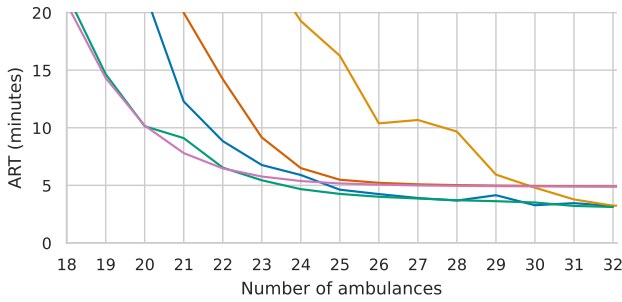
to achieve a similar level of performance compared to ICE ambulances. Additionally, an interesting finding is that the difference between 50 kW and 100 kW charging is minimal in contrast to charging with 22 kW.

5.4.2 Performance of MED. We now introduce results for our approach *MED* and compare them to state-of-the-art conventional methods developed for DAR, as well as our DEAR baselines. Results for all methods are shown in Table 2. We again chose 24 ambulances and 50kW charging power due to the previously mentioned practical relevance of this scenario. Our approach *MED* outperforms all other methods across all metrics. Specifically, it achieves an RTT10 value of 0.92, which is well within the 90% target. The average response time (ART) of 4.64 minutes is about 45s faster than the second-best method *NEARF*, and 75s less than *DRLSN*, the best conventional method from related work. It is worth noting that another nearest station method, *NEAR*, also demonstrates surprisingly good performance, securing the second-best RTT10 value of 0.88. The best performing conventional method is *DRLSN* (0.85), followed by *ERTM* (0.72). The difference between RTT10 and ART scores, especially when considering the comparatively good performance of simplistic baselines such as *NEAR* or *NEARF* underlines the observation that conventional methods do not perform well in the evaluated EV scenario. In contrast, *MED* performs better in DEAR (RTT10 of 0.92) than the best DAR approach in the corresponding ICE scenario (RTT10 of 0.90, compare Figure 1).

The relationship between the number of deployed ambulances and the performance is illustrated in Figure 4 for the best-performing methods. *MED* consistently demonstrates strong results across all metrics. Analyzing the RTT10 performance in Figure 4a, it becomes evident that *MED* outperforms other approaches with a substantial gap to the second best method up to a number of 29 ambulances. As noted before, it is the first to exceed the 90% RTT10 target (dashed red line). Furthermore, its performance considering the ART metric (Figure 4b) is superior to others in the most interesting region (due to its closeness to the 90% RTT10 target) of about 24 ambulances. When 22 or fewer ambulances are used, method *NEARF* yields lower ART values. This is because in these cases, demand for ambulances, and the energy use that comes with it, is so high that all other objectives fade in comparison to obtaining energy as fast as possible. As the *NEARF* method is designed to immediately drive to the nearest free charger, regardless of its location or any



(a) RTT10 metric. Dashed line indicates 90% target.



(b) ART metric.

Figure 4: Performance comparison of best methods for 50 kW charging power.

other criteria, it fulfills this objective well. In situations where a substantial number of ambulances (27 or more) are available, methods from previous research narrow the gap. At this point, the locations and availability of chargers become less critical as it becomes more likely that a charged ambulance is stationed sufficiently close to any incident. Furthermore, slow charging is sufficient to make sure that drained ambulances will be available at a later point in time. It is, however, interesting that the gap for the RTT10 metric (c.f. Figure 4a) closes more slowly than the gap in ART (c.f. Figure 4b). This indicates that *MED* still allows significantly fewer incidents that are not handled within the 10-minute limit than compared methods up to 29 ambulances.

As emergency service providers usually aim to fulfill a certain minimum service level, we provide the number of ambulances needed to reach a 90% fraction of common RTT values in Table 3. An important observation is that *MED* requires the lowest number of ambulances to reach the target in all cases. A RTT8 target is reached by deploying 26 ambulances with *MED*, whereas the second best method, *DRLSN*, requires 29 ambulances. The RTT10 and RTT12 targets are reached with 24 and 22 ambulances, respectively, requiring two and one ambulances less than the runner-up. It is worth mentioning that most methods failed to reach the RTT8 target for fleet sizes up to 40, which is the maximum number of ambulances considered in our experiments.

Table 3: Number of ambulances needed to reach the 90% RTT target for various RTT values. 50 kW charging power.

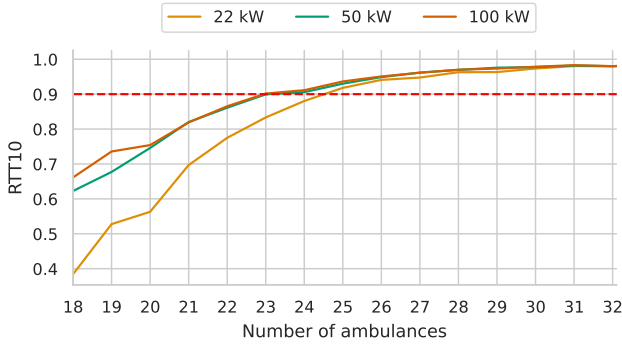
Method	8 min	10 min	12 min
MED	26	24	22
DRLSN	29	26	25
NEAR	> 40	26	24
ERTM	30	29	29
DMEXCLP	> 40	32	31
MEXCLP	> 40	32	29
NEARC	> 40	> 40	23
NEARF	> 40	> 40	23
RAND	> 40	> 40	> 40

Table 4: Performances of *MED* compared to best method from related work for each evaluation year. In each year, *MED* performed best, followed by *DRLSN*. The number of ambulances in each row was determined as the lowest amount that reached 90% RTT10 for the given year. Column *Diff* for RTT10 is the decrease of incidents that could not be reached within 10 minutes. Column *Diff* for ART is the decrease in response times.

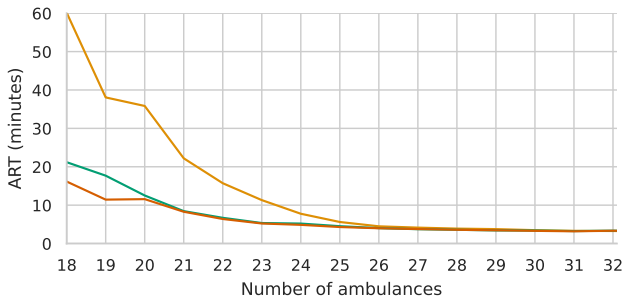
Year	RTT10			ART		
	MED	DRLSN	Diff	MED	DRLSN	Diff
2015	0.901	0.860	-29.18%	4.810	5.551	-13.34%
2016	0.907	0.867	-30.57%	5.101	5.916	-13.78%
2017	0.919	0.877	-34.28%	4.494	5.345	-15.92%
2018	0.912	0.867	-33.63%	4.679	5.491	-14.79%
2019	0.910	0.875	-28.19%	4.579	5.469	-16.27%
2020	0.908	0.872	-28.18%	4.635	5.476	-15.37%
2021	0.905	0.852	-35.42%	4.927	5.767	-14.57%

To see if the superior performance of *MED* can be reproduced in other years, we repeated the experiment above for each pair of years starting in 2015. This includes fitting parameters on the given year and testing methods' performance in the following year. The results summarized in Table 4 show that *MED* can reach the 90% RTT10 target with fewer ambulances than methods from related work each year. The difference in incidents that could not be reached within 10 minutes is considerably lower in these cases, namely between 28.18% to 35.42% lower. Average response times also decrease consistently for all years. In absolute numbers, this means reducing average response times by about 50 seconds in our experiments, which can be valuable in critical emergencies.

These results demonstrate the superior performance of *MED* for the DEAR problem across various scenarios. Furthermore, as *MED* in DEAR displays a similar or better performance than compared methods in the ordinary DAR environments based on ICE ambulances, we can conclude that switching to an equally-sized fleet of E-Ambulances can be done without significantly decreasing response times.



(a) RTT10 metric.



(b) ART metric.

Figure 5: Performance of MED for different charging powers.

5.4.3 *Varying power and number of chargers.* The performance of *MED* for different charging rates and numbers of deployed ambulances is presented in Figure 5. It can be seen again that the difference between 50 kW and 100 kW fast charging power is minimal. However, 22 kW charging results in inferior performance. For example, the RTT10 target is reached with 25 ambulances instead of only 24 for both higher charging rates. This disparity becomes more apparent as the number of ambulances decreases, as the per-ambulance energy use and corresponding charging activity increases in such scenarios. With increasing numbers of ambulances, the charging pressure vanishes, which can be seen in the convergence of all powers’ measurements. Figure 6 depicts the results of different methods for varying the number of installed fast chargers. As before, we use 24 ambulances as the lowest amount to be sufficient to reach the 90% RTT10 target. *ERTM*, *MEXCLP*, and *DMEXCLP* exhibit a slow increase of performance when increasing the number of fast chargers and thus appear to be especially ill-suited for the EV scenario. In contrast, the performance of *MED*, *NEAR*, *NEARF* and *DRLSN* follows an early quick increase with a slower rise once about three fast chargers are installed, i.e., they appear to either use less energy or utilize fast chargers better, or both. It should be noted that *MED* is the only approach that meets the 90% RTT10 level. Furthermore, *MED*’s performance does not substantially increase when more than 11 chargers are installed in the environment. To summarize, our method tailored for EV

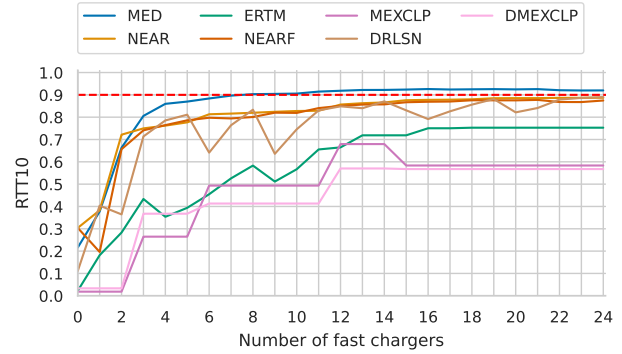


Figure 6: Comparison of RTT10 performance for different numbers of fast chargers. Scenario with 24 ambulances and 50 kW charging power.

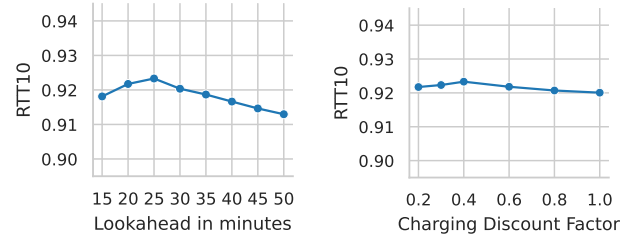


Figure 7: Performance of MED for varying hyper-parameters. Scenario with 24 ambulances and 50 kW charging power.

scenarios requires not only fewer ambulances but also fewer fast chargers.

5.4.4 *Parameter sensitivity.* Figure 7 shows how varying *MED*’s two parameters affect its RTT10 performance, using the scenario of 24 ambulances and 50 kW charging. Examining the parameter lookahead duration Δt (Figure 7 left), the optimal value is 25 minutes, with a roughly linear decrease when higher or lower values are used. The sensitivity of our approach to this parameter is low, as doubling it to 50 minutes only marginally decreases RTT10 performance by about 0.01. Varying the charging discount factor γ (Figure 7 right) appears to have little effect on performance. The optimum is at a value of 0.4, which can be explained by charging processes at base stations being frequently interrupted due to incoming incidents in this challenging scenario.

5.4.5 *Qualitative analysis.* Figure 8 shows a snapshot of our simulation from the point of view of our approach *MED*. The weights assigned by the method (orange bars) are calculated in a way that expected demand (red bars) is offset by available energy (blue bars), i.e., ambulances assigned to the respective base station. The energy distribution appears to be pretty spread out to minimize response times. Several base stations necessarily contain zero energy because, in this scenario, 25 ambulances have to cover all 45 base stations. However, the gaps are mostly in lower demand areas and can be covered by nearby base stations with assigned ambulances.

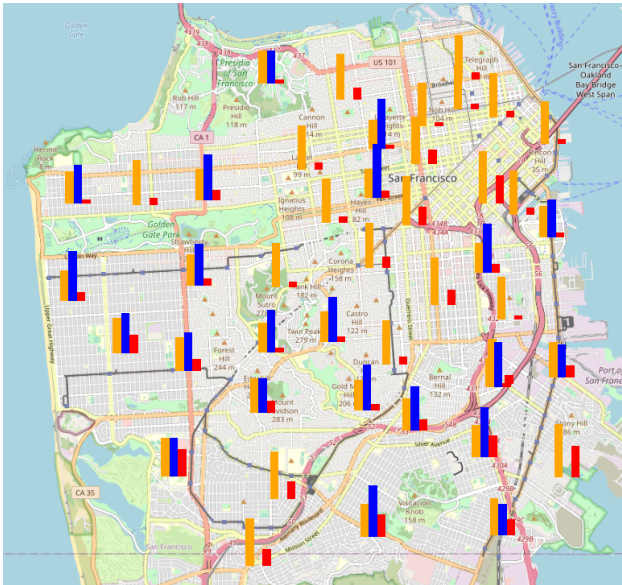


Figure 8: Snapshot of a simulation with 25 ambulances. Bar plots indicate base stations' estimated future energy values as calculated by MED: Supply (blue); Demand (red); Deficits (yellow) after nonlinear scaling (higher scores mean higher priority). Map data © OpenStreetMap contributors.

6 CONCLUSION

In this paper, we introduce the Dynamic Electric Ambulance Redeployment (DEAR) problem extending the Dynamic Ambulance Redeployment (DAR) problem to electric ambulances. We propose the Minimize Energy Deficits (MED) method, which determines redeployment actions by estimating the future energy deficit over all base stations. The energy deficit of a base station weighs a prediction of future demand against a prediction of the available energy level corresponding to the remaining range of stationed ambulances. We conducted experiments in a realistic scenario using an event-based simulator based on real-world incidents. Results show that MED reaches better performance than compared DAR methods, as well as baselines for EV settings. Furthermore, our results indicate that transitioning to electric ambulances can be done without increasing the number of available ambulances while maintaining comparable response times.

For future work, we plan to explore using more sophisticated prediction methods for demand and available energy. Furthermore, we want to examine sequential planning approaches considering multiple decisions in advance.

REFERENCES

- [1] San Francisco Emergency Medical Services Agency. [n.d.]. *San Francisco EMS Ambulance Destinations*. <http://sfemergencymedicalresponse.weebly.com/ambulance-destinations.html>
- [2] San Francisco Emergency Medical Services Agency. 2011. *Prehospital Provider Standards (Policy 4000)*. <https://www.sfdph.org/dph/files/EMS/Policy-Protocol-Manuals/Policy-Manual/1538-4000ProviderStandards09-01-2011.pdf>
- [3] Pieter L Van Den Berg and J Theresia Van Essen. 2019. Comparison of static ambulance location models. *International Journal of Logistics Systems and Management* 32, 3-4 (2019), 292–321.
- [4] Lawrence H Brown, Christa L Whitney, Richard C Hunt, Michael Addario, and Troy Hogue. 2000. Do warning lights and sirens reduce ambulance response times? *Prehospital Emergency Care* 4, 1 (2000), 70–74.
- [5] Andreas Bürger, Jan Wnent, Andreas Bohn, Tanja Jantzen, Sigrid Brenner, Rolf Lefering, Stephan Seewald, Jan-Thorsten Gräsner, and Matthias Fischer. 2018. The effect of ambulance response time on survival following out-of-hospital cardiac arrest: an analysis from the German resuscitation registry. *Deutsches Ärzteblatt International* 115, 33-34 (2018), 541.
- [6] Nabil Channouf, Pierre L'Ecuyer, Armann Ingólfsson, and Athanasios N Avramidis. 2007. The application of forecasting techniques to modeling emergency medical system calls in Calgary, Alberta. *Health care management science* 10 (2007), 25–45.
- [7] Richard Church and Charles ReVelle. 1974. The maximal covering location problem. In *Papers of the regional science association*, Vol. 32. Springer-Verlag Berlin/Heidelberg, 101–118.
- [8] Mark S Daskin. 1983. A maximum expected covering location model: formulation and heuristic solution. *Transportation science* 17, 1 (1983), 48–70.
- [9] C Dugas, Y Bengio, F Bélisle, and C Nadeau. 2001. Incorporating second order functional knowledge into learning algorithms. *Advances in Neural Information Processing Systems* 13 (2001), 472–478.
- [10] Shakiba Enayati, Maria E Mayorga, Hari K Rajagopalan, and Cem Saydam. 2018. Real-time ambulance redeployment approach to improve service coverage with fair and restricted workload for EMS providers. *Omega* 79 (2018), 67–80.
- [11] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. 2001. A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel computing* 27, 12 (2001), 1641–1653.
- [12] Jeffrey Goldberg, Robert Dietrich, Jen Ming Chen, M George Mitwasi, Terry Valenzuela, and Elizabeth Criss. 1990. Validating and applying a model for locating emergency medical vehicles in Tucson, AZ. *European Journal of Operational Research* 49, 3 (1990), 308–324.
- [13] Caroline J Jagtenberg, Sandjai Bhulai, and Robert D van der Mei. 2015. An efficient heuristic for real-time ambulance redeployment. *Operations Research for Health Care* 4 (2015), 27–35.
- [14] Shengdong Ji, Yu Zheng, Zhaoyuan Wang, and Tianrui Li. 2019. A deep reinforcement learning-enabled dynamic redeployment system for mobile ambulances. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 1 (2019), 1–20.
- [15] Olive C Kobusingye, Adnan A Hyder, David Bishai, Manjul Joshipura, Eduardo Romero Hicks, and Charles Mock. 2006. *Emergency medical services. Disease Control Priorities in Developing Countries. 2nd edition* (2006).
- [16] Matthew S Maxwell, Mateo Restrepo, Shane G Henderson, and Huseyin Topaloglu. 2010. Approximate dynamic programming for ambulance redeployment. *INFORMS Journal on Computing* 22, 2 (2010), 266–281.
- [17] Laura A McLay and Maria E Mayorga. 2010. Evaluating emergency medical service performance measures. *Health care management science* 13 (2010), 124–136.
- [18] Jill P Pell, Jane M Sirel, Andrew K Marsden, Ian Ford, and Stuart M Cobbe. 2001. Effect of reducing ambulance response times on deaths from out of hospital cardiac arrest: cohort study. *Bmj* 322, 7299 (2001), 1385–1388.
- [19] Charles ReVelle and Kathleen Hogan. 1989. The maximum availability location problem. *Transportation science* 23, 3 (1989), 192–200.
- [20] Emmanouil S Rigas, Antonis Billis, and Panagiotis D Bamidis. 2022. Can Artificial Intelligence Enable the Transition to Electric Ambulances? In *Challenges of Trustable AI and Added-Value on Health*. IOS Press, 73–77.
- [21] Hubert Setzler, Cem Saydam, and Sungjune Park. 2009. EMS call volume predictions: A comparative study. *Computers & Operations Research* 36, 6 (2009), 1843–1851.
- [22] Krisjanis Steins, Niki Matinrad, and Tobias Granberg. 2019. Forecasting the demand for emergency medical services. (2019).
- [23] Niklas Strauß, Max Berrendorf, Tom Haider, and Matthias Schubert. 2022. A Comparison of Ambulance Redeployment Systems on Real-World Data. In *Proceedings of the 1st Workshop on Urban Internet-of-Things Intelligence (UNIT 2022) co-located with the 22nd IEEE International Conference on Data Mining (ICDM 2022)*.
- [24] Niklas Strauß, Lukas Rottkamp, Sebastian Schmolli, and Matthias Schubert. 2021. Efficient Parking Search using Shared Fleet Data. In *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 115–120.
- [25] Zhaonan Wang, Tianqi Xia, Renhe Jiang, Xin Liu, Kyoung-Sook Kim, Xuan Song, and Ryosuke Shibusaki. 2021. Forecasting ambulance demand with profiled human mobility via heterogeneous multi-graph neural networks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 1751–1762.
- [26] Zhengyi Zhou and David S Matteson. 2015. Predicting ambulance demand: A spatio-temporal kernel approach. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2297–2303.

E Constrained Portfolio Management Using Action Space Decomposition for Reinforcement Learning

Venue 2023 Pacific-Asia Conference on Knowledge Discovery and Data Mining






DOI https://doi.org/10.1007/978-3-031-33377-4_29

Declaration of authorships The research idea was proposed by David Winkel and developed and discussed with Niklas Stauss and Matthias Schubert. The other co-authors also participated in some of the discussions. David Winkel wrote the manuscript, which was improved by Niklas Strauss and Matthias Schubert.

Publication



Constrained Portfolio Management Using Action Space Decomposition for Reinforcement Learning

David Winkel^{1,2} , Niklas Strauß^{1,2} , Matthias Schubert^{1,2} ,
Yunpu Ma^{1,2} , and Thomas Seidl^{1,2} 

¹ LMU Munich, Munich, Germany

{winkel,strauss,schubert,ma,seidl}@dbs.ifi.lmu.de

² Munich Center for Machine Learning (MCML), Munich, Germany

Abstract. Financial portfolio managers typically face multi-period optimization tasks such as short-selling or investing at least a particular portion of the portfolio in a specific industry sector. A common approach to tackle these problems is to use constrained Markov decision process (CMDP) methods, which may suffer from sample inefficiency, hyperparameter tuning, and lack of guarantees for constraint violations. In this paper, we propose Action Space Decomposition Based Optimization (ADBO) for optimizing a more straightforward surrogate task that allows actions to be mapped back to the original task. We examine our method on two real-world data portfolio construction tasks. The results show that our new approach consistently outperforms state-of-the-art benchmark approaches for general CMDPs.

Keywords: Reinforcement Learning · Constrained Action Space · Decomposition · CMDP · Portfolio Optimization

1 Introduction

Constrained portfolio optimization is an important problem in finance. A typical example is a portfolio that must have at least 40% of the total portfolio value invested in environmentally friendly companies at each time step of the investment horizon or a portfolio that is not permitted to invest more than 20% in a particular industry sector. Another example of an action constraint task is a 130-30 strategy, in which the portfolio manager bets on group A of (potentially) overperforming stocks against group B of (potentially) underperforming stocks. This strategy is carried out by short-selling stocks worth 30% of the investment budget from Group B and leveraging the investment into stocks worth 130% of the investment budget from Group A.

The action space for these tasks can be considered as a continuous distribution of weights for a given set of assets. Therefore, reinforcement learning (RL) with policy gradient [16] is well-suited for this task. Because the invested capital

374 D. Winkel et al.

totals 100%, the codomain of the policy function is typically assumed to be a standard simplex. Existing solutions model the policy using a softmax output layer [1] or based on the Dirichlet distribution [20]. However, the constraints mentioned above cause a change in the shape of the policy’s codomain, making the standard solutions no longer directly applicable.

A way to optimize policies for tasks with constrained action spaces is by using approaches for CMDPs with constraints on the action spaces. However, state-of-the-art general approaches for CMDPs often have drawbacks such as expensive training loops, sample inefficiency, or only guarantees for asymptotical constraint compliance [2, 4, 10, 19, 21].

In this paper, we propose ADBO, a dedicated approach for dealing with the two important types of investment tasks mentioned previously: (a) investment tasks that invest *at least* or *at most* a certain percentage of a portfolio in a specific group of assets, and (b) short-selling tasks. ADBO can overcome the aforementioned shortcomings of general policy optimization methods for CMDPs. This is achieved by decomposing the non-standard-simplex action space into a surrogate action space. Solutions found in the surrogate action space can then be mapped back into the original constrained action space. In contrast to the non-standard-simplex action space, the surrogate action space is designed to be easily represented in the policy function approximator, allowing us to model the problem as a standard Markov decision process (MDP). Due to the lack of penalties and reward shaping, finding an optimal policy for an MDP is less complex than finding an optimal policy for a CMDP with constrained actions. Furthermore, ADBO ensures that the actions adhere to the constraints both during and after training.

In the experimental section, we demonstrate that the ADBO approach can handle two types of investment tasks using real-world financial data. The first task focuses on investing each time step at least a certain percentage of the portfolio in companies considered to be environmentally sustainable. The second task allows the agent to short-sell selected stocks, i.e., allowing for negative portfolio weights. Our proposed approach outperforms the state-of-the-art benchmark approaches for handling CMDPs on various criteria in both tasks.

2 Related Work

CMDPs were introduced by [3] to model constrained sequential decision tasks. constrained Reinforcement Learning (CRL) approaches for finding optimal policies for CMDPs have a wide range of applications, including finance [7, 20], autonomous electric vehicle routing [14], network traffic [9], and robotics [2, 8]. A **Trust Region**-based approach was introduced by [2] to find optimal policies for CMDPs that may still exhibit constraint violation due to approximation errors. Another approach proposed by [6] is based on **prior knowledge** and involves a one-time pretraining to predict simple one-step dynamics of the environment. **Lagrangian-based** approaches are another option for dealing with CMDPs. These approaches convert the original constraint optimization problem

into an unconstrained optimization problem by applying a Lagrangian relaxation to the constraints. Lagrangian-based approaches can be classified into two types: The first type is **Primal-Dual algorithms**, in which the Lagrange multipliers for a saddle point problem are chosen *dynamically* [5, 19]. The second type of Lagrangian-based approach employs **manually selected Lagrange multipliers**, which remain *static*, as shown in [13, 17]. Instead of a *saddle point problem*, as in the first type, using a static Lagrange multiplier transforms the problem into a maximization problem, which is more stable and computationally less expensive to solve. Some approaches carefully select Lagrange multipliers to model preferences in a trade-off problem rather than as a means to enforce constraints in an optimization problem. This is commonly seen in risk-return trade-off settings, such as in [7, 17, 20].

The **factorization of high-dimensional action spaces** in RL, i.e., splitting action spaces into smaller sub-action spaces as a Cartesian product, is an active area of research that has resulted in improved scalability and training performance. In their work, [11] introduce the Sequential DQN approach, which trains the agent for a sequence of n 1-dimensional actions rather than training the agent for n -dimensional actions of the original action space, effectively factorizing the original action space. The approach by [18] introduces an action branching architecture, which models the policies for the sub-action spaces in parallel. Our approach, like theirs, uses a Cartesian product of sub-action spaces. However, the sub-action spaces in our new approach ADBO are the outcome of a decomposition based on the Minkowski sum, resulting in a surrogate action space rather than a factorization of the original action space.

3 Problem Setting

We consider an agent that needs to allocate wealth across N different assets over T time steps. The allowed actions of the agent are defined by the investor's investment task and are contained in the *constrained action space* \mathcal{A} . The investment task type $T1$ requires the investor to invest at least c_{T1} of the portfolio into assets from group V_{T1} . In practice, these group definitions are often linked to individual risk profiles, industry sectors, or features such as being an environmentally friendly investment. The action space for investment task type $T1$ is then defined as

$$\mathcal{A}_{T1} = \left\{ a \in \mathbb{R}^N : \sum_{i=0}^{N-1} a_i = 1, \sum_{i \in V_{T1}} a_i \geq c_{T1}, a_i \geq 0, c_{T1} > 0 \right\}$$

and represents an N -dimensional convex polytope. Task type $T1$ also includes cases that require investing *at most* c_{T1} into assets in V_{T1} because this case is equivalent to investing *at least* $(1 - c_{T1})$ into the remaining assets a_i for $i \in I \setminus V_{T1}$.

The investment task type $T2$ represents investors who believe that a group of assets V_{T2} will underperform relatively compared to the rest of the investment universe I . The investor pays a borrowing fee to short-sell assets in group

376 D. Winkel et al.

V_{T2} worth $|c_{T2}|$ of his total portfolio value and then uses the freed-up cash to invest $1 + |c_{T2}|$ into assets of the other investment universe. The action space for investment task type $T2$ is defined as

$$\mathcal{A}_{T2} = \left\{ a \in \mathbb{R}^N : \sum_{i=0}^{N-1} a_i = 1, \sum_{j \in V_{T2}} a_j = c_{T2}, a_j \leq 0, a_k \geq 0 \forall k \in I \setminus V_{T2}, c_{T2} < 0 \right\}$$

and represents an N -dimensional convex polytope as well.

The **observation space** is defined as $\mathcal{O} = \mathcal{W} \times \mathcal{V} \times \mathcal{U}$ where $\mathcal{W} \subseteq \mathbb{R}^+$ is the current absolute wealth level, $\mathcal{V} \subseteq \mathbb{R}^N$ is the current relative portfolio weight of each of the N assets, and $\mathcal{U} \subseteq \mathbb{R}^N$ represents all the observed single asset returns from the previous time step.

The economic return of each asset is individually modeled for each time step by the random vector $\Theta = [\Theta_0, \dots, \Theta_{N-1}] \in \mathcal{U}$. The portfolio return is then a random variable with an expected value denoted as $\mathbb{E}[\Theta_{PF}] = a^\top \mathbb{E}[\Theta]$ with the portfolio weights $a \in \mathcal{A}$. There are two potential sources of cost to consider for the agent: First, the transaction costs caused by changes in the portfolio weights a_t in time step t by the agent defined as $tc_t = (|a_t - v_t|)^\top c$, where $v_t \in \mathcal{V}$ and vector $c = [c_0, \dots, c_{N-1}]$ represents the asset-specific transaction costs caused by trading a specific asset. Second, borrowing fees in case the agent is allowed to short-sell assets. These costs occur every period as long as assets are short-sold, i.e., assigned to a negative portfolio weight. The borrowing fees are defined as $bf_t = (\mathbb{1}_{a_i < 0} \circ a_t)^\top b$ where $\mathbb{1}_{a_i < 0}$ is an indicator vector signaling for each individual asset a_i if the current portfolio weight is negative, \circ is an operator for element-wise vector multiplication, and the vector $b = [b_0, \dots, b_{N-1}]$ represents asset-specific borrowing fees per time step.

The **reward** for the agent is a combination of transaction costs tc , borrowing fees bf , and a realization ϑ_{PF} of the random variable of the portfolio's economic return Θ_{PF} , i.e., $r = \vartheta_{PF} - tc - bf$. The agent's goal is to maximize the expected cumulative reward, which we will refer to as *total economic payoff*.

4 Solution as CMDP

A CMDP is an extension of an MDP and is defined as a tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma, \mathcal{C})$ where \mathcal{S} is the set of states, \mathcal{A} is the set of actions, R is the immediate reward function, which maps transition tuples to their respective expected reward, i.e., $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. P denotes the transition probability function, whereas $P(s_{t+1}|s_t, a_t)$ gives the probability of transitioning to state $s_{t+1} \in \mathcal{S}$ given state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$. The parameter $\gamma \in [0, 1)$ represents a discount factor. $\mathcal{C} = \{C_0, \dots, C_m\}$ is a set of immediate constraint functions $C_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ for $i \in \{0, \dots, m\}$ that map transition tuples to the respective cost. We let $r_{t+1} := R(s_t, a_t, s_{t+1})$ and define the return for a trajectory τ as the observed discounted cumulative rewards. The objective function J is then defined as the expected return for a given policy π , i.e., $J(\pi) := \mathbb{E}_{\tau \sim P(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_{t+1} \right]$.

The expected cumulative discounted immediate cost for constraint i under policy π is defined as $J_{C_i}(\pi) := \mathbb{E}_{\tau \sim P(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t C_i(s_t, a_t, s_{t+1}) \right]$. We also define constant trajectory constraint limits d_0, \dots, d_m . The optimization problem for the CMDP is then defined as:

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & J(\pi) = \mathbb{E}_{\tau \sim P(\tau|\pi)} (G) = \mathbb{E}_{\tau \sim P(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_{t+1} \right] \\ \text{s.t.} \quad & J_{C_i}(\pi) \leq d_i \quad \forall i \end{aligned}$$

In the following, we will show how to formulate the tasks defined in Sect. 3 as a CMDP. In Sect. 3, we defined the observation space \mathcal{O} , the constrained action space \mathcal{A}_i , and the reward R . The transition function P and the state space \mathcal{S} are unknown. However, we assume that we can sample transitions from an environment. Therefore, we can employ reinforcement learning based on a learned state representation function to learn effective policies. To address the action constraints of tasks $T1$ and $T2$, we define the following cost function for each respective task $i \in \{1, 2\}$: $C_{T_i}(s_t, a_t, s_{t+1}) = \mathbb{1}_{a_t \notin \mathcal{A}_{T_i}} \cdot \zeta$ where constant $\zeta > 0$ indicates the non-zero cost of a constraint violation. The respective constraint for each task is then defined as $J_{C_{T_i}}(\pi) \leq 0$.

5 Action Space Decomposition Based Optimization

We define a surrogate MDP $(\mathcal{S}, \tilde{\mathcal{A}}, R, P, \gamma)$ and ensure that there exists a surjective function $f : \tilde{\mathcal{A}} \rightarrow \mathcal{A}$ that allows reaching any $a \in \mathcal{A}$ from at least one $\tilde{a} \in \tilde{\mathcal{A}}$. For a formal description of our method, we first introduce the Minkowski sum:

Definition 1. Given two sets A and B of vectors in n -dimensional Euclidean space, the **Minkowski sum** of A and B is generated by adding each vector in A to each vector in B , i.e., the set $A + B = \{a + b | a \in A, b \in B\}$ in which we refer to A and B as Minkowski summands.

In our setting, the Minkowski sum describes how multiple decomposed action sets can be combined to reconstruct the original constraint action set. The masked scaled standard simplex (MSSS) describes a part of the original constrained action which can be described as a simplex:

Definition 2. Let mask $M \subseteq \{0, \dots, N - 1\}$. MSSS is defined as:

$$MSSS_{M,c} = \left\{ y \in \mathbb{R}^N : \sum_{j \in M} y_j = c, y_i = 0 \forall i \in I \setminus M \right\}$$

with either $(c \geq 0 \wedge y_i \geq 0 \forall i \in M)$ or $(c < 0 \wedge y_i \leq 0 \forall i \in M)$.

The surrogate action space is modeled as the Cartesian product of independent sub-action spaces $\tilde{\mathcal{A}} = \tilde{\mathcal{A}}_1 \times \tilde{\mathcal{A}}_2$. The sub-action spaces $\tilde{\mathcal{A}}_i$ with $i \in \{1, 2\}$ are required to have the two properties: (a) being a decomposition of \mathcal{A} in such a way

378 D. Winkel et al.

that the Minkowski sum (see Definition 1) of all the sub-action spaces $\tilde{\mathcal{A}}_i$ is \mathcal{A} , i.e., $\mathcal{A} = \tilde{\mathcal{A}}_1 + \tilde{\mathcal{A}}_2$, and (b) being an MSSS as defined in Definition 2. Property (a) guarantees the existence of function f that can be defined as $f(\tilde{a}) = \tilde{a}_1 + \tilde{a}_2 = a$ with $\tilde{a} = [\tilde{a}_1, \tilde{a}_2] \in \tilde{\mathcal{A}} \subset \mathbb{R}^{2N}$ and $\tilde{a}_i \in MSSS_i \subset \mathbb{R}^N$ for $i \in \{1, 2\}$, i.e., a summation of vectors in a subspace of \mathbb{R}^N . Property (b) allows utilizing well-established RL methods for handling standard simplex action spaces with only minor modifications by adding a scaling and masking logic in order to model single *MSSS* action spaces.

The following two theorems show that constrained action spaces as defined in Sect. 3 can be decomposed into two MSSS that satisfy both of the requirements mentioned above. Theorem 1 describes the decomposition for task *T1*:

Theorem 1. *Any convex polytope $P \neq \emptyset$ defined as*

$$\sum_{i \in I} x_i = 1, x_i \geq 0 \quad \forall i \in I, \sum_{i \in V_1} x_i \geq c_1$$

with $c_1 > 0$, $I = \{0, \dots, N-1\}$ and $V_1 \subseteq I$ can be decomposed into two MSSSs:

$$MSSS_{S_1, z_1}, \text{ i.e. } \forall y_1 \in MSSS_{S_1, z_1} : \sum_{S_1} y_{i,1} = z_1 \quad \text{with } S_1 = V_1 \text{ and } z_1 = c_1$$

$$MSSS_{S_2, z_2}, \text{ i.e. } \forall y_2 \in MSSS_{S_2, z_2} : \sum_{S_2} y_{i,2} = z_2 \quad \text{with } S_2 = I \text{ and } z_2 = 1 - c_1$$

so that the Minkowski sum of the MSSSs equals the original polytope P .

Correspondingly, the following theorem formulates the decomposition of the action space in task *T2*:

Theorem 2. *Any convex polytope $P \neq \emptyset$ defined as*

$$\sum_{i \in I} x_i = 1, \sum_{i \in V_1} x_i = c_1, x_i \geq c_1 \quad \forall i \in V_1, x_i \leq 0 \quad \forall i \in V_1, x_i \geq 0 \quad \forall i \in I \setminus V_1$$

with $c_1 < 0$, $I = \{0, \dots, N-1\}$ and $V_1 \subseteq I$ can be decomposed into two MSSSs:

$$MSSS_{S_1, z_1}, \text{ i.e. } \forall y_1 \in MSSS_{S_1, z_1} : \sum_{S_1} y_{i,1} = z_1 \quad \text{with } S_1 = V_1 \text{ and } z_1 = c_1$$

$$MSSS_{S_2, z_2}, \text{ i.e. } \forall y_2 \in MSSS_{S_2, z_2} : \sum_{S_2} y_{i,2} = z_2 \quad \text{with } S_2 = I \setminus V_1 \text{ and } z_2 = 1 - c_1$$

so that the Minkowski sum of the MSSSs equals the original polytope P .

Theorem 1 and 2 can be proven by showing that the two sets of closed half-spaces, one describing the polytope P and the other describing the Minkowski sum of the two MSSSs, are equal resulting in the equality of the two polytopes.

ADBO is based on the PPO algorithm [15]. The agent's policy network is designed in such a way that the action representation is distributed across multiple *independent* segments, i.e., one head for each MSSS. A *shared* state encoder,

on the other hand, provides a learned state representation to both heads. For the state encoder we use a neural network of four fully connected layers of size 1024, 512, 256, and 64 with ReLU activation functions followed by GTrXL element allowing to handle tasks requiring memory. The GTrXL element is based on [12]. The element is composed of a single *transformer unit* with a single encoder layer as well as a single decoder layer with four attention heads and an embedding size of 64. While the sub-actions in ADBO are stochastically independent, the parameters of the two distributions from which the sub-actions are drawn are partially coordinated, i.e., parts of the actions rely on the same shared latent state representation. To further ensure coordination between the sub-actions, the different sub-actions are all evaluated using a joint reward. This means that if a *joint action* performs poorly, all independent segments receive a poor reward signal, regardless of individual sub-action performance.

We use a Dirichlet distribution to model each MSSS in the architecture of the policy function approximator. The expected value of a random vector $X = [X_0, \dots, X_{N-1}]$ following a Dirichlet distribution with a parameter vector of $\alpha = [\alpha_0, \dots, \alpha_{N-1}]$ is defined as $\mathbb{E}[X_i] = \alpha_i \cdot \left(\sum_{n=0}^{N-1} \alpha_n \right)^{-1}$ with $\alpha_i > 0$ for $i \in \{0, \dots, N-1\}$. By adjusting the parameter vector of a Dirichlet distribution and applying a linear scaling transformation, we can create a random variable with the set of all possible realizations equaling $MSSS_{M,c}$. The set M contains index values which we map to an N -dimensional indicator vector $\mathbb{1}_M$, with the vector's elements set to one if their respective index occurs in M and zero otherwise. The parameter vector passed to the Dirichlet distribution is calculated as $\alpha_{\mathbb{1}_M} = \max(\alpha \circ \mathbb{1}_M, \epsilon)$, where α is the initial parameter vector before applying the masking and $\epsilon > 0$ is an arbitrary small number. The operator \circ represents element-wise multiplication for vectors. In the final step, a linear scaling transformation is applied, i.e., $Y = c \cdot X$ with $X \sim \text{Dir}(\alpha_{\mathbb{1}_M})$.

ADBO requires the uses of two MSSSs, i.e., $MSSS_{M_1,c_1}$ and $MSSS_{M_2,c_2}$. It should be noted that the gradient of the policy during training is based on a policy interacting with the surrogate action space $\tilde{\pi}(\cdot|s)$ rather than a policy interacting with the original constrained action space $\pi(\cdot|s)$. We only use f to convert \tilde{a} into a representation a that can interact with the environment. Various inputs \tilde{a} for f may sum to the same output value a , resulting in f being a many-to-one function. For some $\tilde{a} \in \tilde{\mathcal{A}}$, this results in $\mathbb{P}(\tilde{a}|s) \neq \mathbb{P}(a|s)$ with $a = f(\tilde{a})$. However, we argue that finding *one* possible representation for an action a belonging to an optimal policy for the original problem is sufficient from an optimization standpoint.

6 Experiments

The environment is based on [20], and uses the same real-world financial data from the Nasdaq-100 index that was fetched and processed using the qlib package.¹ The investment universe of the environment consists of 13 assets, one of

¹ <https://github.com/microsoft/qlib/tree/main>.

380 D. Winkel et al.

which is cash. The remaining 12 assets are chosen at random from a list of 35 stocks that remain after filtering the Nasdaq-100 data set for companies that have been part of the index since January 1, 2010 and have no missing data. The original Nasdaq-100 data set is supplemented with the Environmental Score Metric (ESM) assigned by financial data provider LSEG.² The score rates a company’s environmental sustainability based on various evaluation categories, such as carbon emissions, willingness to innovate in this field, and transparency in reporting relevant information. The score ranges from 0 to 100, representing the percentiles of a ranking system.

We compare **ADBO** to three other state-of-the-art approaches for optimizing policies in CMDPs. **RCPO** is proposed by [19] and belongs to the class of Lagrangian-based approaches. The interior-point policy optimization approach **IPO** is introduced by [10]. **P3O** is proposed by [21] and uses a first-order optimization over an unconstrained objective with a penalty term equal to the original constraint objective. All benchmark approaches are implemented in the RLlib framework³ based on their papers and publically available.⁴

Two experimental settings are examined: the **SUSTA setting** is based on task type *T1*. The investor must invest at least 40% of his capital in the top 20% of environmentally sustainable companies, i.e., companies with an ESM score of 80 or higher. A score of 80 or higher “indicates excellent relative [...] performance and a high degree of transparency in reporting material” by a company.⁵ The **SHORT setting** is based on task type *T2*. It employs a **130-30 strategy**, a popular long/short equity strategy among investors to invest 130% of the available capital in stocks they believe will outperform and short-sell stocks worth 30% of the available capital they believe will underperform. In the experiments, we choose the companies Automatic Data Processing Inc., Paccar Inc., and Amgen Inc. to be sold short based on being the worst performers in 2020, the final year before the start of the backtesting period.

\mathcal{A}_{short} is not a subset of the standard simplex because negative weights are permitted. As a result, the RCPO, IPO, and P3O approaches must be modified to be applicable to SHORT setting. The agent performed very poorly in initial tests using \mathbb{R}^N as a base action space and applying constraints accordingly and was unable to learn meaningful policies. Instead, using a standard simplex as the base action space and applying action scaling produced better results. For *action scaling*, the agent uses the output of a Dirichlet distribution as an encoded action $\tilde{a} = [\tilde{a}_0, \dots, \tilde{a}_{N-1}]$ that is then transformed, i.e., scaled into the final action $a = [a_0, \dots, a_{N-1}]$: the cumulative weights of the stocks sold *short* and the cumulative weights of the stocks bought *long* are added up in their absolute values, resulting in a scaling factor $\alpha_{total} = |\alpha_{long}| + |\alpha_{short}|$. Then, for all elements i of the encoded action $a_i = \tilde{a}_i \cdot \alpha_{total}$ that are bought, a positive scaling factor is applied, and for all elements j of the encoded action $a_j = \tilde{a}_j \cdot (-\alpha_{total})$

² <https://www.lseg.com/>.

³ <https://docs.ray.io/en/master/rllib/index.html>.

⁴ https://github.com/DavWinkel/RL_ADBO.

⁵ <https://www.refinitiv.com/en/sustainable-finance/esg-scores>.

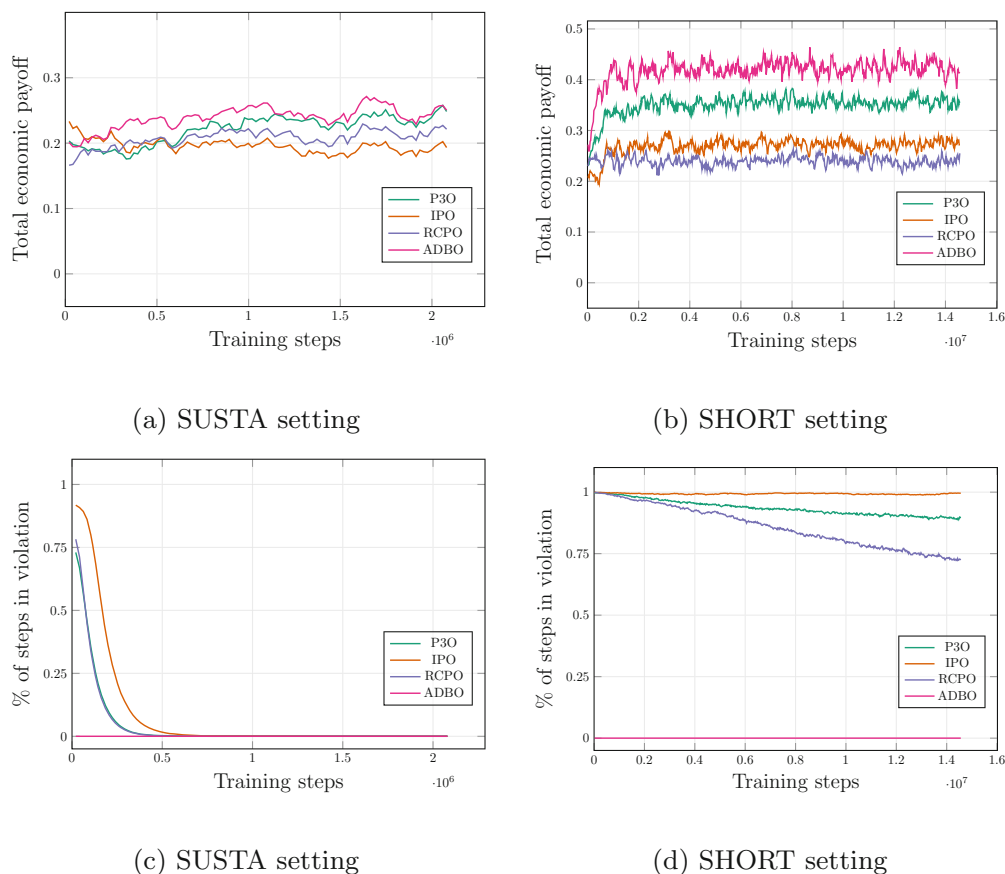


Fig. 1. Performance during training for all four approaches in the SUSTA setting and the SHORT setting regarding Total economic payoff and % of steps in violation.

that are sold short, a negative scaling factor is applied. It should be noted that actions generated as described above are no longer guaranteed to sum up to 1.0. Because IPO is a logarithmic barrier function-based approach that does not apply to equality constraints, we must additionally soften equality constraints of the form $x = c$ to inequality constraints that allow values in a α -neighborhood of x , i.e., $x \leq c + \alpha$ and $x \geq c - \alpha$.

To evaluate the four approaches, we will report performance during and after training for both the SUSTA setting and the SHORT setting. The total economic payoff defined in Sect. 3 is used to measure economic performance. The results of the SUSTA setting will be discussed first. The training in the SUSTA setting lasts 500 iterations and consists of approximately 2.1 million training steps. Figure 1a shows that ADBO and P3O perform best during training by steadily improving their total economic payoff. RCPO also shows improvements, although at a much slower rate. Table 1 shows the evaluation of economic performance following training completion in two setups: in the **(A) environment setup** 1000 trajectories are sampled from the same environment used for the training. ADBO generates the highest total economic payoff in the SUSTA setting, followed by

382 D. Winkel et al.

P3O, RCPO, and IPO. In the **(B) backtesting setup** a single trajectory, namely the real-world Nasdaq-100 trajectory in 2021, is used for evaluation. In the backtesting year 2021, the overall yearly performance of the Nasdaq-100 index was above average, returning 27.5%, indicating that the individual stocks that comprise the index were also performing well. As a result, the four approaches generated high returns in the (B) backtesting setup, with ADBO performing best, followed by P3O. In the SHORT setting, the training time had to be increased significantly. This increase was required because IPO, RCPO, and P3O failed to generate constraint-compliant actions satisfactorily. However, due to insufficient training progress, which will be discussed in detail later in this section, the training was eventually stopped after 3500 iterations, consisting of approximately 14.7 million training steps. Figure 1b depicts the evolution of the total economic payoff during training. After roughly 1 million training steps, the performance of ADBO converges to a level that it then maintains for the remainder of the training. P3O improves its performance over 3 million training steps until it reaches a stable level. IPO improves its performance during the first million training steps and then stabilizes, whereas RCPO does not show significant improvements in total economic payoff during training. Table 1 shows the performance evaluation in the SHORT setting after the training is completed. In the (A) environment setup, ADBO performs best, with an average total economic payoff of 42.72%, followed by P3O with 35.12%. ADBO outperforms its benchmark approaches by a wide margin in the (B) backtesting setup, achieving a total economic payoff of 102.05%.

The experiments show that violations of the action constraints occurred during the training of IPO, RCPO, and P3O in the SUSTA setting. Figure 1c shows that this is especially true at the beginning of the training phase, while the number of time steps with actions in violation decreases almost to zero later on. After completion of the training in the (A) environment setup, RCPO is the only approach generating actions in violations, as shown in Table 1. However, violations occur only on a small number of time steps, i.e., nine out of 12'000 time steps. All approaches are free of constraint violations in the (B) backtesting setup. For the SHORT setting, the majority of actions generated by the approaches IPO, RCPO, and P3O violated the constraints during training. However, as training time progresses, the number of actions in constraint violation decreases for RCPO and P3O. As a result, the training time was increased sevenfold when compared to SUSTA setting. Nevertheless, the training was eventually halted due to insufficient speed in reducing constraint violations. Figure 1d shows the best-performing variants of the agents after extensive tuning of their hyperparameters. Table 1 displays the evaluation results after the training in the SHORT setting was completed. In the SHORT setting, IPO, RCPO, and P3O fail to generate results free of constraint violations for both the (A) environment and (B) backtesting setups. ADBO, on the other hand, guarantees by design actions free of violations during and after training.

Table 1. Evaluation after training is completed. (A) environment setup has a total of 12'000 time steps (1000 trajectories), (B) backtesting setup has a single trajectory with 12 time steps.

	SUSTA setting		SHORT setting	
	Total econ. payoff (12 months)	Total violations	Total econ. payoff (12 months)	Total violations
(A) environment				
RCPO	0.2238	0	0.2418	8656
IPO	0.2013	0	0.2721	11943
P3O	0.2561	9	0.3512	10865
ADBO (Ours)	0.2603	0	0.4272	0
(B) backtesting				
RCPO	0.4640	0	0.5285	9
IPO	0.3499	0	0.6262	12
P3O	0.5475	0	0.7654	11
ADBO (Ours)	0.5758	0	1.0205	0

7 Conclusion

In this paper, we train agents to manage investment portfolios over multiple periods, given two types of tasks that are commonly encountered in practice. Task type $T1$ constrains the allocation of a particular group of assets, e.g., assets belonging to a specific industry sector. Task type $T2$ requires the investor to short-sell one group of assets while increasing the investment in another. We propose ADBO, which finds a performant policy for a surrogate MDP rather than for the more complex CMDP. The surrogate MDP is based on an action space decomposition of the original action space. We show that ADBO outperforms general CMDP approaches for both task types in experimental settings. For future work, we will examine extensions of action space decomposition based on the Minkowski sums to a broader group of convex polytopes.

References

1. Abrate, C., et al.: Continuous-action reinforcement learning for portfolio allocation of a life insurance company. In: Dong, Y., Kourtellis, N., Hammer, B., Lozano, J.A. (eds.) ECML PKDD 2021. LNCS (LNAI), vol. 12978, pp. 237–252. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86514-6_15
2. Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained policy optimization. In: International Conference on Machine Learning, pp. 22–31. PMLR (2017)
3. Altman, E.: Constrained Markov decision processes: stochastic modeling. Routledge (1999)
4. Ammar, H.B., Tutunov, R., Eaton, E.: Safe policy search for lifelong reinforcement learning with sublinear regret. In: International Conference on Machine Learning, pp. 2361–2369. PMLR (2015)

384 D. Winkel et al.

5. Bhatnagar, S., Lakshmanan, K.: An online actor-critic algorithm with function approximation for constrained Markov decision processes. *J. Optim. Theory Appl.* **153**(3), 688–708 (2012)
6. Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., Tassa, Y.: Safe exploration in continuous action spaces. arXiv preprint [arXiv:1801.08757](https://arxiv.org/abs/1801.08757) (2018)
7. Di Castro, D., Tamar, A., Mannor, S.: Policy gradients with variance related risk criteria. arXiv preprint [arXiv:1206.6404](https://arxiv.org/abs/1206.6404) (2012)
8. Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3389–3396. IEEE (2017)
9. Hou, C., Zhao, Q.: Optimization of web service-based control system for balance between network traffic and delay. *IEEE Trans. Autom. Sci. Eng.* **15**(3), 1152–1162 (2017)
10. Liu, Y., Ding, J., Liu, X.: Ipo: Interior-point policy optimization under constraints. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 4940–4947 (2020)
11. Metz, L., Ibarz, J., Jaitly, N., Davidson, J.: Discrete sequential prediction of continuous actions for deep RL. arXiv preprint [arXiv:1705.05035](https://arxiv.org/abs/1705.05035) (2017)
12. Parisotto, E., et al.: Stabilizing transformers for reinforcement learning. In: International Conference on Machine Learning, pp. 7487–7498. PMLR (2020)
13. Peng, X.B., Abbeel, P., Levine, S., Van de Panne, M.: DeepMimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph. (TOG)* **37**(4), 1–14 (2018)
14. Qin, Z., Chen, Y., Fan, C.: Density constrained reinforcement learning. In: International Conference on Machine Learning, pp. 8682–8692. PMLR (2021)
15. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) (2017)
16. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems 12 (1999)
17. Tamar, A., Mannor, S.: Variance adjusted actor critic algorithms. arXiv preprint [arXiv:1310.3697](https://arxiv.org/abs/1310.3697) (2013)
18. Tavakoli, A., Pardo, F., Kormushev, P.: Action branching architectures for deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
19. Tessler, C., Mankowitz, D.J., Mannor, S.: Reward constrained policy optimization. In: International Conference on Learning Representations (2018)
20. Winkel, D., Strauß, N., Schubert, M., Seidl, T.: Risk-aware reinforcement learning for multi-period portfolio selection. In: Amini, M.R., Canu, S., Fischer, A., Guns, T., Kralj Novak, P., Tsoumakas, G. (eds.) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2022. LNCS, vol. 13718. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-26422-1_12
21. Zhang, L., et al.: Penalized proximal policy optimization for safe reinforcement learning. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, pp. 3744–3750 (2022)

Constrained Portfolio Mgmt. Using Action Space Decomp. for RL 385

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



A Proofs

Proof (of theorem 1). If two polytopes P and M can be fully described by an equivalent system of equations and inequalities then they are equal. Given a convex polytope $P \neq \emptyset$ defined by the following system

$$\begin{aligned} \sum_{i \in I} x_i &= 1 \\ x_i &\geq 0 \quad \forall i \in I \\ \sum_{i \in V_1} x_i &\geq c_1 \end{aligned}$$

with $c_1 > 0$ (otherwise the constraint is a non-active constraint and can be dropped), $I = \{0, \dots, N-1\}$ and $V_1 \subseteq I$ and the set of two masked scaled standard simplices (MSSSs):

$$\begin{aligned} \text{MSSS}_{S_1, z_1}, \text{ i.e. } \forall y_1 \in \text{MSSS}_{S_1, z_1} : \sum_{S_1} y_{i,1} &= z_1 \quad \text{with } S_1 = V_1 \text{ and } z_1 = c_1 \\ \text{MSSS}_{S_2, z_2}, \text{ i.e. } \forall y_2 \in \text{MSSS}_{S_2, z_2} : \sum_{S_2} y_{i,2} &= z_2 \quad \text{with } S_2 = I \text{ and } z_2 = 1 - c_1 \end{aligned}$$

on which we apply the Minkowski sum $\text{MSSS}_{S_1, z_1} + \text{MSSS}_{S_2, z_2} = M$, we proof the equivalence of both systems describing the polytopes P and M . We note for the upper and lower bounds of the single variables in the elements of $y_1 \in \text{MSSS}_{S_1, z_1}$ and $y_2 \in \text{MSSS}_{S_2, z_2}$ that

$$\begin{aligned} y_{i,1} \leq z_1 &\Rightarrow y_{i,1}^{max} = z_1 && \text{for } i \in S_1 \\ y_{i,2} \leq z_2 &\Rightarrow y_{i,2}^{max} = z_2 && \text{for } i \in S_2 \\ y_{i,1} \geq 0 + z_1 \mathbb{1}_{|S_1|=1} &\Rightarrow y_{i,1}^{min} = 0 + z_1 \mathbb{1}_{|S_1|=1} && \text{for } i \in S_1 \\ y_{i,2} \geq 0 + z_2 \mathbb{1}_{|S_2|=1} &\Rightarrow y_{i,2}^{min} = 0 + z_2 \mathbb{1}_{|S_2|=1} && \text{for } i \in S_2 \end{aligned}$$

Additionally, we note that the resulting upper and lower bounds for the single variables x_i for $i \in I$ in the original polytope P are:

$$\begin{aligned} x_i &\leq 1 && \text{for } i \in V_1 \\ x_i &\leq 1 - c_1 && \text{for } i \in I \setminus V_1 \\ x_i &\geq 0 + c_1 \mathbb{1}_{|V_1|=1} + (1 - c_1) \mathbb{1}_{|I|=1} && \text{for } i \in V_1 \\ x_i &\geq 0 && \text{for } i \in I \setminus V_1 \end{aligned}$$

We can deduce that for any element $(y_1 + y_2) \in M$ with $y_1 \in \text{MSSS}_{S_1, z_1}$ and $y_2 \in \text{MSSS}_{S_2, z_2}$ the following conditions are fulfilled:

$$\begin{aligned} \sum_{V_1} x_i &= \underbrace{\sum_{V_1 \cap S_1} y_{i,1}}_{=z_1=c_1} + \underbrace{\sum_{V_1 \cap S_2} y_{i,2}}_{\geq 0} \geq c_1 \\ \sum_I x_i &= \underbrace{\sum_{I \cap S_1} y_{i,1}}_{=z_1=c_1} + \underbrace{\sum_{I \cap S_2} y_{i,2}}_{=z_2=1-c_1} = 1 \end{aligned}$$

Additionally, we also check the conditions for the single variables x_i for $i \in I$. For the upper and lower bound for any element $(y_1 + y_2) \in M$ the following is true

$$\begin{aligned}
 x_i = y_{i,1} + y_{i,2} &\leq \underbrace{y_{i,1}^{max} \mathbb{1}_{i \in S_1}}_{=z_1=c_1} + \underbrace{y_{i,2}^{max} \mathbb{1}_{i \in S_2}}_{=z_2=1-c_1} = 1 && \text{for } i \in V_1 \\
 x_i = y_{i,1} + y_{i,2} &\leq \underbrace{y_{i,1}^{max} \mathbb{1}_{i \in S_1}}_{=0} + \underbrace{y_{i,2}^{max} \mathbb{1}_{i \in S_2}}_{=z_2=1-c_1} = 1 - c_1 && \text{for } i \in I \setminus V_1 \\
 x_i = y_{i,1} + y_{i,2} &\geq \underbrace{y_{i,1}^{min} \mathbb{1}_{i \in S_1}}_{=z_1 \mathbb{1}_{|S_1|=1}=c_1 \mathbb{1}_{|V_1|=1}} + \underbrace{y_{i,2}^{min} \mathbb{1}_{i \in S_2}}_{=z_2 \mathbb{1}_{|S_2|=1}=(1-c_1) \mathbb{1}_{|I|=1}} = c_1 \mathbb{1}_{|V_1|=1} + (1 - c_1) \mathbb{1}_{|I|=1} && \text{for } i \in V_1 \\
 x_i = y_{i,1} + y_{i,2} &\geq \underbrace{y_{i,1}^{min} \mathbb{1}_{i \in S_1}}_{=0} + \underbrace{y_{i,2}^{min} \mathbb{1}_{i \in S_2}}_{=z_2 \mathbb{1}_{|S_2|=1}=(1-c_1) \mathbb{1}_{|I|=1}} = \underbrace{(1 - c_1) \mathbb{1}_{|I|=1}}_{=0} && \text{for } i \in I \setminus V_1
 \end{aligned}$$

Note that the bounds for $I \setminus V_1$ can only be active for $I \setminus V_1 \neq \emptyset$ which can only occur iff $|I| > 1$. As just shown the set P and M can be described by an same the system of equations and inequalities which proves that $P = M$.

Proof (of theorem 2). If two polytopes P and M can be fully described by an equivalent system of equations and inequalities then they are equal. Given a convex polytope $P \neq \emptyset$ defined by the following system

$$\begin{aligned}
 \sum_{i \in I} x_i &= 1 \\
 x_i &\geq c_1 \quad \forall i \in V_1 \\
 x_i &\leq 0 \quad \forall i \in V_1 \\
 x_i &\geq 0 \quad \forall i \in I \setminus V_1 \\
 \sum_{i \in V_1} x_i &= c_1
 \end{aligned}$$

with $c_1 < 0$, $I = \{0, \dots, N - 1\}$ and $V_1 \subseteq I$ and the set of two MSSSs:

$$\begin{aligned}
 MSSS_{S_1, z_1}, \text{ i.e. } \forall y_1 \in MSSS_{S_1, z_1} : \sum_{S_1} y_{i,1} &= z_1 \quad \text{with } S_1 = V_1 \text{ and } z_1 = c_1 \\
 MSSS_{S_2, z_2}, \text{ i.e. } \forall y_2 \in MSSS_{S_2, z_2} : \sum_{S_2} y_{i,2} &= z_2 \quad \text{with } S_2 = I \setminus V_1 \text{ and } z_2 = 1 - c_1 \geq 0
 \end{aligned}$$

on which we apply the Minkowski sum $MSSS_{S_1, z_1} + MSSS_{S_2, z_2} = M$, we proof the equivalence of both systems describing the polytopes P and M . We note for the upper and lower bounds of the single variables in the elements of $y_1 \in MSSS_{S_1, z_1}$ and $y_2 \in MSSS_{S_2, z_2}$ that

$$\begin{aligned}
 y_{i,1} \leq 0 + z_1 \mathbb{1}_{|S_1|=1} &\Rightarrow y_{i,1}^{max} = 0 + z_1 \mathbb{1}_{|S_1|=1} && \text{for } i \in S_1 \\
 y_{i,2} \leq z_2 &\Rightarrow y_{i,2}^{max} = z_2 && \text{for } i \in S_2 \\
 y_{i,1} \geq z_1 &\Rightarrow y_{i,1}^{min} = z_1 && \text{for } i \in S_1 \\
 y_{i,2} \geq 0 + z_2 \mathbb{1}_{|S_2|=1} &\Rightarrow y_{i,2}^{min} = 0 + z_2 \mathbb{1}_{|S_2|=1} && \text{for } i \in S_2
 \end{aligned}$$

Additionally, we note that the resulting upper and lower bounds for the single variables x_i for $i \in I$ in the original polytope P are:

$$\begin{aligned}
 x_i &\leq 0 + c_1 \mathbb{1}_{|V_1|=1} && \text{for } i \in V_1 \\
 x_i &\leq 1 - c_1 && \text{for } i \in I \setminus V_1 \\
 x_i &\geq c_1 && \text{for } i \in V_1 \\
 x_i &\geq 0 + (1 - c_1) \mathbb{1}_{|(I \setminus V_1)|=1} && \text{for } i \in I \setminus V_1
 \end{aligned}$$

We can deduce that for any element $(y_1 + y_2) \in M$ with $y_1 \in MSSS_{S_1, z_1}$ and $y_2 \in MSSS_{S_2, z_2}$ the following conditions are fulfilled:

$$\begin{aligned} \sum_{V_1} x_i &= \underbrace{\sum_{V_1 \cap S_1} y_{i,1}}_{=z_1=c_1} + \underbrace{\sum_{V_1 \cap S_2} y_{i,2}}_{=0} = c_1 \\ \sum_I x_i &= \underbrace{\sum_{I \cap S_1} y_{i,1}}_{=z_1=c_1} + \underbrace{\sum_{I \cap S_2} y_{i,2}}_{=z_2=1-c_1} = 1 \end{aligned}$$

Additionally, we also check the conditions for the single variables x_i for $i \in I$. For the upper and lower bound for any element $(y_1 + y_2) \in M$ the following is true

$$\begin{aligned} x_i = y_{i,1} + y_{i,2} &\leq \underbrace{y_{i,1}^{max} \mathbb{1}_{i \in S_1}}_{=0+z_1 \mathbb{1}_{|S_1|=1}=0+c_1 \mathbb{1}_{|V_1|=1}} + \underbrace{y_{i,2}^{max} \mathbb{1}_{i \in S_2}}_{=0} = 0 + c_1 \mathbb{1}_{|V_1|=1} && \text{for } i \in V_1 \\ x_i = y_{i,1} + y_{i,2} &\leq \underbrace{y_{i,1}^{max} \mathbb{1}_{i \in S_1}}_{=0} + \underbrace{y_{i,2}^{max} \mathbb{1}_{i \in S_2}}_{=z_2=1-c_1} = 1 - c_1 && \text{for } i \in I \setminus V_1 \\ x_i = y_{i,1} + y_{i,2} &\geq \underbrace{y_{i,1}^{min} \mathbb{1}_{i \in S_1}}_{=z_1=c_1} + \underbrace{y_{i,2}^{min} \mathbb{1}_{i \in S_2}}_{=0} = c_1 && \text{for } i \in V_1 \\ x_i = y_{i,1} + y_{i,2} &\geq \underbrace{y_{i,1}^{min} \mathbb{1}_{i \in S_1}}_{=0} + \underbrace{y_{i,2}^{min} \mathbb{1}_{i \in S_2}}_{=0+z_2 \mathbb{1}_{|S_2|=1}=0+(1-c_1) \mathbb{1}_{|(I \setminus V_1)|=1}} = 0 + (1 - c_1) \mathbb{1}_{|(I \setminus V_1)|=1} && \text{for } i \in I \setminus V_1 \end{aligned}$$

which proves the equivalence of the system of equations and inequalities describing P and M , i.e., $P = M$.

B Investment universe

Index	ISIN	Ticker	Name	Environmental Score Metric	NAICS Sector Name
0	-	CASH	CASH	-	-
1	US7475251036	QCOM	Qualcomm Inc.	67.47	Manufacturing
2	US0567521085	BIDU	Baidu Inc.	41.51	Other information services
3	US0530151036	ADP	Automatic Data Processing Inc.	43.25	Professional, Scientific, and Technical Services
4	US8552441094	SBUX	Starbucks Corporation	86.32	Accommodation and Food Services
5	US0311621009	AMGN	Amgen Inc.	76.28	Professional, Scientific, and Technical Services
6	US6937181088	PCAR	Paccar Inc.	83.90	Manufacturing
7	US0231351067	AMZN	Amazon.com Inc.	88.52	Retail Trade
8	US4612021034	INTU	Intuit Inc.	78.06	Information
9	US00724F1012	ADBE	Adobe Inc.	77.74	Information
10	US67066G1040	NVDA	NVIDIA Corporation	68.58	Manufacturing
11	US5949181045	MSFT	Microsoft Corporation	77.92	Information
12	US2786421030	EBAY	eBay Inc.	45.49	Retail Trade

Table 1: List of assets used in the environment.

F Simplex Decomposition for Portfolio Allocation Constraints in Reinforcement Learning

Venue 26th European Conference on Artificial Intelligence

DOI <https://doi.org/10.3233/FAIA230573>

Declaration of authorships The research idea was proposed by David Winkel and developed and discussed with Niklas Stauss and Matthias Schubert. David Winkel and Niklas Strauss wrote the manuscript, which was improved by all co-authors.

Publication

Simplex Decomposition for Portfolio Allocation Constraints in Reinforcement Learning

David Winkel^{a,b,*}, Niklas Strauß^{a,b}, Matthias Schubert^{a,b} and Thomas Seidl^{a,b}

^aLMU Munich, Germany

^bMunich Center for Machine Learning (MCML)

ORCID ID: David Winkel <https://orcid.org/0000-0001-8829-0863>,

Niklas Strauß <https://orcid.org/0000-0002-8083-7323>, Matthias Schubert <https://orcid.org/0000-0002-6566-6343>,

Thomas Seidl <https://orcid.org/0000-0002-4861-1412>

Abstract. Portfolio optimization tasks describe sequential decision problems in which the investor’s wealth is distributed across a set of assets. Allocation constraints are used to enforce minimal or maximal investments into particular subsets of assets to control for objectives such as limiting the portfolio’s exposure to a certain sector due to environmental concerns. Although methods for constrained Reinforcement Learning (CRL) can optimize policies while considering allocation constraints, it can be observed that these general methods yield suboptimal results. In this paper, we propose a novel approach to handle allocation constraints based on a decomposition of the constraint action space into a set of unconstrained allocation problems. In particular, we examine this approach for the case of two constraints. For example, an investor may wish to invest at least a certain percentage of the portfolio into green technologies while limiting the investment in the fossil energy sector. We show that the action space of the task is equivalent to the decomposed action space, and introduce a new reinforcement learning (RL) approach CAOSD, which is built on top of the decomposition. The experimental evaluation on real-world Nasdaq-100 data demonstrates that our approach consistently outperforms state-of-the-art CRL benchmarks for portfolio optimization.

1 Introduction

Portfolio optimization tasks belong to the family of resource allocation tasks in which an actor needs to allocate the available resources over a set of choices in each time step. Technically, resource allocation tasks can be considered multi-step decision problems with a standard-simplex action space describing all possible allocation choices, e.g., the set of all possible portfolio allocations in a financial setting. Policy gradient based RL can be used to optimize stochastic policies over the corresponding simplex action space and thus, they are often used to optimize portfolio allocation agents. For example, [20] proposes to use PPO [16] in combination with a Dirichlet action distribution for risk-aware portfolio optimization.

In many real-world financial settings, investors set maximum and minimum allocation weights to certain groups of assets for their portfolio. These constraints might originate from their client’s investment guidelines, restrictions posed by the regulator, or the investors’ eco-

nomical opinion. For example, an investor might need to consider sustainability aspects in addition to generating economic returns. In this setting, the investor might be required to invest a minimal amount of 30% of the portfolio into green technologies and a maximum amount of 15% into companies belonging to the fossil energy sector. Allocation constraints reduce the set of allowed actions for the agent within the simplex action space, constraining the action space to a *subset of the original simplex action space* that can be described geometrically as a convex polytope. Unfortunately, directly modeling a suitable action distribution on this polytope, which can be used to formulate a parametrizable policy function, is inherently difficult. A viable alternative approach is constrained Reinforcement Learning (CRL), which penalizes constraint violations in order to teach the agent to avoid disallowed actions, e.g., [12, 21, 18]. However, most of these approaches cannot *guarantee* that no constraint will be violated, may exhibit unstable training behavior, or produce suboptimal results, especially if more than a single allocation constraint is needed.

In this paper, we propose an alternative approach that decomposes the original constraint action space into unconstrained sub-action spaces, each containing a subset of the assets. The actions from these sub-action spaces are then combined back into the original action space using a weighted Minkowski sum. We exploit that any constraint requiring a maximum investment into a subset of assets is equivalent to requiring a minimum investment in the inverse subset of assets. This allows us to consider only constraints requiring a minimum allocation to asset groups. For the case of two allocation constraints, we decompose the action space into four sub-action spaces. The first sub-action space invests into the assets that are restricted by both allocation constraints. The second and third sub-action spaces ensure the fulfillment of each constraint after the allocations in the first sub-action space. The final sub-action space freely distributes the remaining funds which were not needed to fulfill the constraints. The allocation of assets in each sub-action space can be parameterized using an unconstrained Dirichlet distribution. We employ PPO [16] to optimize the policy function over the joint distribution of the sub-action spaces. Our new approach CAOSD ensures a tractable computation of the joint probability and gradients of the sub-action spaces through an auto-regressive architecture. Additionally, we use a transformer-based encoder of the current pricing structure of the market which is based on the recent price development.

We demonstrate the effectiveness of our novel approach in port-

* Corresponding Author. Email: winkel@dbs.ifi.lmu.de

folio optimization tasks based on real-world Nasdaq-100 index data. The results show that our approach is able to generate significantly higher returns than state-of-the-art constraint RL methods in the overwhelming majority of cases.

The main contributions of this paper are the following:

- We introduce a novel decomposition for constrained simplex action spaces with two allocation constraints into several unconstrained sub-action spaces.
- We propose a new method named CAOSD utilizing this decomposition to effectively apply standard RL algorithms like PPO on a surrogate action space.
- We demonstrate that CAOSD is able to significantly outperform state-of-the-art CRL benchmark approaches on real-world market data.

Our paper is structured as follows: In Section 2, we give an overview of the related work and continue in Section 3 with a formal problem definition. Afterward, we introduce our decomposition and show how to utilize the decomposition for RL in Section 4. We proceed with an extensive experimental evaluation of our approach in Section 5 before concluding the paper in Section 6.

2 Related Work

Portfolio optimization tasks with allocation constraints can be formulated as constrained Markov decision processes (CMDPs) [3] and policies can be optimized using CRL approaches [12, 21, 18]. Note that our novel decomposition is able to parameterize the constraint action space directly, and thus, the task can be formulated as an (unconstrained) Markov decision process (MDP), for which optimal policies can be found through standard RL algorithms.

CMDPs have seen successful applications in fields such as network traffic [11], and robotics [10, 2]. Finance applications such as [4, 20] use a scalarized objective function to maximize the return while penalizing for risk to perform a multi-objective, i.e. risk/return, portfolio optimization. The allocation constraints in our setting allow also to control for risk as they can be used to limit the investor’s exposure to risky sub-markets. [1] combine the risk/return optimization while also considering allocation constraints. They enforce these constraints by using a penalizing CRL approach stating that “there is no straightforward way to design an actor-network so that all proposed actions are compliant”. Our approach tackles this challenge and provides a way to do so on an actor-network level without the need for a penalty term.

There are different approaches to identify optimal policies in a CMDP setting. Penalty-based approaches include an additional penalty term into the objective function representing the constraints. An example is Lagrangian-based approaches such as [18, 5] that transform a constrained optimization problem into an unconstrained one by applying Lagrangian relaxation. A subsequent step solves a saddle point problem by optimizing the objective function and dynamically adjusting the penalty factor λ .

Alternative penalty-based approaches, such as [12], employ interior-point methods. Common drawbacks of penalty-based approaches are the need for additional hyperparameter tuning, expensive training loops, and the lack of guarantees for satisfying the constraints. An alternative approach is based on defining Trust Regions [2], which may produce constraint violations due to approximation errors. Furthermore, there are approaches based on prior knowledge [9] which pretrain a model to predict simple one-step dynamics of the environment. Other approaches are based on the use of Lyapunov

functions to solve CMDPs by projecting either the policy parameter or the action onto the set of feasible solutions induced by state-dependent linearized Lyapunov constraints [6, 7]. However, this approach can be computationally expensive and, in some cases, numerically intractable, especially as the action space grows larger [7].

Our method relies on a novel decomposition of the action space into sub-spaces. Thus, approaches *factorizing the action spaces* are another important research area that is related to this work. In [17], the authors introduce action branching, which divides the action space into independent sub-action spaces. In contrast, our problem involves modeling sub-action spaces that depend on each other. Auto-regressive approaches, which can model dependencies between sub-action spaces [13, 19, 15], are another common method for the factorization of the action spaces. Unlike these works, our approach focuses on a novel decomposition of a *constrained simplex action space* to make the optimization problem easier to solve using standard RL methods.

3 Problem Setting

An MDP is a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where \mathcal{S} represents the state space, \mathcal{A} the set of available actions, \mathcal{P} the transition function describing the distribution over future states s' given a state-action pair (s, a) , \mathcal{R} is a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ and γ is the discount factor.

For portfolio allocation tasks, \mathcal{A} is defined as a simplex over a set of N assets $I = \{0, \dots, N - 1\}$, i.e., $\mathcal{A} = \left\{ a \in \mathbb{R}_{0,+}^N : \sum_{i=0}^{N-1} a_i = 1 \right\}$. In other words, a_i describes the positive relative amount of capital assigned to the i^{th} asset.

An allocation constraint is defined by a subset $V \subseteq I$ of assets and a threshold value $c \in [0, 1]$ and implies that $\sum_{j \in V} a_j \geq c$ for all allocations $a \in \mathcal{A}$. Thus, at least the amount c of the available capital must be allocated to assets from the set V . Let us note that any *less-than* constraint can be rewritten into a greater-equal constraint and vice versa. In particular, assigning *at most* c into assets from V is equivalent to assigning *more than* $(1 - c)$ into the remaining assets $I \setminus V$, e.g., in a three asset setup with asset weights $x_1 + x_2 + x_3 = 1$ the greater-equal constraint $x_1 + x_2 \geq 0.3$ is equivalent to the constraint $x_3 < 0.7$. Thus, without the loss of generality, we will assume greater-equal constraints in the following. For portfolio allocation tasks with one or more allocation constraints, the action space is a convex polytope within the original simplex of all N -dimensional allocation vectors.

The goal of a constrained portfolio allocation task is to find a policy π_θ maximizing the expected reward $\mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t \cdot r_{t+1} \right]$ where r_{t+1} is the direct reward observed for the t^{th} state transition of episodes τ sampled by π_θ while only using allowed allocations. In our setting, the reward corresponds to the direct economic returns of the entire portfolio. Finding a suitable formulation for π_θ becomes more and more complex with an increasing number of allocation constraints. In the following, we will formulate π_θ for up to two allocation constraints which can directly be used in combination with standard actor-critic and policy gradient RL algorithms. Thus, we consider the following action space with $V_1 \subseteq I$ and $V_2 \subseteq I$:

$$\mathcal{A}_{2C} = \left\{ a \in \mathbb{R}_{0,+}^N : \sum_{i \in I} a_i = 1, \sum_{j \in V_1} a_j \geq c_1, \sum_{k \in V_2} a_k \geq c_2, 0 \leq c_1 \leq 1, 0 \leq c_2 \leq 1 \right\}$$

4 Constrained Allocation Optimization with Simplex Decomposition (CAOSD)

As mentioned in Section 3, the action space underlying our problem is a convex polytope within the standard simplex over a universe of n assets. Thus, directly defining a parameterizable probability distribution that could be used in a policy function for reinforcement learning is rather complex. To avoid this complexity, we propose to decompose the action space into four standard simplices over subsets of I . For a proper weighting, allocations taken from these four standard simplices add up to form a complete allocation action in the original action space. In the following, we will describe the simplices and how to compute weights, guaranteeing that the original and the decomposed action set are equivalent. Afterward, we will describe how a policy function can be defined on top of the decomposed action space and how policy gradient based reinforcement learning methods can be applied.

4.1 Action Space Simplex Decomposition

To formalize our approach, we begin by defining the basic elements of our decomposition, i.e., the padded standard simplices and their combination with the weighted Minkowski sum.

Definition 1. Let $I = \{0, \dots, N-1\}$ be a set of indices referring to respective dimensions in \mathbb{R}^N . Let S_K be a standard simplex in the subspace defined over the dimensions indicated by the index set $K \subseteq I$. Let $g_K : \mathbb{R}^{|K|} \rightarrow \mathbb{R}^N$ be a function that projects S_K into \mathbb{R}^N , by padding the entries for any elements in \mathbb{R}^N in those dimensions with indices $I \setminus K$ with 0. Applying the function g on S_K then yields a **padded standard simplex (PSS)** defined as:

$$PSS_K = g_V(S_K) = \left\{ y \in \mathbb{R}_{0,+}^N : \sum_{j \in K} y_j = 1; y_i \geq 0 \forall i \in K; \right. \\ \left. y_j = 0 \quad \forall j \in I \setminus K \right\} \quad \text{for } K \neq \emptyset$$

and

$$PSS_K = g_V(S_K) = \left\{ y \in \mathbb{R}_{0,+}^N : y_j = 0 \quad \forall j \in I \right\} \quad \text{for } K = \emptyset$$

Definition 2. Given n sets of vectors Q_1, \dots, Q_n in \mathbb{R}^N , the **weighted Minkowski sum** of Q_1, \dots, Q_n is generated by adding each combination of vectors from sets Q_i after applying a respective weighting factor z_i , i.e., $(Q_i)_{z_i} = \{z_i \cdot q_i | q_i \in Q_i\}$ with $i = \{1, \dots, n\}$. We write the weighted Minkowski sum of the sets as $M_z = (Q_1)_{z_1} + \dots + (Q_n)_{z_n} = \{z_1 \cdot q_1 + \dots + z_n \cdot q_n | q_i \in Q_1, \dots, q_n \in Q_n\}$. We refer to $(Q_i)_{z_i}$ for $i = \{1, \dots, n\}$ as the **weighted Minkowski summands**.

Our approach identifies four PSSs and a weighting vector $z = [z_1, z_2, z_3, z_4]$ that can be combined as a weighted Minkowski sum $M_z = (PSS_1)_{z_1} + (PSS_2)_{z_2} + (PSS_3)_{z_3} + (PSS_4)_{z_4}$ such that $M_z = \mathcal{A}_{2C}$.

The first weighted Minkowski summand PSS_{K_1} is built over the intersection of assets $K_1 = V_1 \cap V_2$. Investing into these assets contributes to fulfilling both constraints. In fact, if $c_1 + c_2 > 1$, we need to invest at least a portion of $z_1 = c_1 + c_2 - 1$ into assets from $V_1 \cap V_2$ to avoid over investment.

The second weighted Minkowski summand PSS_{K_2} is defined over the assets in $V_1 = K_2$. To fulfill the first constraint, we have to make sure that we at least invest c_1 into assets in V_1 . However,

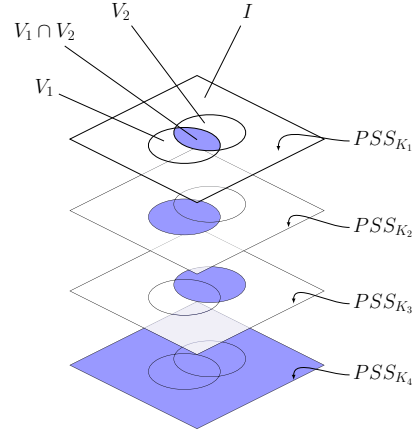


Figure 1: Set of padded variables represented as white area, set of modeled variables represented as colored area for each of the four PSSs.

we need to consider that any capital z_1 already being allocated into PSS_{K_1} also contributes towards fulfilling this constraint. Correspondingly, PSS_{K_3} is defined over the assets in $K_3 = V_2$ and requires an investment of c_2 minus any allocation made to $V_1 \cap V_2$ from PSS_{K_1} and PSS_{K_2} . Finally, PSS_{K_4} is defined over the complete asset universe I . It covers the case, that not any available capital is needed to fulfill the allocation constraints. Thus, any remaining capital $1 - (z_1 + z_2 + z_3)$ can be freely allocated among the assets in I to maximize the economic return. An illustration of the four sets being covered by these weighted Minkowski summands can be found in Figure 1.

To demonstrate the basic principle, consider an allocation task in which our capital must be allocated over five assets a_1, a_2, a_3, a_4 and a_5 . The first constraint c_1 requires to allocate at least 30% weight to the group of variables with index $V_1 = \{1, 3\}$. The second constraint c_2 requires to allocate at least 50% weight to the group of variables with index $V_2 = \{2, 4\}$. Thus, the set of feasible solutions, i.e. the action space, is defined by the polytope

$$P_0 = \left\{ a \in \mathbb{R}^5 : \sum_{i \in I} a_i = 1; \sum_{i \in V_1} a_i \geq 0.3; \sum_{i \in V_2} a_i \geq 0.5; \right. \\ \left. x_i \geq 0 \quad \forall i \in I = \{1, 2, 3, 4, 5\} \right\}$$

Given the four PSS_{K_j} with the respective index sets of $K_1 = \emptyset$, $K_2 = V_1$, $K_3 = V_2$, $K_4 = I$ and the weighting vector $z = [z_1, z_2, z_3, z_4] = [0.0, 0.3, 0.5, 0.2]$. The corresponding weighted Minkowski sum M will equal P_0 as shown in the following: Any final allocation $a = [a_1, a_2, a_3, a_4, a_5] \in M$ is the vector sum of four vectors $\tilde{a}_j = [\tilde{a}_{1,j}, \tilde{a}_{2,j}, \tilde{a}_{3,j}, \tilde{a}_{4,j}, \tilde{a}_{5,j}] \in (PSS_{K_j})_{z_j} \subset \mathbb{R}^5$ for $j \in \{1, 2, 3, 4\}$, i.e. $a = \tilde{a}_1 + \tilde{a}_2 + \tilde{a}_3 + \tilde{a}_4$. The first sub-weighting vector \tilde{a}_1 will be $(0, 0, 0, 0, 0)$ due to $K_1 = \emptyset$ as we cannot invest into any assets. Any sub-weighting vector $\tilde{a}_2 \in (PSS_{K_2})_{z_2}$ will allocate a total of $z_2 = 0.3$ weight to the variables with an index in V_1 , ensuring that any $a \in M$ will always satisfy the *lower bound* of the first constraint $c_1 = 0.3$. Any vector $\tilde{a}_3 \in (PSS_{K_3})_{z_3}$ will allocate a total of $z_3 = 0.5$ weight to the variables with an index in V_2 , ensuring that any $a \in M$ will always satisfy the *lower bound*

of the second constraint $c_2 = 0.5$. Any vector $\tilde{a}_4 \in (PSS_{K_4})_{z_4}$ will allocate the remainder of $z_4 = 0.2$ weight to any combination of variables in I , (a) ensuring that $\sum_{i \in I} a_i = 1$ and (b) potentially allocating additional weight to the variables with indices in V_1 and V_2 (since $V_1 \subseteq I$ and $V_2 \subseteq I$), allowing $a \in M$ to exceed the lower bounds of c_1 and c_2 , i.e. $\sum_{i \in V_1} a_i \geq 0.3$ and $\sum_{i \in V_2} a_i \geq 0.5$. As a result, the sets M and P_0 have an identical H-representation (see Definition 3), i.e. being specified by the identical sets of constraints, making them identical polytopes.

In the following, we will introduce a weighting scheme selecting z which guarantees general equivalence between M and P_0 . First, we formalize our constrained action space A_{2C} as convex polytope and introduce its H-representation.

Definition 3. A convex polytope P in \mathbb{R}^n is defined as a polytope that additionally is also a convex set. P can be viewed as the set of solutions to a system of linear inequalities, i.e., the intersection of a finite number of closed half-spaces, called P 's half-space representation (H-representation):

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + \cdots + & a_{1n}x_n & \geq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + \cdots + & a_{2n}x_n & \geq & b_2 \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + \cdots + & a_{mn}x_n & \geq & b_m \end{array}$$

The general formulation on how to identify the four PSS_{K_j} with their respective index sets K_j and the definition of the weighting vector z can be found in Theorem 1. Let us note that we define the weighting vector z as an autoregressive function with $z = [z_1, z_2(z_1), z_3(z_1, z_2, y_2), z_4(z_1, z_2, y_2, z_3)]$ which results in an adaptive weighting vector depending on each current combination of elements in PSS_{K_j} for all $j = \{1, 2, 3, 4\}$.

Theorem 1. Any polytope P defined by the system

$$\sum_{i \in I} x_i = 1; \quad x_i \geq 0 \quad \forall i \in I; \quad \sum_{i \in V_1} x_i \geq c_1; \quad \sum_{i \in V_2} x_i \geq c_2$$

with $I = \{0, \dots, N-1\}$, $V_1 \subseteq I$ and $V_2 \subseteq I$ can be expressed as a weighted Minkowski sum with the four weighted Minkowski summands with $y_{i,j} = [y_{0,j}, \dots, y_{N-1,j}] \in (PSS_{K_j})_{z_j}$:

$$(PSS_{K_1})_{z_1} : K_1 = V_1 \cap V_2 \text{ and } z_1 = \max(0, c_1 + c_2 - 1)$$

$$(PSS_{K_2})_{z_2} : K_2 = V_1 \text{ and } z_2 = \max(0, c_1 - z_1)$$

$$(PSS_{K_3})_{z_3} : K_3 = V_2 \text{ and } z_3 = \max(0, c_2 - z_1 - z_{2,\cap})$$

$$\text{where } z_{2,\cap} = \sum_{i \in V_1 \cap V_2} y_{i,2}$$

$$(PSS_{K_4})_{z_4} : K_4 = I \text{ and } z_4 = 1 - z_1 - z_2 - z_3$$

Proof. Showing that two convex polytopes have an equivalent H-representation, i.e. an equivalent system of linear inequalities describing them, proves that they are identical.

When calculating the weighted Minkowski sum M of the four PSSs, we can deduce that for any element $x = (y_1 + y_2 + y_3 + y_4) \in M$ with $y_1 \in (PSS_{K_1})_{z_1}$, $y_2 \in (PSS_{K_2})_{z_2}$, $y_3 \in (PSS_{K_3})_{z_3}$ and $y_4 \in (PSS_{K_4})_{z_4}$ the following constraints are fulfilled:

The contribution to the variables $\sum_{V_1} x_i$ by $(PSS_{K_1})_{z_1}$ and $(PSS_{K_2})_{z_2}$ will always be $\max(0, c_1 + c_2 - 1) + \max(0, c_1 - z_1) =$

c_1 while $(PSS_{K_3})_{z_3}$ and $(PSS_{K_4})_{z_4}$ can optionally contribute positive weight, resulting in

$$\begin{aligned} \sum_{i \in V_1} x_i &= \underbrace{\sum_{i \in V_1 \cap K_1} y_{i,1}}_{=z_1 = \max(0, c_1 + c_2 - 1)} + \underbrace{\sum_{i \in V_1 \cap K_2} y_{i,2}}_{=z_2 = \max(0, c_1 - z_1)} + \underbrace{\sum_{i \in V_1 \cap K_3} y_{i,3}}_{\geq 0} \\ &+ \underbrace{\sum_{i \in V_1 \cap K_4} y_{i,4}}_{\geq 0} \geq c_1. \end{aligned}$$

The contribution to the variables $\sum_{V_2} x_i$ by $(PSS_{K_1})_{z_1}$, $(PSS_{K_2})_{z_2}$, $(PSS_{K_3})_{z_3}$ will always be $z_1 + z_{2,\cap} + \max(0, c_2 - z_1 - z_{2,\cap}) = c_2$ while $(PSS_{K_4})_{z_4}$ can optionally contribute positive weight, resulting in

$$\begin{aligned} \sum_{i \in V_2} x_i &= \underbrace{\sum_{i \in V_2 \cap K_1} y_{i,1}}_{=z_1} + \underbrace{\sum_{i \in V_2 \cap K_2} y_{i,2}}_{=z_{2,\cap}} + \underbrace{\sum_{i \in V_2 \cap K_3} y_{i,3}}_{=\max(0, c_2 - z_1 - z_{2,\cap})} \\ &+ \underbrace{\sum_{i \in V_2 \cap K_4} y_{i,4}}_{\geq 0} \geq c_2. \end{aligned}$$

The total weight contribution from all four $(PSS_{K_j})_{z_j}$ for $j = \{1, 2, 3, 4\}$ to all variables $\sum_I x_i$ will be always $z_1 + z_2 + z_3 + (1 - z_1 - z_2 - z_3) = 1$, i.e.

$$\begin{aligned} \sum_{i \in I} x_i &= \underbrace{\sum_{i \in I \cap K_1} y_{i,1}}_{=z_1} + \underbrace{\sum_{i \in I \cap K_2} y_{i,2}}_{=z_2} + \underbrace{\sum_{i \in I \cap K_3} y_{i,3}}_{=z_3} \\ &+ \underbrace{\sum_{i \in I \cap K_4} y_{i,4}}_{=z_4 = 1 - z_1 - z_2 - z_3} = 1. \end{aligned}$$

Additionally, we check the constraints for the single variables x_i for $i \in I$. Since for $y_j = [y_{0,j}, \dots, y_{N-1,j}]$ with $j = \{1, 2, 3, 4\}$ all single variables $y_{i,j}$ with $i \in I$ are defined to be greater equal than zero, it follows that

$$x_i = \underbrace{y_{i,1}}_{\geq 0} + \underbrace{y_{i,2}}_{\geq 0} + \underbrace{y_{i,3}}_{\geq 0} + \underbrace{y_{i,4}}_{\geq 0} \geq 0 \quad \forall i \in I$$

which shows the equivalence of the two sets of convex closed half-spaces defining P and M , which proves that $P = M$. \square

4.2 Task optimization via Reinforcement Learning

After describing the simplex decomposition of the action space, we will now define a stochastic policy function based on our novel decomposition which can be used for policy optimization with policy gradient based RL approaches.

We optimize the policy on a surrogate action space $\tilde{\mathcal{A}} = \tilde{\mathcal{A}}_1 \times \tilde{\mathcal{A}}_2 \times \tilde{\mathcal{A}}_3 \times \tilde{\mathcal{A}}_4$, which is the Cartesian product of the four sub-action spaces. These sub-action spaces correspond to the four PSS_{K_j} when decomposing \mathcal{A}_{2C} as introduced in the previous section. An action $\tilde{a} \in \tilde{\mathcal{A}}$ can be mapped into the original action space by using the weighted asset-wise sum over the sub-action spaces: $a = z_1 \cdot \tilde{a}_1 + z_2 \cdot \tilde{a}_2 + z_3 \cdot \tilde{a}_3 + z_4 \cdot \tilde{a}_4$. Note that each surrogate action \tilde{a} maps to one particular action a in the original action space \mathcal{A} but not vice versa. In other words, the mapping is surjective but not bijective. Based on this property, we can show that any optimal policy on the surrogate action set $\tilde{\mathcal{A}}$ is optimal on the original action space \mathcal{A} as well.

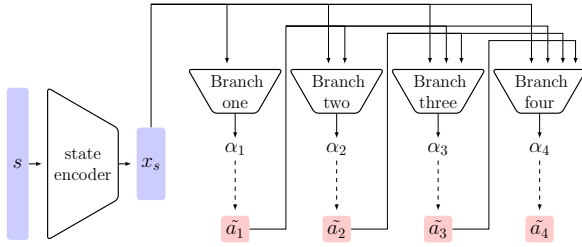


Figure 2: Auto-regressive architecture. The dashed arrows represent the process of sampling from a Dirichlet distribution to generate a sub-action \tilde{a}_j .

Theorem 2. Given the constraint allocation task $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ as defined in Section 3 with the surrogate action space $\tilde{\mathcal{A}}$ corresponding to the action decomposition defined in Theorem 1. Any optimal policy over the surrogate action space $\tilde{\mathcal{A}}$, $\pi_{\tilde{\mathcal{A}}}^*$ with $Q(s, \pi_{\tilde{\mathcal{A}}}^*(s)) \geq Q(s, \pi_{\mathcal{A}}(s))$ for any state $s \in \mathcal{S}$ and any policy $\pi_{\mathcal{A}}(s)$, implies an optimal policy $\pi_{\mathcal{A}}^*$ on the original action space \mathcal{A} .

Proof. We know from Theorem 1 that there exist weights z which allow to represent any $a \in \mathcal{A}$ by at least one surrogate action \tilde{a} . In addition, we know that for any state $s \in \mathcal{S}$ and any action $a \in \mathcal{A}$, the state-action value function $Q(s, a)$ is the same for any \tilde{a} mapping to a as the reward received for performing \tilde{a} is provided by the environment via performing the joint action a .

Thus, for any optimal policy $\pi_{\tilde{\mathcal{A}}}^*$, there exists a corresponding policy $\pi_{\mathcal{A}}^*$ providing the same Q-values. Now assume that $\pi_{\tilde{\mathcal{A}}}^*$ is not optimal and thus, there would be a policy $\hat{\pi}_{\tilde{\mathcal{A}}}$ with $Q(s, \hat{\pi}_{\tilde{\mathcal{A}}}(s)) > Q(s, \pi_{\tilde{\mathcal{A}}}^*(s))$ for at least one state $s \in \mathcal{S}$. Since the mapping between the $\tilde{\mathcal{A}}$ and \mathcal{A} is surjective, there must exist a decomposition of $\hat{\pi}_{\tilde{\mathcal{A}}}(s)$ to the surrogate action space yielding higher Q-values than $\pi_{\tilde{\mathcal{A}}}^*(s)$ which contradicts the optimality of $\pi_{\tilde{\mathcal{A}}}^*$. \square

Theorem 2 shows that any well-performing policy in $\tilde{\mathcal{A}}$ can be mapped to an equally well-performing policy over \mathcal{A} .

After introducing the surrogate action space, we will introduce our stochastic policy function over $\tilde{\mathcal{A}}$. We model each sub-action space with a Dirichlet distribution with a parameter vector α_{K_j} , which is obtained by a neural network. Our policy function that is based on the decomposition described in Theorem 1, requires an iterative computation of the weighting vector z . Our algorithm for sampling an action is detailed in Algorithm 1. In each step, we sample the asset allocation for PSS_{K_j} by the corresponding Dirichlet distribution Dir_j and then determine the corresponding weight z_j . In the end, the weighted asset-wise sum is computed and returned as joint action which is applied to the environment.

An overview of our architecture is depicted in Figure 2. We first create a representation x_s of the observation s using a transformer model. Each sub-action space is parameterized by an MLP using the representation x_s as well as all sampled surrogate actions from the previous sub-action spaces. This structure allows a tractable computation and optimization of the joint surrogate action \tilde{a} probability: $P(\tilde{a}_1, \tilde{a}_2, \tilde{a}_3, \tilde{a}_4 | x_s) = P(\tilde{a}_1 | x_s) \cdot P(\tilde{a}_2 | \tilde{a}_1, x_s) \cdot P(\tilde{a}_3 | \tilde{a}_2, \tilde{a}_1, x_s) \cdot P(\tilde{a}_4 | \tilde{a}_3, \tilde{a}_2, \tilde{a}_1, x_s)$.

We employ a policy gradient approach based on the PPO algorithm introduced by [16]. Note that our method can also be used with other policy gradient based RL methods.

The state encoder is composed of three fully connected layers of size 512, 256, and 128 with ReLU activation functions that feed into

Algorithm 1 Action Generation using the Simplex Decomposition

Input: Index set of all N assets in the investable universe $I = \{0, 1, \dots, N - 1\}$; Two allocation constraints $C_1 : \sum_{i \in V_1} x_i \geq c_1$ and $C_2 : \sum_{i \in V_2} x_i \geq c_2$

Define:

$K_1 = V_1 \cap V_2$, $K_2 = V_1$, $K_3 = V_2$ and $K_4 = I$, f_j is an autoregressive policy network branch for $j = \{1, 2, 3, 4\}$

Begin action generation:

- 1: calculate α_1 from $f_1(x_s)$, sample \tilde{a}_1 from $Dir(\alpha_1)$
- 2: set $z_1 = \max(0, c_1 + c_2 - 1)$
- 3: set $q_{V_1 \cap V_2} := z_1 \sum_{i \in V_1 \cap V_2} x_{i,1}$
- 4: calculate α_2 from $f_2(x_s, \tilde{a}_1)$, sample \tilde{a}_2 from $Dir(\alpha_2)$
- 5: set $z_2 = \max[0, c_1 - z_1]$
- 6: update $q_{V_1 \cap V_2} := q_{V_1 \cap V_2} + z_2 \sum_{i \in V_1 \cap V_2} x_{i,2}$
- 7: calculate α_3 from $f_3(x_s, \tilde{a}_1, \tilde{a}_2)$, sample \tilde{a}_3 from $Dir(\alpha_3)$
- 8: set $z_3 = \max[0, c_2 - q_{V_1 \cap V_2}]$
- 9: calculate α_4 from $f_4(x_s, \tilde{a}_1, \tilde{a}_2, \tilde{a}_3)$, sample \tilde{a}_4 from $Dir(\alpha_4)$
- 10: set $z_4 = 1 - z_1 - z_2 - z_3$
- 11: calculate action a by adding the weighted sub-actions:

$$z_1 \cdot \tilde{a}_1 + z_2 \cdot \tilde{a}_2 + z_3 \cdot \tilde{a}_3 + z_4 \cdot \tilde{a}_4 = a$$

a GTrXL element, allowing also to handle tasks that require memory. GTrXL is based on [14] and is specifically designed to utilize transformers in an RL setting. The GTrXL element is composed of a single *transformer unit* with a single encoder layer and a single decoder layer with four attention heads and an embedding size of 64. The different branching modules are all made up of two fully connected layers of size 64 and 32, respectively, with a ReLU activation function after the first layer and a softplus activation function for the final output layer.

5 Experiments

5.1 Constrained Portfolio Optimization Tasks

We evaluate our approach in the financial setting on various constrained portfolio optimization tasks. The environment is based on [20] and uses real-world data from the Nasdaq-100 index that has been processed by the qlib package.¹ The data is used to estimate the parameters of a hidden Markov model (HMM), which is then used to generate trajectories. The monthly closing stock prices from January 1, 2010 to December 31, 2020 are included in the data set. An additional data set containing monthly closing prices from January 1, 2021 to December 31, 2021 is exclusively used to backtest the approaches. The environment's investment universe consists of 12 assets plus the special asset cash. Cash has neither a positive or negative return and remains stable over time. The remaining 12 assets are chosen at random from a pool of 35 pre-selected assets from the Nasdaq-100 data set. The assets were pre-selected based on the fact that they have been a member of the index at least since January 1, 2010, and there were no missing data entries.

In the following, we provide a detailed description of the portfolio optimization task. An agent is required to invest his wealth into N different assets based on asset allocation decisions made at each time step of the investment horizon T . The **constrained action space** for this task is described by \mathcal{A}_{2C} as defined in Section 3, where the sets V_1 and V_2 contain the indices of assets affected by the respective constraint. These allocation constraints in the financial setting can be motivated by various factors, such as the need to invest *at least* a

¹ <https://github.com/microsoft/qlib/tree/main>

certain percentage of the portfolio into a group of assets in a specific sector or with a certain risk classification.

The **observation space** is defined as $\mathcal{O} = \mathcal{W} \times \mathcal{V} \times \mathcal{U}$. The set $\mathcal{W} \subseteq \mathbb{R}$ represents the agent’s current absolute level of wealth, the set $\mathcal{V} \subseteq \mathbb{R}^N$ describes the current portfolio allocation, and the set $\mathcal{U} \subseteq \mathbb{R}^N$ is the observed single asset economic returns from the previous time step.

The total portfolio return $r = \vartheta_{PF} - tc$ is the agent’s **reward** for each time step and is defined as the realized portfolio’s return ϑ_{PF} minus any transaction costs tc that occurred. The portfolio’s return is a random variable $\Theta_{PF} = a^T \Theta$ based on the random vector $\Theta = [\Theta_0, \dots, \Theta_{N-1}]$ representing the economic returns of the single assets and the deterministic vector a that represents the portfolio’s allocation weights selected by the agent. The cumulative portfolio total return over the investment horizon of 12 months, i.e. $T = 12$ time steps, is defined as

$$\nu = \sum_{t=0}^{T-1} r_{t+1}$$

and in the following referred to as the *annualized total portfolio return*.

5.2 Experimental Setup

As previously stated, two evaluation environments are used: (1) the *simulation environment* and (2) the *backtesting environment*. We conduct a total of 100 experiments, each with a unique constraint configuration, i.e., a unique combination of two allocation constraints. Each constraint configuration is evaluated on both evaluation environments with the goal of comparing the performance of the approaches for different constraint configurations (a) on the environment the agents were originally trained on and (b) on *unseen, real world* data.

Each experiment uses a different random seed as well as a randomly generated constraint configuration. A constraint configuration is made up of two allocation constraints C_j of the form $\sum_{i \in V_j} a_i \geq c_j$ with $j \in \{1, 2\}$ where V_j represents the set of affected assets and c_j the constraint’s threshold value. To generate both allocation constraints at random, we sample the number of affected assets between 1 and 12. We rule out the possibility of selecting 0 or 13 assets since any greater equal allocation constraint would be either infeasible or trivial. The sampled number defines the number of specific assets which are then sampled from the list of 13 assets, i.e. the investment universe, without replacement resulting in V_j . Subsequently c_j is sampled from a uniform distribution in the interval $[0, 1]$. The process is repeated for the second allocation constraint as well resulting in a randomly generated constraint configuration. In a final step it is verified that the resulting polytope P as defined in Section 3 is not an empty set, i.e. a system that does not have a feasible solution.²

We compare our CAOSD approach to four other approaches, one of which is a naive random approach and three of which are state-of-the-art CRL approaches. The CRL approaches typically model constraint violations on a trajectory level, which means that they constrain the expected discounted sum of costs that occurred in each time step [3]. However, they can be adjusted to model allocation constraints that must be satisfied at each time step. This can be done by defining the costs at each time step in such a way that they return a

positive value if an allocation constraint is violated and zero otherwise. When a violation occurs in any time step, the discounted sum of costs will be greater than zero. Therefore, we can constrain every time step in the trajectory implicitly by imposing a constraint on the trajectory level that the expected discounted sum of costs needs to be less than or equal to zero.

The first CRL benchmark approach is the Lagrangian-based RCPO introduced by [18]. The second benchmark approach is IPO proposed by [12] that uses an interior-point method to optimize the policy. The third approach is P3O by [21], a first-order optimization approach that uses an unconstrained objective in combination with a penalty term equaling the original constraint objective. The benchmark approaches are implemented in the RLlib framework based on their papers.³ The code for all approaches is made publicly available.⁴ All agents were trained on a cluster using various types of commercially available single GPUs. All approaches were extensively tuned in terms of hyperparameters using a grid search. Additional information on the hyperparameter tuning process can be found in the Appendix. During evaluation, RL agents take the action with the highest likelihood.

In addition to the three benchmark approaches we also utilize a random approach. This approach uniformly draws actions, i.e. asset allocations, from the constrained polytope. Efficient uniform sampling from a polytope is a surprisingly complex task, therefore we follow [8] to obtain uniform samples from the constrained action space. Using several rollouts of this baselines allows us to establish an estimate of the difficulty of an experiment, since the possible returns are highly dependent on the allocation constraints.

5.3 Evaluation

In our evaluation, we first compare the performance of our approach and the benchmarks over the entire set of experiments, which demonstrates the effectiveness of our approach in various settings. Afterward, we discuss the performance and convergence during training and take a detailed look at a single experiment. A key metric of the evaluation is the agents’ mean annualized total portfolio returns. We define the mean annualized total portfolio return for each of the five approaches, i.e. $app = \{RCPO, IPO, P3O, CAOSD, RDM\}$, and each of the environments, i.e. $env = \{sim, bt\}$, as $\bar{\nu}_{app}^{env} = \frac{1}{J} \sum_{j=0}^{J-1} \nu_{app,j}^{env}$ where J is the number of evaluation trajectories. For the simulation environment $J = 1000$ trajectories per approach are evaluated after the agents’ training is completed.

In backtesting – with the exception of the random approach – only the single real-world trajectory is evaluated to measure the agents’ performance since their evaluation is deterministic. The random approach is treated differently since its evaluation remains stochastic due to its previously mentioned design to always sample uniformly an action from P . To reduce the variance in the results, we evaluate $\bar{\nu}_{RDM}^{bt}$ on $J = 1000$ rollouts during backtesting.

We use two measures to evaluate the performance of the approaches over all experiments. The first measure, $\bar{\theta}_{app}^{env}$, is the average of the mean annualized return of each approach over all experiments. More formally, $\bar{\theta}_{app}^{env} = \frac{1}{N} \sum_{i=0}^{N-1} \bar{\nu}_{app,i}^{env}$, where $N = 100$ is the number of experiments. Since the return that can be achieved in each experiment varies greatly depending on the constraint configuration, our second measure is defined as the difference of returns

² This can be checked by determining whether the V-representation of P contains at least one vertex.

³ <https://docs.ray.io/en/master/rllib/index.html>

⁴ <https://github.com/DavWinkel/SimplexDecompositionECAI>

	$\bar{\theta}_{app}^{env}$	Upper 95% CI	Lower 95% CI
Simulation			
RCPO	0.292	0.299	0.284
IPO	0.212	0.217	0.207
P3O	0.305	0.314	0.296
CAOSD	0.327	0.335	0.319
Random	0.209	0.217	0.201
Backtesting			
RCPO	0.497	0.521	0.474
IPO	0.35	0.365	0.334
P3O	0.522	0.552	0.492
CAOSD	0.552	0.582	0.522
Random	0.333	0.355	0.311

Table 1: Evaluation of $\bar{\theta}_{app}^{env}$ and its 95% confidence interval for all approaches in both environments for N=100 experiments after training is completed.

of each approach to the performance of the random baseline in the same experiment: $\bar{\delta}_{app}^{env} = \frac{1}{N} \sum_{i=0}^{N-1} \bar{v}_{app,i}^{env} - \bar{v}_{RDM,i}^{env}$.

	$\bar{\delta}_{app}^{env}$	Upper 95% CI	Lower 95% CI
Simulation			
RCPO	0.082	0.093	0.071
IPO	0.003	0.012	-0.007
P3O	0.096	0.108	0.084
CAOSD	0.118	0.129	0.106
Backtesting			
RCPO	0.164	0.196	0.132
IPO	0.017	0.046	-0.012
P3O	0.189	0.227	0.151
CAOSD	0.219	0.258	0.179

Table 2: Evaluation of $\bar{\delta}_{app}^{env}$ and its 95% confidence interval for the non-random approaches in both environments for N=100 experiments after training is completed.

Table 1 and Table 2 show the performance of the approaches for both metrics in both environments as well as their corresponding 95% confidence intervals. The CAOSD approach shows considerable improvements over the other approaches in both metrics and both environments. These improvements are statistically significant on a 95% confidence interval. The P3O approach ranks second in both environments for both metrics before RCPO. IPO is only able to outperform the random approach in the backtesting environment while producing similar performance results to the random approach in the simulation environment.

In the second part of the evaluation, we will discuss the performance of the agents *during training* on a representative experiment. The experiment has a constraint configuration with the two allocation constraints $C_1 : \sum_{i \in V_1} a_i \geq 0.23$ with V_1 containing the indices referring to the company stocks [BIDU, QCOM] and $C_2 : \sum_{i \in V_2} a_i \geq 0.32$ with V_2 referring to the indices of the companies [ADBE, SBUX, QCOM] (see Appendix for a detailed list of the environment’s investment universe). During training, an evaluation with $J = 200$ trajectories is performed every 80000 environment steps.

Figure 3 shows the agents’ mean annualized total portfolio return during training on the y-axis and the number of environment steps on the x-axis. The figure also shows the 95% confidence interval of the mean annualized total portfolio return for each of the ap-

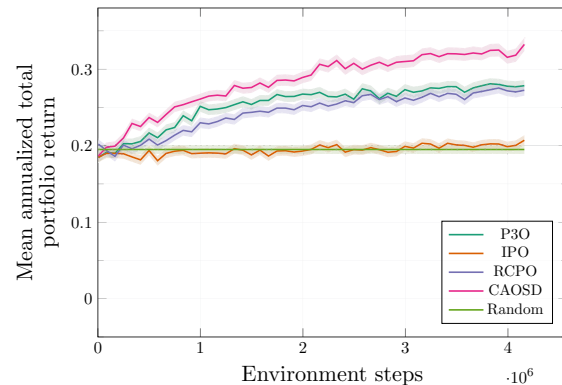


Figure 3: Mean annualized total portfolio return during training and its 95% confidence interval. This figure is best viewed in color.

proaches seen as the shaded areas around the lines. Note that we only show the *training performance* for the simulation environment, since there is no training in the backtesting environment. The CAOSD approach had the best training performance in the experiment shown in Figure 3, followed by P3O and RCPO. The IPO approach is not able to improve the mean annualized total portfolio return during training and stays comparable to the random approach.

6 Conclusion

In this paper, we examine portfolio optimization tasks with allocation constraints that require investing at least a certain portion of the available capital into a subset of assets. The task covers many real-world use-cases such as investors wanting to limit their exposure to certain groups of assets due to risk concerns, or investors who want to reflect aspects such as sustainability or social responsibility in their portfolio allocation. We examine settings that consider two allocation constraints and present CAOSD which decomposes the constrained action space into multiple unconstrained sub-action spaces. We show that the weighted Minkowski sum of these sub-action spaces is equivalent to the original action space if weights are chosen properly. Based on the decomposition we introduce a stochastic policy function that computes proper weights with an autoregressive pattern. To optimize the policy for a given task, we apply a transformer-based state encoder and employ PPO [16] to train our agent. In the experimental part, we apply our approach to a variety of constrained portfolio optimization tasks, each characterized by a different set of constraints. We significantly outperform state-of-the-art approaches from CRL on real-world market data which demonstrates the effectiveness of our proposed method.

While this work shows decomposition with up to two allocation constraints, we will investigate decompositions for a greater number of constraints in future work. This increases the complexity of the possible relationship structures between the sets of choices, necessitating increasingly complex decompositions. In another line of future work, we want to examine the application of our approach to other tasks than portfolio optimization.

References

- [1] Carlo Abrate, Alessio Angius, Gianmarco De Francisci Morales, Stefano Cozzini, Francesca Iadanza, Laura Li Puma, Simone Pavanelli,

- Alan Perotti, Stefano Pignataro, and Silvia Ronchiadin, ‘Continuous-action reinforcement learning for portfolio allocation of a life insurance company’, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 237–252. Springer, (2021).
- [2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel, ‘Constrained policy optimization’, in *International conference on machine learning*, pp. 22–31. PMLR, (2017).
- [3] Eitan Altman, *Constrained Markov decision processes: stochastic modeling*, Routledge, 1999.
- [4] Eric André and Guillaume Coqueret, ‘Dirichlet policies for reinforced factor portfolios’, *arXiv preprint arXiv:2011.05381*, (2020).
- [5] Shalabh Bhatnagar and K Lakshmanan, ‘An online actor–critic algorithm with function approximation for constrained markov decision processes’, *Journal of Optimization Theory and Applications*, **153**(3), 688–708, (2012).
- [6] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh, ‘A lyapunov-based approach to safe reinforcement learning’, *Advances in neural information processing systems*, **31**, (2018).
- [7] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh, ‘Lyapunov-based safe policy optimization for continuous control’, *arXiv preprint arXiv:1901.10031*, (2019).
- [8] Mario Vazquez Corte and Luis V Montiel, ‘Novel matrix hit and run for sampling polytopes and its gpu implementation’, *arXiv preprint arXiv:2104.07097*, (2021).
- [9] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa, ‘Safe exploration in continuous action spaces’, *arXiv preprint arXiv:1801.08757*, (2018).
- [10] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine, ‘Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates’, in *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, (2017).
- [11] Chen Hou and Qianchuan Zhao, ‘Optimization of web service-based control system for balance between network traffic and delay’, *IEEE Transactions on Automation Science and Engineering*, **15**(3), 1152–1162, (2017).
- [12] Yongshuai Liu, Jiaxin Ding, and Xin Liu, ‘Ipo: Interior-point policy optimization under constraints’, in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4940–4947, (2020).
- [13] Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson, ‘Discrete sequential prediction of continuous actions for deep rl’, *arXiv preprint arXiv:1705.05035*, (2017).
- [14] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al., ‘Stabilizing transformers for reinforcement learning’, in *International conference on machine learning*, pp. 7487–7498. PMLR, (2020).
- [15] Thomas Pierrot, Valentin Macé, Jean-Baptiste Sevestre, Louis Monier, Alexandre Laterre, Nicolas Perrin, Karim Beguir, and Olivier Sigaud, ‘Factored action spaces in deep reinforcement learning’, 2021.
- [16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, ‘Proximal policy optimization algorithms’, *arXiv preprint arXiv:1707.06347*, (2017).
- [17] Arash Tavakoli, Fabio Pardo, and Petar Kormushev, ‘Action branching architectures for deep reinforcement learning’, in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, (2018).
- [18] Chen Tessler, Daniel J Mankowitz, and Shie Mannor, ‘Reward constrained policy optimization’, in *International Conference on Learning Representations*, (2018).
- [19] Ermo Wei, Drew Wicke, and Sean Luke, ‘Hierarchical approaches for reinforcement learning in parameterized action space’, in *2018 AAAI Spring Symposium Series*, (2018).
- [20] David Winkel, Niklas Strauß, Matthias Schubert, and Thomas Seidl, ‘Risk-aware reinforcement learning for multi-period portfolio selection’, in *European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, (2022).
- [21] Linrui Zhang, Li Shen, Long Yang, Shixiang Chen, Xueqian Wang, Bo Yuan, and Dacheng Tao, ‘Penalized proximal policy optimization for safe reinforcement learning’, in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 3744–3750, (2022).

G Autoregressive Policy Optimization for Constrained Allocation Tasks

Venue The Thirty-eighth Annual Conference on Neural Information Processing Systems

Declaration of authorships The research idea was initially proposed by David Winkel and developed and discussed with Niklas Stauss and Matthias Schubert. David Winkel and Niklas Strauss did the implementation and experiments. The other co-authors also participated in some of the discussions. David Winkel and Niklas Strauss wrote the manuscript, which was improved by all co-authors.

Publication

Autoregressive Policy Optimization for Constrained Allocation Tasks

David Winkel* Niklas Strauß*
Maximilian Bernhard Zongyue Li Thomas Seidl Matthias Schubert
Munich Center for Machine Learning, LMU Munich
{winkel,strauss,bernhard,li,seidl,schubert}@dbs.ifi.lmu.de

Abstract

Allocation tasks represent a class of problems where a limited amount of resources must be allocated to a set of entities at each time step. Prominent examples of this task include portfolio optimization or distributing computational workloads across servers. Allocation tasks are typically bound by linear constraints describing practical requirements that have to be strictly fulfilled at all times. In portfolio optimization, for example, investors may be obligated to allocate less than 30% of the funds into a certain industrial sector in any investment period. Such constraints restrict the action space of allowed allocations in intricate ways, which makes learning a policy that avoids constraint violations difficult. In this paper, we propose a new method for constrained allocation tasks based on an autoregressive process to sequentially sample allocations for each entity. In addition, we introduce a novel de-biasing mechanism to counter the initial bias caused by sequential sampling. We demonstrate the superior performance of our approach compared to a variety of Constrained Reinforcement Learning (CRL) methods on three distinct constrained allocation tasks: portfolio optimization, computational workload distribution, and a synthetic allocation benchmark. Our code is available at: <https://github.com/niklasdbs/paspo>.

1 Introduction

Continuous allocation tasks are a class of problems where an agent needs to distribute a limited amount of resources over a set of entities at each time step. Many complex real-world problems are formulated as allocation tasks, and state-of-the-art solutions rely on using Reinforcement Learning (RL) to learn effective policies [6, 26, 3, 20, 27]. Notable examples include portfolio allocation tasks, where portfolio managers must allocate the available financial resources among various assets [27], or allocation tasks of computational workloads to a set of compute instances in data centers [3]. In many cases, allocation tasks come with allocation constraints [6, 20, 27, 26], such as investing at most 30 % of the portfolio into a specific subset of the assets or to restrict the maximum workload to certain servers in a data center. Formally, allocation constraints are expressed as linear constraints and form a system of linear inequalities, geometrically describing a convex polytope. Each point in this polytope describes a possible allocation and each dimension corresponds to one of the entities. Allocation tasks often require hard constraints, i.e., constraints that are explicitly given and must be satisfied at any point in time. However, most of the existing CRL literature focuses on soft constraints that are not explicitly given [2, 29, 31, 14, 25]. These approaches typically cannot guarantee constraint satisfaction and tend to have many constraint violations during training. The majority of these methods approximate the cumulative costs of constraint violations and optimize the cumulative reward while trying to adhere to the maximum cumulative costs. While less explored, there exist

*Both authors contributed equally.

several techniques that ensure the satisfaction of hard constraints [18, 6, 11, 10, 20]. These approaches might generate actions that do not satisfy the constraints but utilize a correction mechanism to map the actions back into the valid action space. In addition, most of these approaches are restricted to off-policy algorithms [11, 20]. In another line of research, solutions tailored for constrained allocation tasks have been proposed [27, 28]. However, these solutions are severely limited since they can only handle a specific subset of linear constraints and cannot handle more than two.

In this paper, we propose Polytope Action Space Policy Optimization (PASPO), a novel RL-based method that, firstly, decomposes the action space into several dependent sub-problems and, secondly, autoregressively computes the allocations step-by-step for each entity individually. In contrast to previous methods for hard constraints, we directly generate an action within the action space. This makes the correction of invalid actions unnecessary and, thus, avoids potential sampling bias introduced by the correction. Our new decomposition approach is implemented in a neural network-based policy function, which can be employed in on-policy and off-policy RL algorithms. We show that initialization bias can prevent proper exploration in early training which leads to premature convergence. Thus, we propose a de-biasing mechanism to stabilize exploration in early training stages.

We evaluate our approach against various baselines on three distinct allocation tasks: portfolio optimization, distributing computational workload in data centers, and a synthetic benchmark. These experiments demonstrate that our approach can outperform existing methods consistently and show the importance of the proposed de-biasing mechanism.

To summarize, the main contributions of our paper are:

- A new autoregressive stochastic policy function applicable to arbitrary convex polytope action spaces of constrained allocation tasks.
- A new de-biasing mechanism to prevent premature convergence to a sub-optimal policy.
- An empirical evaluation that optimizes our new policy function using PPO [22] and demonstrates improved results compared to state-of-the-art CRL methods.

The remainder of the paper is structured as follows: In Section 2, we provide an overview of the related work in CRL, constrained allocation tasks, and autoregressive policy functions. Afterward, we formalize constrained allocation tasks in Section 3 and present our novel approach in Section 4. Section 5 describes the results of our experimental evaluation, Section 6 briefly discusses limitations and future work before Section 7 concludes the paper.

2 Related Work

Resource allocation tasks are a widely researched area with numerous applications spanning logistics, power distribution, computational load balancing, security screening, and finance [6, 26, 3, 20, 27]. We identify three key research directions that are particularly important when discussing resource allocation tasks.

Safe Reinforcement Learning The majority of work in CRL addresses soft constraints, a setting often referred to as Safe RL. We will provide a brief overview of the most important methods in this field. For a more comprehensive examination of Safe RL, we direct readers to the survey papers by [15, 12]. A common technique in Safe RL is the use of Lagrangian relaxation [4, 15]. Several works employ primal-dual optimization to leverage the Lagrangian duality, including [9, 23, 13]. Another frequently used approach involves different penalty terms [25, 14, 31]. The authors of IPO [14] propose to use logarithmic barrier functions. CPO [2] extends TRPO [21] to ensure near-constraint satisfaction with each update. Additionally, two-step approaches such as FOCOPS [32] and CUP [30] are popular in the field. However, unlike our method, these approaches do not guarantee strict constraint satisfaction, particularly during training.

Hard Constraints Although less studied than Safe RL, several works address hard instantaneous constraints on actions to ensure full constraint satisfaction at any time step. Most of these approaches employ mechanisms to correct infeasible actions, i.e., those that violate constraints, into feasible actions [18, 6, 20, 11]. In contrast, our method always generates feasible actions without the need for correction. OptLayer [18] is one of the most prominent examples in this field, which employs OptNet [5] to map infeasible actions to the nearest feasible action. Similarly, [20] propose a more

efficient projection on the polytope action space than OptLayer. The authors of [6] focus on resource allocation with hierarchical allocation constraints by proposing a faster approximate version of OptLayer. In [11], the authors propose an off-policy algorithm based on the generalized reduced gradient method [1] to handle non-linear hard constraints by projecting infeasible actions. In contrast, our method is not limited to off-policy algorithms.

In [27, 28], the action space is decomposed into independent subspaces. However, these approaches can only handle up to two allocation constraints. Furthermore, they are only applicable to binary allocation constraints. In contrast, our approach can handle an arbitrary number of constraints as well as any type of linear allocation constraints.

Action Space Decomposition/Factorization The decomposition or factorization of multi-dimensional action spaces has been examined in several works [24, 16, 19]. A notable example is [24], in which the authors discretize a continuous action space into several independent action branches, each parameterized by individual network branches. In [16], a variant of DQN [17] that discretizes a continuous action space into multiple discrete dimensions is proposed. These dimensions are sequentially parameterized, conditional on the previous sub-actions. Similarly, [19] propose an autoregressive factorization of an unconstrained action space into dependent sub-problems. Unlike our approach, these methods focus either on decomposing continuous action spaces into discrete action spaces or decomposing unconstrained action spaces. However, the decomposition of arbitrary convex polytope action spaces into tractable sub-action spaces remains a non-trivial challenge that our approach addresses.

3 Problem Description

An allocation task can be described as a finite-horizon Markov decision process (MDP) (S, A, T, R, γ) , where S represents the state space, A the action space, $T : S \times A \times S \rightarrow [0, 1]$ the state transition function, R the reward function, and $\gamma \in [0, 1]$ a discount factor. The goal of this task is to find a policy π maximizing the expected cumulative reward $J_R^\pi = \mathbb{E}_\pi [\sum_{t=1}^n \gamma^t R(s_t, \pi(s_t), s_{t+1})]$.

The action a is an allocation $a = \{a_1, \dots, a_n\} \in A$ over a set of n entities $E = \{e_1, \dots, e_n\}$ at each time step. Each element a_i of the action vector a represents the proportion allocated to entity e_i . Furthermore, allocation tasks require a complete allocation, i. e., $\sum_{i=1}^n a_i = 1$ and allocations cannot be negative ($a_i \geq 0$). Thus, the action space of unconstrained allocation tasks forms an n -dimensional standard simplex. A visualization of an unconstrained allocation action space is provided in Figure 1a.

Allocation tasks frequently include constraints, such as allocating at most 30% to a subset of the entities. An example of a constrained action space is visualized in Figure 1b. Formally, an allocation constraint can be expressed as a linear inequality $\sum_{i=1}^n c_i a_i \leq b$, where c_i denotes the weighting of the allocation variable a_i of entity e_i and $b \in \mathbb{R}$ denotes the corresponding constraint limit. For the sake of readability and simplicity, we only define \leq constraints since $a \geq b$ can be transformed into $-a \leq -b$ and $a = b$ can be rewritten as $a \leq b$ and $-a \leq -b$.

The action space A of constrained allocation tasks can be easily expressed by a set of linear inequalities, defining a polytope $A = \{a \in [0, 1]^n | Ca \leq b\}$, where

$$C \in \mathbb{R}^{m \times n} = \begin{bmatrix} c_{11} & \dots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{m1} & \dots & c_{mn} \end{bmatrix} \quad (1)$$

is a matrix of coefficients for the m constraints, including those linked to the simplex constraints $\sum_{i=1}^n a_i = 1$ and $a_i \geq 0 \forall i \in \{1, \dots, n\}$, as well as all coefficients for additional allocation constraints. Let $a \in [0, 1]^n$ represent an allocation vector and $b \in \mathbb{R}^m$ is the vector of constraint limits.

Alternatively, constrained allocation tasks can be defined using the framework of constrained Markov decision processes (CMDPs). A CMDP extends the standard MDP by a number of cost functions to incorporate the constraints. The goal is to maximize the expected cumulative reward while satisfying m constraints on the expected cumulative costs. The expected cumulative costs for the k -th cost function CF_k are defined as $J_{CF_k}^\pi = \mathbb{E}[\sum_{t=0}^T \gamma^t CF_k(s_t, a_t)]$. The m constraints to be satisfied in the CMDP are then stated as $J_{CF_k}^\pi \leq d_k$, where d_k denotes the cost limit with

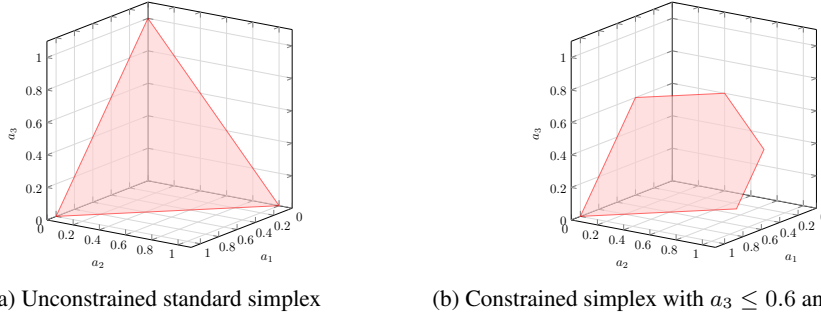


Figure 1: **Examples of 3-dimensional allocation action spaces** (a) unconstrained and (b) constrained (valid solutions as red area).

$k \in \{1, \dots, m\}$. To formulate constrained allocation tasks using CMDPs, the cost functions can be defined as $CF_k(s, a) = \max\{0, (Ca)_k - b_k\}$ to measure any allocation constraint violation as a cost. In addition, strict adherence to all allocation constraints at any point in time is required, i.e., $d_k = 0$. By formulating constraint allocation tasks using CMDPs, it becomes possible to use existing methods from Safe RL for soft constraints. However, these methods cannot guarantee constraint satisfaction at all times [15]. Let us note that our method does not use cost functions, instead it samples actions directly from the constrained action space.

4 Polytope Action Space Policy Optimization (PASPO)

Our approach PASPO autoregressively computes the allocation to every single entity in an iterative process until all allocations are fixed. We will later show that this step-wise decomposition allows for a tractable parametrization of the action space.

4.1 Autoregressive Polytope Decomposition

PASPO starts by determining the feasible interval $[a_1^{min}, a_1^{max}]$ for allocations into the first entity e_1 . Then, we sample the first allocation a_1 from this interval. The details of the sampling process will be further discussed in Section 4.2. Fixing an allocation impacts the shape of the remaining action space. Thus, we have to compute the shape of the polytope $A^{(2)}$ described by $C^{(2)}$ and $b^{(2)}$ before we can sample the next allocation a_2 .

Each iteration i starts with determining the interval $[a_i^{min}, a_i^{max}]$ of all feasible values for a_i . Geometrically, this interval is bounded by the minimum and the maximum value of the remaining polytope $A^{(i)}$ in the i -th dimension associated with the allocation a_i . To determine a_i^{min} , we solve the following linear program:

$$\begin{aligned} & \text{minimize} && a_i \\ & \text{s.t.} && C^{(i)} a^{(i)} \leq b^{(i)} \end{aligned}$$

where $C^{(i)}$ are the constraint coefficients for the entities e_i, \dots, e_n , $b^{(i)}$ are the adjusted constraint limits, and $a^{(i)}$ describes the unfixed allocations. We determine a_i^{max} by solving the respective maximization problem. For the first iteration $i = 1$, we define $C^{(1)} = C$, $b^{(1)} = b$ and $a^{(1)} = a$. After sampling an allocation a_i from the interval $[a_i^{min}, a_i^{max}]$. The resulting polytope $A^{(i+1)}$ for the next iteration $i + 1$ is described by the following inequality system:

$$\underbrace{\begin{bmatrix} c_{1,i+1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{m,i+1} & \cdots & c_{m,n} \end{bmatrix}}_{C^{(i+1)}} \underbrace{\begin{bmatrix} a_{i+1} \\ \vdots \\ a_n \end{bmatrix}}_{a^{(i+1)}} \leq \underbrace{\begin{bmatrix} b_1^{(i)} \\ \vdots \\ b_m^{(i)} \end{bmatrix}}_{b^{(i+1)}} - a_i \underbrace{\begin{bmatrix} c_{1,i} \\ \vdots \\ c_{m,i} \end{bmatrix}}_{b^{(i)}} \quad (2)$$

To define the new coefficient matrix $C^{(i+1)}$ (red), we remove the first column of the coefficient matrix of the previous iteration $C^{(i)}$. To calculate the new vector $b^{(i+1)}$ of constraint limits, we subtract

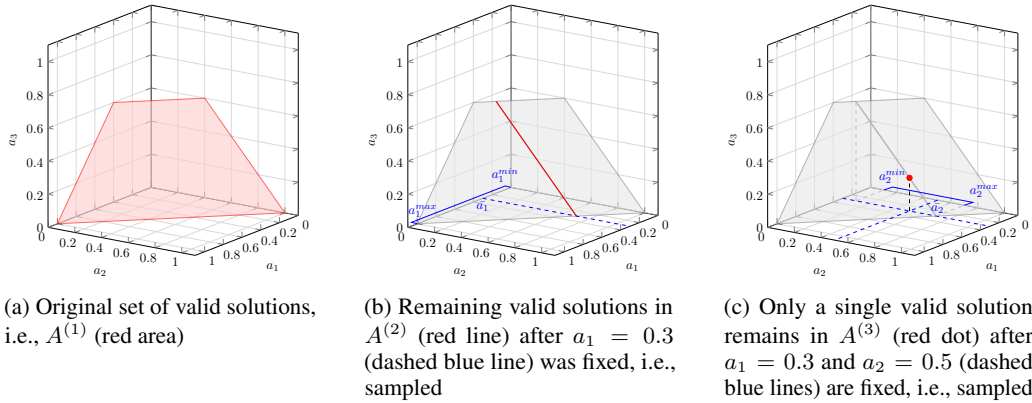


Figure 2: **Example of sampling process** of an action $a = (a_1, a_2, a_3)$ in a 3-dimensional constrained allocation task.

the removed column (blue) scaled by the fixed allocation a_i from the previous constraint limits $b^{(i)}$ (yellow). We iterate over all entities until we determine a_{n-1} . Allocation a_n is already determined as soon as the allocations a_1, \dots, a_{n-1} are fixed because of the simplex constraint $\sum_{i=1}^n a_i = 1$. Sampling an allocation using this approach always guarantees constraint satisfaction and it is possible to sample any action in the constrained action space. A formal proof of these guarantees can be found in Appendix D.

Figure 2 displays a visualization of the process for a 3-dimensional case. The set of valid solutions before any allocations have been fixed is shown in Figure 2a. Figure 2b depicts the first iteration after $a_1 = 0.3$ has been determined, and the resulting new polytope $A^{(2)}$, i.e., a set of valid solutions, shrinks to the red line is shown. Figure 2c shows the second iteration after also $a_2 = 0.5$ has been determined. It can be seen that the new polytope $A^{(3)}$ contains only a single valid solution represented as a red dot, making a third iteration unnecessary since the only remaining solution is to allocate $a_3 = 0.2$, resulting in a final allocation of $a = (0.3, 0.5, 0.2)$.

4.2 Parameterizable Policy Process

Our goal is to define a learnable stochastic policy function over the action space. For unconstrained allocation tasks, a Dirichlet distribution can be used to parameterize the action space [26, 28]. Unfortunately, to the best of our knowledge, there is no known parameterizable, closed-form distribution function over arbitrary convex polytopes as in our setting. In fact, even uniform sampling over a convex polytope is an active research problem [8].

We sequentially constructed an action a from the polytope action space A in the previous section. Now, we describe how to utilize this process to define a parameterizable policy function over the action space A . We model the distribution for allocating each individual entity using a beta distribution that is normalized to the range $[a_i^{min}, a_i^{max}]$. This distribution is also known as the four-parameter beta distribution [7]. Its probability density function is defined as:

$$p(x; \alpha, \beta, a_i^{min}, a_i^{max}) = \frac{(x - a_i^{min})^{\alpha-1} (a_i^{max} - x)^{\beta-1}}{(a_i^{max} - a_i^{min})^{\alpha+\beta-1} B(\alpha, \beta)},$$

where $B(\alpha, \beta)$ is the beta function. It is important to note that any other parameterizable distributions with bounded support in the range $[a_i^{min}, a_i^{max}]$ can be used, such as a squashed Gaussian distribution. However, our preliminary experiments indicated that the beta distribution performs particularly well.

To optimize the policy $\pi_\theta(s)$ over the complete allocations, we follow the approach of [19] for training an autoregressively dependent series of sub-policies. A fixed but arbitrary order of entities is used for sampling the allocations a_i . The sub-policy $\pi_\theta^i(a_i | a_1, \dots, a_{i-1})$ is conditional on the previous allocations a_1, \dots, a_{i-1} . Using this autoregressive dependence structure, the policy is defined as: $\pi_\theta(a|s) = \pi_\theta^1(a_1|s) \cdot \pi_\theta^2(a_2|s, a_1) \cdot \dots \cdot \pi_\theta^n(a_n|s, a_1, \dots, a_{n-1})$. This policy can be jointly optimized. We parameterize each sub-policy using a neural network that receives an embedding of the state and the previously selected actions as input.

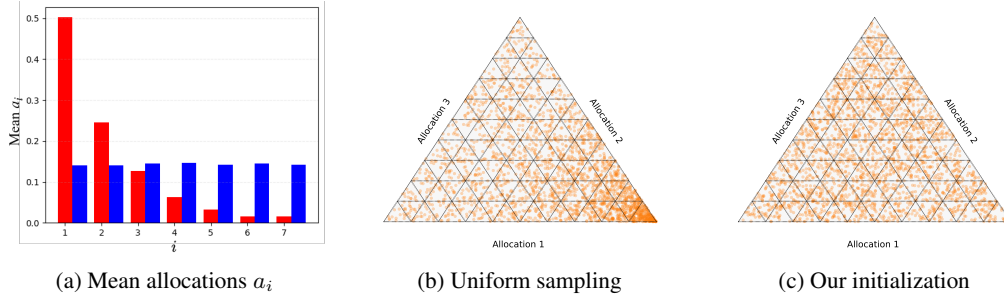


Figure 3: **The impact of initialization in an unconstrained simplex.** (a) Mean allocations a_i to each entity in a seven entity setup when sampling each individual allocation using the **uniform distribution** (red) vs. **our initialization** (blue). (b,c) Distribution of 2500 allocations in a three entity setup when sampling each individual allocation uniformly (b) or using beta distributions with parameters set according to our initialization (c).

Algorithm 1 Maximum likelihood estimation of parameter de-biasing terms

Input: Polytope $A = \{a \in \mathbb{R}_{0,+}^n \mid Ca \leq b\}$, number of samples k

Output: Beta distribution shape parameters $\hat{\alpha}_i, \hat{\beta}_i$ as de-biasing terms

- 1: Let $A_j^{(1)} = A$ for $j = \{1, \dots, k\}$
 - 2: Sample k allocations $\{a_{(j)}\}_{j=1\dots k}$ uniformly within A via rejection sampling
 - 3: **for** each dimension i in $\{1, \dots, n\}$ **do**
 - 4: **for** each sample j in $\{1, \dots, k\}$ **do**
 - 5: Calculate interval $[a_{(j)i}^{min}, a_{(j)i}^{max}]$ using LP based on $A_j^{(i)}$
 - 6: Normalize sampled allocation to support $[0,1]$ of beta distribution: $a_{(j)i}^{norm} = \frac{a_{(j)i} - a_{(j)i}^{min}}{a_{(j)i}^{max} - a_{(j)i}^{min}}$
 - 7: Compute polytope $A_j^{(i+1)}$ from $A_j^{(i)}$ using sampled allocation $a_{(j)i}$ (see Eq. 2)
 - 8: **end for**
 - 9: ML estimation of beta distribution parameters $\hat{\alpha}_i, \hat{\beta}_i$ using $\{a_{(j)i}^{norm}\}_{j=1\dots k}$
 - 10: **end for**
-

An entropy term is often used to encourage exploration. However, our policy does not have a closed-form solution for entropy. Therefore, we follow [19] to empirically estimate the entropy:

$$H_{\text{emp}}(\pi_{\theta}(\cdot|s)) = \mathbb{E}_{a \sim \pi_{\theta}(\cdot|s)} \left[\sum_{i=1}^n H(\pi_{\theta}^i(\cdot|s, a_1, \dots, a_{i-1})) \right] \quad (3)$$

Here, H denotes the entropy of the beta distribution. We compute the expectation within each training batch to estimate the entropy of the complete policy function over the entropies of single actions. Let us note that when using an off-policy algorithm, the actions must be resampled using the current policy. As the current policy might have a significantly different parametrization than the sampling policy, we have to generate actions based on the current policy to estimate the entropy properly.

4.3 Policy Network Architecture

We create an embedding of the state using an MLP, denoted as $f_{\theta}(s) = MLP(s)$. We parameterize the probability distribution over allocations for each entity using an MLP $\pi_{\theta_i}^i(s) = MLP(f_{\theta}(s), a_1, \dots, a_{i-1})$, which receives the latent encoding of the state and the previously sampled allocations a_1, \dots, a_{i-1} as input. Note that each of the MLPs $\pi_{\theta_i}^i$ has its own parameters. For further details, we refer to the Appendix.

4.4 De-biasing Mechanism

A drawback of generating actions by an autoregressive process is that a random initialization of the beta distributions leads to a sampling bias towards the entities selected earlier in the process. The

effect is caused by the autoregressive dependency structure of our process. To sample an allocation of 80% for entity a_i the cumulated fixed allocations for earlier entities $\sum_{j=1}^{i-1} a_j$ must be at most 20%. However, this is rather unlikely if we initialize the distribution for all entities in a similar way. This effect can be observed in the red bars of Figure 3a. The red bars correspond to the average allocation for each dimension when uniformly drawing from an unconstrained seven-dimensional simplex with our autoregressive process for each entity e_i . As expected, the mean for the first dimension is 0.5, which is the mean of a uniform distribution over the interval $[0, 1]$. Correspondingly, the mean is decreased by half for any successive further entity until entity 6, which has the same mean as entity 7 due to the simplex constraint. Even though the bias is more complex for constrained allocation spaces, a similar effect can be expected.

For policy gradient methods, such a bias in the initialization of the policy function can lead to convergence to poorly performing policies or long training times. As the initial policy is crucial for ensuring sufficient exploration of the state-action space, a biased initial distribution leads to underexplored regions in the state-action space. Consequently, well-performing actions might not be discovered. To counter this effect, we propose a de-biasing mechanism that adjusts the initial parameters of beta distributions to estimate a uniform sampling over the joint action space. During learning, the amount of required exploration decreases, and the parameters of the policy function are optimized to increase the cumulative rewards. Thus, the impact of our de-biasing mechanism should diminish over time. We achieve this effect by adding a de-biasing term to the linear layers' initial bias terms, predicting α_i and β_i for entity i . As the default initialization of the bias terms has zero means, the first iterations use α and β values close to the de-biasing terms.

To determine suitable initial values for each iteration step, we proceed as described in Algorithm 1. We start by uniformly sampling n data points from the complete action space A . We do this by rejection sampling, i.e., we sample over the standard simplex and reject the samples outside the action polytope A . To determine the parameters corresponding to the acquired uniform sample, we project any allocation for each entity to a standard interval between $[0, 1]$. However, for this step, we have to determine the interval $[a_i^{min}, a_i^{max}]$ for each entity following the above process. We determine the relative position in this interval, corresponding to the position in the named standard interval. After collecting relative values for each sample and entity, we employ the standard maximum likelihood estimator to generate an empirical estimate of the α_i and β_i for each entity e_i . The blue bars in Figure 3a correspond to the results on the unconstrained seven-dimensional unconstrained simplex. Figure 3b shows autoregressive sampling based on uniform distribution, whereas Figure 3c displays the result of our initialization for a three-dimensional example. It can be seen that the result of our initialization of the autoregressive process closely resembles a uniform distribution over the complete action space.

5 Experiments

In this section, we provide an extensive experimental evaluation of our approach in various scenarios demonstrating its ability to handle various allocation tasks and constraints. We use two real-world tasks: Portfolio optimization [27] and compute load distribution [3]. Additionally, we create a synthetic benchmark with a reward surface generated by a randomly initialized MLP. Each of these tasks comes with a different set of allocation constraints. We will briefly describe each setting in the following and refer the reader to the Appendix for more details.

Portfolio Optimization Portfolio optimization is a prominent constrained allocation task. In this task the agent has to allocate its wealth over 13 assets at each time step. We use the environment of [27]. Each investment period contains 12 months and the investor needs to reallocate the portfolio each month. This environment is highly stochastic since each trajectory is sampled from a hidden Markov model fitted on real-world NASDAQ-100 data. After every 5120 environment steps, we run eight parallel evaluations on 200 fixed trajectories. Constraints in this setting define minimum and maximum allocation to groups of assets. Additionally, we add constraints where the constraint coefficients in C correspond to portfolio measures like a minimum dividend yield or a maximum on the CO2 intensity.

Compute Load Distribution The environment is based on the paper of [3] and simulates a data center in which computational jobs need to be split into sub-jobs to enable parallel processing across nine servers. Here, we use five constraints that are randomly sampled as follows: First, we sample

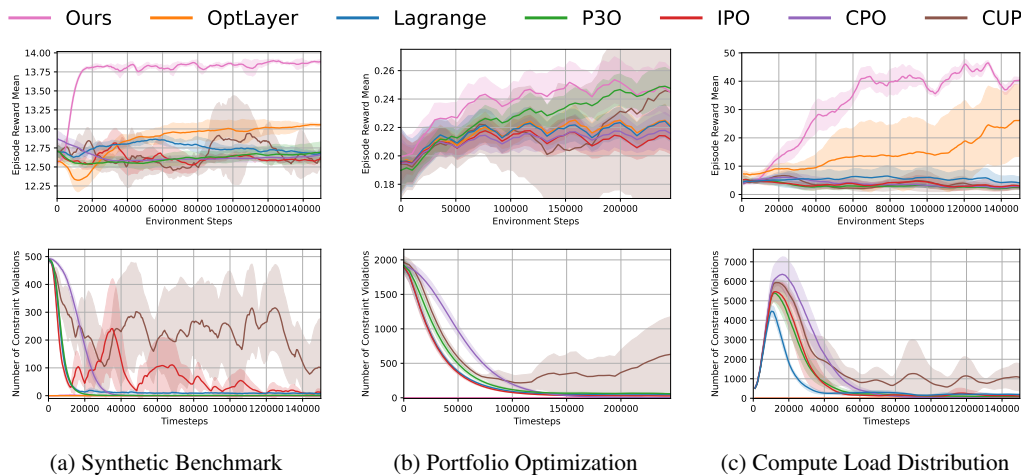


Figure 4: **Learning curves of all methods in three environments.** The x-axis corresponds to the number of environment steps. The y-axis is the average episode reward (first row), and the number of constraint violations during every epoch (second row). For portfolio optimization (b) we report the performance running eight evaluation on 200 fixed market trajectories. This is because in training, every trajectory is different which makes comparisons hard. Curves smoothed for visualization.

the number of affected entities for each constraint. We then sample the constraint coefficients from the range $[0, 1]$.

Synthetic Environment In addition to the aforementioned environments, we propose a synthetic benchmark. The reward surface consists of an MLP with random weights. Each episode comprises two states. As it is completely deterministic, it provides a simple yet effective way to benchmark approaches for constrained allocation tasks. In this setting, we create the constraints by randomly sampling 30 points and use their convex hull as the polytope defining the action space. We utilize a seven-dimensional setting with 611 constraints in our experiments.

5.1 Experimental Setup

We train PASPO using PPO [22] and compare our approach to various baselines, including state-of-the-art approaches for constrained allocation tasks and Safe RL. Specifically, we compare PASPO with five representative approaches from Safe RL: CPO [2], CUP [30], IPO [14], P3O [31], and PPO with Lagrangian relaxation. Additionally, we compare our method to OptLayer [18], a popular projection-based method for linear hard constraints. To maintain a consistent and fair comparison across different methods, we use the same hyperparameters across the different methods if possible. Many Safe RL approaches have difficulties handling equality constraints [11]. Therefore, we use a Dirichlet distribution to represent the policy in the baselines, thereby ensuring satisfaction of the simplex equality constraint. We do not share the parameters between the policy and value function. We use a fully-connected MLP with two hidden layers of 32 units and ReLU non-linearities for each policy, cost, and value function. In our approach, the state encoder and each policy head consists of a two-layer MLP. The training process is run for 150,000 steps and the results are averaged over five different seeds. In the portfolio optimization task, we use ten different seeds due to the stochasticity of the financial environment and train for 250,000 steps. Given the relatively small network sizes, training is conducted exclusively on CPUs. We implement our algorithm and the baselines using RLlib and PyTorch. More details regarding the environments, training, and hyperparameters can be found in the Appendix.

5.1.1 Performance of PASPO

We visualize the performance and constraint violations of all methods across our three environments in Figure 4. A tolerance of $1e^{-3}$ is used for evaluating constraint violations and we report the total number of violations per episode. In all three environments, PASPO converges faster to a higher

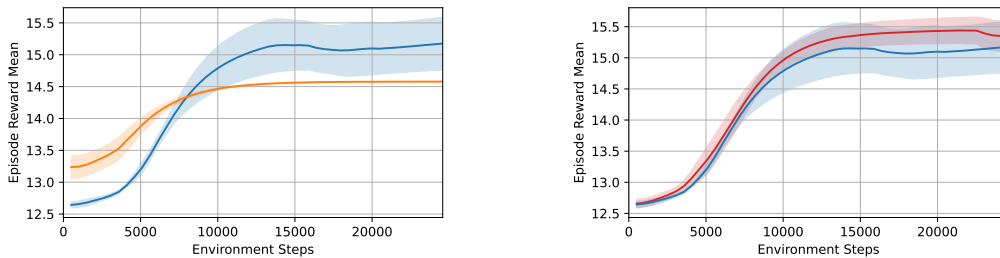


Figure 5: **Ablations** in (a) show the performance of our approach with (blue) and without (orange) the de-biased initialization. In (b) depicts the impact of the allocation order. We reverse the allocation order (red).

average return compared to baselines. Additionally, while all compared soft-constraint methods display constraint violations, only the hard constraint approaches PASPO and OptLayer guarantee to permanently satisfy the constraints. Finally, we can observe that the variance of PASPO is rather low compared to other methods. However, in portfolio optimization task (b) our approach displays some variance which we attribute to the stochasticity of the environment. Overall, these results demonstrate that our approach is not only able to consistently outperform other algorithms in terms of rewards but also guarantees no constraint violations.

5.1.2 Importance of de-biased Initialization and Order

We conduct ablation studies to investigate the impact of our de-biased initialization and the order of entity allocation on our synthetic benchmark. No constraints are applied except for the simplex constraint to highlight the effects. The results, shown in Figure 5, indicate that without de-biased initialization (orange in (a)), learning is slower and converges prematurely to a sub-optimal policy. In (b), we explore the impact of allocation order by reversing it (red) and observe no significant performance difference. This indicates that our approach is robust to the allocation order due to the use of the de-biasing initialization.

6 Limitations and Future Work

While PASPO guarantees that constraints are always satisfied, it is considerably more computationally expensive than standard neural networks in allocation tasks with many entities, as the sampling of each action requires solving a series of linear programs. RL in high-dimensional continuous action spaces is a very challenging task. Our approach cannot overcome this issue and also struggles in very high-dimensional settings. For future work, we plan to extend PASPO to also incorporate state-dependent constraints. While we evaluate our approach only on benchmarks with hard constraints, it can be applied to settings with both hard and soft cumulative constraints. In these scenarios, our method for handling hard constraints can be easily combined with most Safe RL algorithms to handle soft cumulative constraints.

7 Conclusion

In this paper, we examine allocation tasks where a certain amount of a resource has to be distributed over a set of entities at every step. This problem has many applications like logistics tasks, portfolio management, and computational workload processing in distributed environments. In all these applications, the set of feasible allocations might be bound by a set of linear constraints. Formally, these restrict the action space to a convex polytope. To define a stochastic policy function that can be used with policy gradient methods in RL, we propose an autoregressive process that computes allocation sequentially. We employ linear programming to compute the range of feasible allocations for an entity given the already fixed allocations of other entities. Our policy function consists of a sequence of one-dimensional beta distributions where the shape parameters α and β are learned by neural networks. To counter the effect of initialization bias, we utilize a de-biasing mechanism to ensure sufficient exploration and prevent premature convergence to a sub-optimal policy. In our

experiments, we demonstrate that our novel method PASPO yields better results than state-of-the-art approaches while not having any constraint violations. Furthermore, we show that our initialization method yields better results than random initializations and counters the impact of the allocation order.

References

- [1] J. Abadie. Generalization of the wolfe reduced gradient method to the case of nonlinear constraints. *Optimization*, pages 37–47, 1969.
- [2] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.
- [3] L. Ale, S. A. King, N. Zhang, A. R. Sattar, and J. Skandaraniyam. D3pg: Dirichlet ddpq for task partitioning and offloading with constrained hybrid action space in mobile-edge computing. *IEEE Internet of Things Journal*, 9(19):19260–19272, 2022.
- [4] E. Altman. *Constrained Markov Decision Processes*, volume 7. CRC Press, 1999.
- [5] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.
- [6] A. Bhatia, P. Varakantham, and A. Kumar. Resource constrained deep reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 610–620, 2019.
- [7] J. Carnahan. Maximum likelihood estimation for the 4-parameter beta distribution. *Communications in Statistics-Simulation and Computation*, 18(2):513–536, 1989.
- [8] Y. Chen, R. Dwivedi, M. J. Wainwright, and B. Yu. Fast mcmc sampling algorithms on polytopes. *Journal of Machine Learning Research*, 19(55):1–86, 2018.
- [9] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18(167):1–51, 2018.
- [10] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.
- [11] S. Ding, J. Wang, Y. Du, and Y. Shi. Reduced policy optimization for continuous control with hard constraints. *Advances in Neural Information Processing Systems*, 36, 2024.
- [12] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.
- [13] Q. Liang, F. Que, and E. Modiano. Accelerated primal-dual policy optimization for safe reinforcement learning. *arXiv preprint arXiv:1802.06480*, 2018.
- [14] Y. Liu, J. Ding, and X. Liu. Ipo: Interior-point policy optimization under constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4940–4947, 2020.
- [15] Y. Liu, A. Halev, and X. Liu. Policy learning with constraints in model-free reinforcement learning: A survey. In *The 30th international joint conference on artificial intelligence (ijcai)*, 2021.
- [16] L. Metz, J. Ibarz, N. Jaitly, and J. Davidson. Discrete sequential prediction of continuous actions for deep rl. *arXiv preprint arXiv:1705.05035*, 2017.
- [17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- [18] T.-H. Pham, G. De Magistris, and R. Tachibana. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6236–6243. IEEE, 2018.
- [19] T. PIERROT, V. Macé, J.-B. Sevestre, L. Monier, A. Laterre, N. Perrin, K. Beguir, and O. Sigaud. Factored action spaces in deep reinforcement learning, 2021. URL <https://openreview.net/forum?id=naSAkn2Xo46>.

- [20] S. Sanket, A. Sinha, P. Varakantham, P. Andrew, and M. Tambe. Solving online threat screening games using constrained action space reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2226–2235, 2020.
- [21] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.
- [22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [23] A. Stooke, J. Achiam, and P. Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.
- [24] A. Tavakoli, F. Pardo, and P. Kormushev. Action branching architectures for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [25] C. Tessler, D. J. Mankowitz, and S. Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2018.
- [26] Y. Tian, M. Han, C. Kulkarni, and O. Fink. A prescriptive dirichlet power allocation policy with deep reinforcement learning. *Reliability Engineering & System Safety*, 224:108529, 2022.
- [27] D. Winkel, N. Strauß, M. Schubert, Y. Ma, and T. Seidl. Constrained portfolio management using action space decomposition for reinforcement learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 373–385. Springer, 2023.
- [28] D. Winkel, N. Strauß, M. Schubert, and T. Seidl. Simplex decomposition for portfolio allocation constraints in reinforcement learning. In K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, and R. Radulescu, editors, *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 2655–2662. IOS Press, 2023. doi: 10.3233/FAIA230573. URL <https://doi.org/10.3233/FAIA230573>.
- [29] L. Yang, J. Ji, J. Dai, L. Zhang, B. Zhou, P. Li, Y. Yang, and G. Pan. Constrained update projection approach to safe policy optimization. *Advances in Neural Information Processing Systems*, 35:9111–9124, 2022.
- [30] L. Yang, J. Ji, J. Dai, Y. Zhang, P. Li, and G. Pan. Cup: A conservative update policy algorithm for safe reinforcement learning. *arXiv preprint arXiv:2202.07565*, 2022.
- [31] L. Zhang, L. Shen, L. Yang, S. Chen, X. Wang, B. Yuan, and D. Tao. Penalized proximal policy optimization for safe reinforcement learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3744–3750, 2022.
- [32] Y. Zhang, Q. Vuong, and K. Ross. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 33:15338–15349, 2020.

A Environments

The implementation for all three environments can be found at: <https://github.com/niklasdbs/paspo>.

A.1 Financial Environment

The financial environment used for testing our approach is based on [28]. The financial market trajectories in this environment are sampled from a hidden Markov model, which was fitted based on real-world NASDAQ-100 data from January 3rd, 2011 to December 1st, 2021. The environment offers differently sized data sets of randomly selected assets contained in the NASDAQ-100. The experiments in the financial environment for this paper are run with 13 assets, which corresponds to

the *model_parameter_data_set_G_markov_states_2_12* data set with the *include_cash_asset=true* option. We initialize the environment with *seed=2*.

Table 1 shows a list of the assets used in the experiments.

For the experiments a random combination of two types of constraints typical for financial tasks is used: (a) a randomly selected subset of assets to either stay above or below an randomly selected allocation threshold and (b) thresholds for financial or environmental portfolio measures that can be calculated as an weighted average of the assets' individual measures, i.e., as a weighted linear combination. A list of these measure can be found in Table 2.

The experiments include 5 constraints, of which the number constraints of type (a) and type (b) is randomly decided. The exact implementation can be found in our code *polytope_loader.py* (*generate_random_fin_env_polytope_rejection_sampling*). We use the seed 2 to generate the constraints.

Index	ISIN	Ticker	Name
1	-	CASH	CASH
2	US5949181045	MSFT	Microsoft Corporation
3	US0567521085	BIDU	Baidu Inc.
4	US00724F1012	ADBE	Adobe Inc.
5	US6937181088	PCAR	Paccar Inc.
6	US67066G1040	NVDA	NVIDIA Corporation
7	US8552441094	SBUX	Starbucks Corporation
8	US4612021034	INTU	Intuit Inc.
9	US0530151036	ADP	Automatic Data Processing Inc.
10	US0231351067	AMZN	Amazon.com Inc.
11	US2786421030	EBAY	eBay Inc.
12	US0311621009	AMGN	Amgen Inc.
13	US7475251036	QCOM	Qualcomm Inc.

Table 1: List of assets used in the environment.

	Est. Total Energy Use To EVIC USD in million	Est. Total CO2 Equivalent Emissions To EVIC USD in million	Est. Weighted Average Cost of Capital, (%)	Est. Dividend yield, (%)	Est. Return On Equity, (%)
CASH	0.00	0.00	0.00	0.00	0.00
MSFT	23.49	1.75	8.19	1.89	40.15
BIDU	70.80	15.17	7.12	0.00	9.01
ADBE	12.98	1.12	6.81	2.28	92.49
PCAR	83.66	10.27	7.31	3.14	-43.29
NVDA	17.17	1.58	6.20	3.00	127.85
SBUX	85.73	6.79	7.24	1.13	26.61
INTU	41.23	17.15	7.87	0.00	21.50
ADP	1.70	0.15	8.12	0.56	22.46
AMZN	4.69	0.39	8.28	0.00	44.48
EBAY	3.49	0.30	9.35	0.02	80.24
AMGN	33.87	3.29	7.09	0.73	37.76
QCOM	47.07	4.09	8.33	2.16	36.06

Table 2: KPI estimates for assets based on 2021 (final year of the used data set, source: Refinitiv); EVIC - Enterprise value including Cash

A.2 Compute Environment

The compute environment used for testing our approach is based on [3]. The agent’s task is to allocate compute jobs to a given set of servers in a data center. A reward is triggered for each job that was completed in a predetermined maximum allowed computation time. The challenge of this environment is that the agent needs to match the queue of jobs still to be allocated with the different computational capabilities of the servers as well as each server’s individual queue of jobs still to be computed. It is assumed that the compute jobs in the environment can be arbitrarily split and computed in parallel. The creation of new compute jobs is triggered by n users and follows a Poisson process. A job is defined by its *payload size*, i.e., the data to be transferred to a server, its *required CPU cycles* for the processing workload, and its maximum allowed time until the job needs to be completely processed. These attributes for the jobs that can be created by each user are randomly sampled at creation of the environment.

The experiments in this paper run with a setup of 9 servers and 9 users that generate compute jobs. The parameter set used is *parameter_set_9_9_id_0*. We initialize the environment with *seed=1*. The randomly sampled specifications for the nine servers can be found in Table 3 and the job attributes created by the nine users can be found in Table 4.

To generate the constraints, we first sample the number of affected entities between 2 and 8 for each constraint and randomly choose the affected entities accordingly. We then uniformly sample constraint coefficients from the interval $[0, 1]$, as well as a corresponding constraint limit between 0 and 1. We use a seed of 1 to generate 5 constraints. The implementation can be found in *polytope_loader.py* (*generate_random_polytope_rejection_sampling*).

Index	Max Compute Cycles per Second
1	2 836 258 583
2	855 913 878
3	652 109 364
4	789 819 414
5	3 187 852 760
6	974 311 629
7	2 005 143 973
8	1 481 875 307
9	2 216 715 088

Table 3: Server Specifications

User	Data Size in Bits per Job	Required Compute Cycles per Job	Average Number of Jobs Created per Interval	Interval Length in Seconds
1	587 168	1 690 694	10	0.01
2	240 447	1 092 255	10	0.01
3	257 396	867 139	10	0.01
4	364 400	819 594	10	0.01
5	387 953	3 463 247	10	0.01
6	309 269	2 300 810	10	0.01
7	44 420	1 129 119	10	0.01
8	318 062	1 092 402	10	0.01
9	490 880	1 044 736	10	0.01

Table 4: User/Job Specifications

A.3 Synthetic Benchmark

In addition to these environments, we propose a synthetic benchmark. Its reward surface consists of an MLP with random weights. An example of the reward surface in three dimensions is visualized in Figure 6. Each episode has two states. Since it is completely deterministic, it provides a simple but

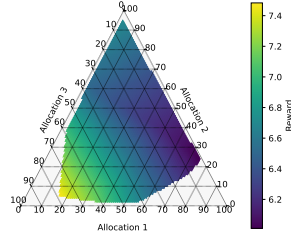


Figure 6: Example of the reward surface of our synthetic benchmark in three dimensions under constraints.

effective way to benchmark approaches for constrained allocation tasks. In this setting, we create the constraints by randomly sampling 30 points and use their convex hull as the polytope defining the action space. We use a seven-dimensional setting with 611 constraints in our experiments. More specifically, the network has one hidden layer and ReLU as a non-linearity. The input layer receives the state (as a number, i.e., 0 or 1) and the action as input and has an output size of 32, the hidden layer has an input size of 32 and output size of 16. The output layer has an input size 16 and a output size of 1. The exact initialization of the neural network weights can be found in our code (synth_env.py: MLPRewardNetwork). To generate the environment we use the seed 1 in our experiments.

To generate the constraints, we sample 30 randomly from a Dirichlet distribution with concentration parameters set to 1. We then build the convex hull of these points and convert the resulting polytope into its halfspace representation, i.e., a system of linear inequalities which we use as constraints. We use the seed of 1 to generate the constraints. This results in 611 constraints. The algorithm to generate the constraints can be found in the code (random_polytope_generator.py)

B Architecture

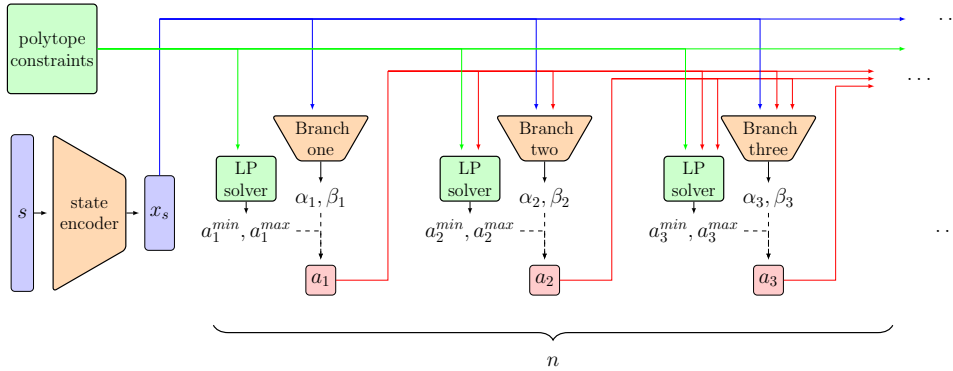


Figure 7: Architecture of PASPO

C Hyperparameters/Training

In Table 5 we list the most important parameters and hyperparameters. The full configurations used can be found in the config files (yaml/hydra based) in our code (run configs directory). We tuned hyperparameters on our synthetic benchmark with five dimensions and five constraints.

We do not train using GPUs because of the small network sizes. We used an internal CPU cluster with consumer machines and servers ranging from 8 to 90 cores and RAM between 32GB and 512GB.

Parameter	Ours	IPO	P3O	CUP	Lag.	OptLayer	CPO
Training env steps	150,000 (synth, compute), 250,000 (portfolio optimization)						
Episode/Rollout length	512 environment steps						
Number of parallel envs	8						
Learning Rate	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	–
Gradient clipping	2.0						
Minibatch size	64						
Optimizer	Adam						
GAE lambda	0.95						
Discount factor	1.0						
No. grad update it per epoch	10 (CPO only for the critic 40)						
PPO clip parameter	0.3	0.3	0.3	0.3	0.3	0.3	–
Entropy coefficient	0.01	0.01	0.01	0.01	0.01	0.01	–
Cost limit	–	1e-3	0.0	0.0	0.0	0.0	0.0

Table 5: The most important Parameters and Hyperparameters for Various Methods

D Guaranteed Constraint Satisfaction

In the following, we proof that our approach PASPO always guarantees constraint satisfaction and that our method is able to sample all possible actions from the constrained action space.

Definitions: Let A be the set of all actions that can be sampled with PASPO, and let $P = \{a \in \mathbb{R}^n | Ca \leq b\}$ be the convex polytope that corresponds to constrained action space. We define $A_*^{(i+1)} = \{a \in \mathbb{R}^n | C^{(i+1)}a^{(i+1)} \leq b_*^{(i+1)}, \forall j = 1, \dots, i : a_j = a_j^*\}$ where $b_*^{(i+1)} = b -$

$$\sum_{j=1}^i a_j^* \begin{bmatrix} c_{1j} \\ \vdots \\ c_{mj} \end{bmatrix} \text{ and } C^{(i+1)} \text{ and } a^{(i+1)} \text{ as defined in the paper.}$$

Thus, $A_*^{(i+1)}$ is the restricted action space after sampling/fixing already the allocations a_1^*, \dots, a_i^* .

Theorem 1. Let $P = \{a \in \mathbb{R}^n | Ca \leq b\} \neq \emptyset$ be the convex polytope that corresponds to a constrained action space. Let A be the set of all the points that can be generated by PASPO. It holds that $A = P$.

Proof. Well-defined: Show that $A^{(n)} \neq \emptyset$ if $P \neq \emptyset$.

Induction over i :

$$i = 1 : A_*^{(1)} = \{a \in \mathbb{R}^n | C^{(1)}a^{(1)} \leq b_*^{(1)}\} = \{a \in \mathbb{R}^n | Ca \leq b\} = P \neq \emptyset$$

$i \rightarrow i + 1 :$

$$(i + 1 \leq n) \quad A_*^{(i)} \neq \emptyset \Rightarrow \exists a^\uparrow, a^\downarrow \in A_*^{(i)} : a_i^\uparrow = a_i^{\min}, a_i^\downarrow = a_i^{\max}$$

Now assume an arbitrary a_i^* is sampled from $[a_i^{\min}, a_i^{\max}]$

$$\Rightarrow \exists \lambda \in [0, 1] : a_i^* = \underbrace{(\lambda a_i^\downarrow + (1 - \lambda) a_i^\uparrow)}_{:= a_i^\lambda}$$

By convexity of polytopes as solution spaces for linear inequality systems, we get:

$$\begin{bmatrix} c_{1,i} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{m,i} & \cdots & c_{m,n} \end{bmatrix} \begin{bmatrix} a_i^\lambda \\ \vdots \\ a_n^\lambda \end{bmatrix} \leq b - \sum_{j=1}^{i-1} a_j^* \begin{bmatrix} c_{1j} \\ \vdots \\ c_{mj} \end{bmatrix} \xLeftrightarrow_{(a_i^\lambda = a_i^*)} \begin{bmatrix} c_{1,i+1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{m,i+1} & \cdots & c_{m,n} \end{bmatrix} \begin{bmatrix} a_{i+1}^\lambda \\ \vdots \\ a_n^\lambda \end{bmatrix} \leq b - \sum_{j=1}^i a_j^* \begin{bmatrix} c_{1j} \\ \vdots \\ c_{mj} \end{bmatrix} \Rightarrow a^\lambda \in A_*^{(i+1)}$$

To show that $A = P$:

$A \subseteq P$: Let $a^* \in A$. In the last step (n), a_n^* is sampled (by design) such that

$$C^{(n)} a_n^* \leq b - \sum_{j=1}^{n-1} a_j^* \begin{bmatrix} c_{1j} \\ \vdots \\ c_{mj} \end{bmatrix} \Leftrightarrow C a^* \leq b \Leftrightarrow a^* \in P$$

$A \supseteq P$: Let $a^* \in P$. $\Leftrightarrow C a^* \leq b \Leftrightarrow C^{(i)} a^* \leq b^{(i)} \forall i \Leftrightarrow a^* \in A_*^{(i)} \forall i$
 \Rightarrow We can construct a^* by sampling a_i^* in every step i . $\Rightarrow a^* \in A$

□

The intuition of why our approach can guarantee the satisfaction of constraints is based on three properties that we utilize: (1) If P_i is the set of solutions to a system of linear inequalities, then by adding further constraints to the system of linear inequalities there will be a new set of solutions P_{i+1} but always such that $P_{i+1} \subseteq P_i$. (2) For any two points a^{\min} and a^{\max} in a convex set it can be implied that there exists a point a^λ for which the following is true for its i -th dimension $\exists \lambda \in [0, 1] : a_i^* = \underbrace{(\lambda a^{\min} + (1 - \lambda) a^{\max})}_{{:=a^\lambda}}_i$ (3) Linear Programming can determine the upper and

lower bounds for single variables in a system of linear inequalities, i.e. $[a_i^{\min}, a_i^{\max}] \forall i$.

We start with the original system of linear inequalities with the solution space $P_1 \neq \emptyset$ which is a convex polytope. We use (3) on P_1 to determine the upper and lower bounds for a_1 , i.e. $[a_1^{\min}, a_1^{\max}]$. We sample a value a_1^* from the range $[a_1^{\min}, a_1^{\max}]$. We know that the solution space P_1 must contain at least one point for which in its 1st dimension $a_1 = a_1^*$ due to (2). In the next step we add the further constraint $a_1 = a_1^*$ to the system of linear inequalities. This updated system of linear inequalities will have the solution space P_2 . Due to (2) $P_2 \neq \emptyset$, as well as $P_2 \subseteq P_1$. We then repeat the entire process and use (3) on P_2 to determine the upper and lower bounds for a_2 , i.e. $[a_2^{\min}, a_2^{\max}]$,....

After the n -th iteration a_n^* will be determined and we then have completed the generation of point $a^* = (a_1^*, \dots, a_n^*) \in P_n \subseteq \dots \subseteq P_1$, i.e. we succeed generating a point a^* that satisfies all original constraints P_1 .

E The Impact of the Allocation Order

As already discussed in the ablations in the main paper, with our de-biased initialization the impact of the allocation order is small. However, without our de-biased initialization, the order of the allocation has a significant impact on the performance, as illustrated in Figure 8.

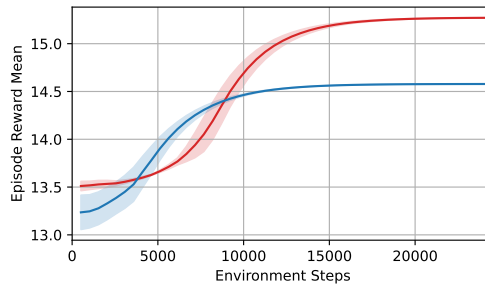


Figure 8: The impact of the allocation order on PASPO without de-biased initialization in the synthetic benchmark with two states, a 7-dimensional action space, and no additional allocation constraints. Blue depicts the standard allocation order (i.e., e_1, e_2, \dots, e_n) and red depicts the reversed allocation order (i.e., the entities are allocated in the reversed order). A significant difference in performance can be observed with respect to the order without our de-biased initialization. In contrast, Figure 5b in the paper shows that with the de-biased initialization the difference is not significant.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: We define the scope by thoroughly describing the considered task and empirically justify our approach in the experiments.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We address the limitations of our method in Section 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Our paper does not contain theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe our experimental setup and the used hyperparameters in the main paper and the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We disclose the code to reproduce the experiments presented in the paper.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We describe the experimental settings in a reasonable level of detail in the main paper and provide complete information in the appendix and the submitted code.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We use a 2-sigma confidence interval around the estimated mean shown in all of our plots.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide information about the compute workers used in the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Our research does not involve human subjects and conforms with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Constrained allocations tasks have no direct societal impacts, however, applying methods for allocation tasks in certain contexts such as finance may indirectly impact society.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: As we are not aware of direct ethical issues or direct negative consequences of our work, we abstain from putting safeguards as described above in place.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We acknowledge the authors of the Portfolio Optimization and Compute Load Distribution environments by citing the corresponding papers.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. New Assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We publish the code of our implementation and benchmark environments. A documentation can be found in the Appendix and code.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and Research with Human Subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The research presented in this paper does not involve crowdsourcing nor human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The research presented in this paper does not involve crowdsourcing nor human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.