Learning to Be Efficient and Fair for Collaborative Order Picking

Smit, Igor G.

Award date:
2023

Department of Industrial Engineering and Innovation Sciences
Department of Mathematics and Computer Science

Learning to Be Efficient and Fair for Collaborative Order Picking

Master Thesis

I.G. (Igor) Smit

In partial fulfillment of the requirements for the degrees of:
Master of Science in Operations Management and Logistics
Master of Science in Data Science in Engineering

Supervisors:
dr. Z.A. (Zaharah) Bukhsh (TU/e)
dr. Y. (Yingqian) Zhang (TU/e)
prof. dr. M. (Mykola) Pechenizkiy (TU/e)

K. (Kostas) Alogariastos, PDEng (Vanderlande)
ir. K. (Kasper) Hendriks (Vanderlande)

Student ID: 1252569

Eindhoven, July 9, 2023

Abstract

Collaborative order picking is a system concept in which human pickers and Autonomous Mobile Robots (AMRs) travel independently through a warehouse. These pickers and vehicles meet at pick locations so that pickers can load items onto the AMRs. To ensure the efficiency of this process, a method is needed that allocates the pickers to pick locations. This method must handle uncertainty and disruptions to make online decisions in dynamic environments. In addition, the method must ensure a fair workload distribution over the human pickers. Oppositely, current methods in collaborative order picking literature consider deterministic one-shot solution approaches that do not consider randomness. Moreover, they do not incorporate workload fairness. Therefore, in this study, we propose a novel deep reinforcement learning approach to sequentially allocate pickers to pick locations in a collaborative order picking system with considerations of pick efficiency and workload fairness. In this approach, we model the warehouse states using a graph. We define a custom network architecture that uses aisle-embeddings to capture regional information and feature separation to extract information from efficiency and workload features efficiently. To find sets of policies that outline the potential trade-offs between efficiency and fairness, we apply a multi-objective deep reinforcement learning approach. We develop a discrete-event simulation model, which we use to train and evaluate our approach. Our experiments show substantial efficiency improvements of up to 40% over the benchmark methods. In addition, we find non-dominated policy sets that outline good trade-offs between fairness and efficiency. Namely, we can considerably increase workload fairness without sacrificing much performance. As a result, the multi-objective policy sets contain multiple policies that outperform the benchmarks on fairness and efficiency. For large warehouses, we achieve policies that improve efficiency by 20% while reducing the workload standard deviation by 90% compared to the existing company benchmark. Based on these results, we suggest starting the development toward real-world implementation of our deep reinforcement learning policies.

Executive Summary

Research Motivation

In this study, we investigated the problem of optimizing the sequential allocation of human pickers to orders at pick locations in a collaborative order picking system. We conducted this study in collaboration with Vanderlande. Vanderlande is a global market leader in future-proof logistic process automation at airports and a leading supplier of warehouse process automation solutions. They are currently developing a new collaborative order picking concept. In this concept, pickers and automated vehicles travel through warehouses independently and meet at pick locations where pickers grab items from the shelve and place them on the vehicles. This method can lead to increased pick capacity of the system.

To achieve a higher pick capacity in this system, Vanderlande requires a method to assign pickers to the pick locations. Existing works in collaborative picking literature propose methods that consider deterministic scenarios without randomness or uncertainty, and they provide complete solutions in one go. However, in practice, the systems are highly dynamic, random delays can occur, and congestion can change the timing of picks. Therefore, we required an online optimization method that can deal with randomness and uncertainty and can allocate pickers while considering the real-time status of the system.

In collaborative order picking systems, human pickers must lift and move many items, which can be heavy labor. If some pickers must pick much larger workloads than others, this can put a considerable physical and mental strain on them. This can lead to worse performance and injuries, eventually. Therefore, ensuring workload fairness is a second important consideration in our study besides efficiency.

Based on the above considerations, we developed a method that fulfills these needs. Hence, in this study, we answered the following research question.

How can we solve the problem of sequentially allocating pickers to orders in a collaborative order picking system with performance considerations in terms of pick efficiency and workload fairness?

Methodology

We proposed a deep reinforcement learning method to solve this problem. To train and evaluate this method, we developed a discrete-event simulation. We included uncertainty, disruptions, congestion, and real product data to mimic the complexity of real-world processes.

In our deep reinforcement learning approach, we modeled warehouses using a graph representation, with nodes representing the locations in the warehouse and edges indicating how pickers can travel between these nodes. In this graph representation, nodes contained features that describe the active system status. We used two feature categories, efficiency features and workload fairness features. We trained and evaluated policies for single-objective optimization on efficiency and used a multi-objective algorithm to learn multi-objective policy sets, which out-

line several trade-offs between fairness and performance. The rewards we defined to learn these policies were based on the two objectives. First, we defined a penalty for the time that passed. Second, for fairness, we defined a penalty for increasing standard deviations of the workloads carried by the pickers. The smaller this standard deviation is, the more equal the workloads are and the fairer the solution is.

To enable the training of well-performing policies, we defined our own neural network architectures that capture regional information in the warehouse graphs well. Traditional graph neural networks struggle to capture the long-distance relations in warehouse graphs. Therefore, we proposed a network architecture based on natural warehouse regions formed by aisles. This network architecture uses so-called aisle-embeddings that capture aggregated information from an entire warehouse aisle. By combining this aisle information with the node information, the network learns to value possible actions based on local and regional features. In addition, for multi-objective learning, we implemented a feature separation structure in which the fairness and efficiency representations are only combined after initial high-level representations have been developed.

We trained policies for multiple warehouse sizes and evaluated their performance. We also tested how well the learned policies transfer to different warehouse sizes and different numbers of pickers and vehicles in the warehouses. Good transferability, especially for different pickers and vehicles, is desired for practical adoption.

Results

In the experiments, we found that our approach could learn good policies that offer substantial efficiency and workload fairness improvements over the current methods used by Vanderlande.

First, our policies focussing purely on efficiency consistently achieved improved order completion times over the benchmark policies. For large warehouse instances with 35 aisles and 2800 pick locations, our method achieved efficiency improvements of up to 40% over the method that Vanderlande currently considers, while for smaller warehouses, the improvements were generally over 20%. The learned policies adapted well to changing numbers of pickers and Autonomous Mobile Robots (AMRs) in the warehouses, as well as warehouses with different sizes and product locations. Thus, the policies are generally adaptable and do not need to be retrained or changed for each change in warehouse layouts or product distribution. By visually inspecting several problem instances, we found that our policies keep a better spread of pickers through the warehouse than the benchmark methods, preventing congestion and making more efficient use of the resources. As our policies also handle smaller picker and AMR numbers well, they can be used to not only increase efficiency but also decrease costs while maintaining efficiency. We estimate the potential salary cost savings to be roughly €2.5 million per year for a large warehouse running 16 hours per day, compared to the Vanderlande benchmark.

Aside from efficiency, we showed that our multi-objective method could achieve policies that balance fairness and efficiency well. We showed that we can create sets of policies that outline the trade-offs between the two objectives. Through this method, we offer insights to decision-makers regarding the possible policy options so they can make informed decisions about which policy they desire to use. These policy sets showed that we could significantly improve workload fairness while only sacrificing limited efficiency performance. As a result, the policy sets all contained multiple policies that increased both efficiency and workload fairness compared to the benchmarks. For instance, on the large warehouses, we found a policy that improved the total picking times by 23.6% compared to the current method by Vanderlande while simultaneously reducing the workload standard deviation by 92%. The transferability of these methods to changing parameters such as picker and AMR numbers and warehouse sizes was also good.

In addition, we evaluated how to create more understandable policies. We found that decision tree policies with deep trees concede a small efficiency loss and a slightly bigger fairness loss compared to the reinforcement learning policies. On the other hand, they are interpretable to some extent and provide insights into a guaranteed, fixed set of rules that are applied. Shallow decision trees did not perform well. Thus, decision-makers can decide whether the interpretability of deep decision trees adds sufficient value to compensate for the performance loss.

Conclusions and Recommendations

We conclude that our method generates picker allocation policies that significantly outperform the existing methods in efficiency and workload fairness. Using the multi-objective approach, we can provide policy sets that outline the trade-offs between the two such that decision-makers can make informed decisions on which policy to use based on their preferences.

Therefore, we recommend that Vanderlande integrates our deep reinforcement learning policies into their collaborative order picking concept. To do so, the policies should be considered a core consideration in further developing the collaborative picking concept. We advise starting this integration as soon as possible such that it gets incorporated into the further development of the collaborative picking systems. This allows for a thorough exploration of the hardware and software requirements in the development process, such as selecting which AMRs to use and how to trace the pickers. By starting the development toward real-world implementation, the relevant real-world considerations, barriers, and chances can be understood and managed, and Vanderlande can start its journey toward successfully integrating deep reinforcement learning approaches in real-time decision-making.

Preface

I am proud to present this master thesis, which has been the culmination of my academic journey at the TU/e. Six years ago, I started as a 17-year-old boy with little idea of the great interests that I would develop during my study. Through the past years, I have grown into someone with a passion for real-world problem-solving, optimization, and AI. I am grateful that I have been able to combine these topics in this thesis.

I want to express my gratitude to my university supervisors, dr. Zaharah Bukhsh, dr. Yingqian Zhang, and prof. dr. Mykola Pechenizkiy, for their excellent guidance. I could not have wished for a better group of supervisors. Dr. Bukhsh, our weekly meetings have been of great help to me during the past year. Your knowledge of the field, supportive feedback, and ability to ask the right questions have allowed me to get the best out of this study. Dr. Zhang, I am grateful for your willingness to supervise me. Your passion for research, collaborative spirit, and dedication to pushing the limits have helped me tremendously to shape this project into what it is. Dr. Pechenizkiy, I am thankful for the opportunity to allow me to align this project with my interests. Your insightful questions and coherent comments during our meetings have helped me take a step back and evaluate the core aspects of my study.

I am equally thankful to my company supervisors, Kostas Alogariastos and Kasper Hendriks. You have been of great help in finding my way within Vanderlande. Your expertise, practical insights, and unwavering support have helped me bridge the gap between theory and practice. Due to your trust and flexibility, I have been able to freely explore my ideas and follow my vision for the project.

I also want to express my love and appreciation to my parents, sister, and girlfriend for their unconditional support. I greatly value your encouragement, interest in my work, and willingness to listen whenever I wanted to share the highlights or barriers I encountered. I could not be more grateful.

Furthermore, I want to thank my friends, co-students, and great group partners that I have had over the years. The dream team from the IE bachelor, the OML boys with whom I have experienced great times and memorable meetings during the study-from-home period, the dual-degree friends with whom I have also been able to publish my first scientific article, and my friends from home. You have all contributed to the amazing time that I have had as a student, and I will cherish the moments that we shared.

Finally, I want to thank those who will read this thesis. I hope to share with you not only valuable insights but also an experience of the remarkable journey that this work has been.

Igor Smit

Contents

List of Tables

List of Figures

Acronyms

AGV Automated Guided Vehicle. 20

AI Artificial Intelligence. 28

AMR Autonomous Mobile Robot. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 21, 27, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 54, 55, 57, 58, 60, 63, 64, 70, 71, 72, 73, 74, 79, 80, 83, 88, 89, 90, 91, 92, 93, 97, 98, 99, 113, 114

DDPG Deep Deterministic Policy Gradient. 14, 26

DQN Deep Q-Network. 14, 24, 25, 26, 28

DRL Deep Reinforcement Learning. 2, 4, 5, 11, 13, 14, 15, 16, 19, 22, 23, 24, 26, 27, 28, 30, 31, 35, 36, 37, 42, 46, 49, 51, 55, 56, 57, 58, 60, 62, 63, 64, 65, 66, 67, 70, 73, 75, 76, 77, 79, 80, 81, 82, 87, 88, 90, 91, 92, 93, 96, 97, 98, 99, 100

GCN Graph Convolutional Network. 17, 58, 66, 67

GIN Graph Isomorphism Network. 17, 18, 58, 66, 67

MDP Markov Decision Process. 11, 12, 13, 15, 25

MILP Mixed-Integer Linear Programming. 19, 20, 21, 31, 32, 33, 34, 35, 57, 65, 66, 79

MOMDP Multi-Objective Markov Decision Process. 15, 23, 26, 30, 37, 42, 47, 48

PGMORL Prediction-Guided Multi-Objective Reinforcement Learning. 48, 49, 50, 59, 60, 67, 79, 97

PPO Proximal Policy Optimization. 14, 25, 26, 28, 34, 48, 49, 50, 56, 59, 61, 97

RL Reinforcement Learning. 11, 12, 13, 15, 21, 22, 24, 26, 27, 28, 29, 30, 31, 34, 47, 48, 49

TLO Thresholded Lexicographic Ordering. 24, 25

Chapter 1

Introduction

Order picking is one of the most fundamental and costly processes in logistics systems. In conventional warehouses, up to 90% of time can be spent on picking, and 55% of all operating costs can be attributed to order picking processes (Dukic & Oluic, 2007). Thus, optimizing order picking processes is crucial to achieving cost savings and increased efficiency. If the efficiency is poor, this leads to delayed deliveries and, hence, customer dissatisfaction.

Most warehouses use a picker-to-parts picking system, in which pickers retrieve items from racks and deliver those items to dedicated drop-off locations. These systems, however, are labor-intensive. Besides, pickers spend a large portion of their time traveling from and to drop-off locations. With increased scarcity of personnel as well as more demanding delivery expectations, it is crucial to prevent such waste of valuable human resources. As indicated by Caputo and Pelagagge (2006), Industry 4.0 technology can be used to improve warehouse operations.

To reduce the dependence on human pickers, Autonomous Mobile Robots (AMRs) can be used. As Srinivas and Yu (2022) explained, these have several advantages over human pickers. First, they can navigate freely without needing a predetermined path or operator supervision while also being safe due to their sensors and onboard computers. Second, they can carry higher loads and travel faster than humans while not getting fatigued. Third, they can easily be integrated into existing warehouses and allow for up- or down-scaling. On the downside, the gripping capabilities of robots are inferior to those of humans, and initial investment costs are high.

Collaborative picking systems have been proposed to combine the advantages of AMRs and humans. In these systems, human pickers and AMRs all traverse through the warehouse. The pickers and AMRs must meet at picking locations, where the pickers grab the items and place them onto the AMRs. Once filled, the AMRs can deliver items to the drop-off location while the pickers continue picking with other AMRs. This allows pickers to spend more time on their strength, which is picking, while AMRs perform the traveling in which they have the advantage.

Due to their integration into existing warehouses, collaborative picking systems are promising for all industries that use picker-to-parts warehouses. For example, e-commerce companies can use such systems to continue meeting increasing demands. In addition, warehouses with a high product variety, such as supermarket fulfillment centers, can benefit from collaborative picking solutions since they cannot depend on robots to grab a wide range of items. Hence, in recent years, increasing attention has been directed toward collaborative picking in both industry and research.

Existing works in collaborative picking have focused on several aspects of the problem, ranging from layout design to order batching and human-robot interaction. Picker routing and allocation have also been studied by various authors, such as Srinivas and Yu (2022) and Žulj et al.

(2022). However, these works all consider deterministic scenarios in which no randomness or uncertainty is taken into account. Oppositely, in practice, many uncertainty factors exist, and congestion and delays can influence the process heavily. Additionally, current methods consider limited warehouse sizes with limited numbers of orders, while in real warehouses, thousands of orders are picked. Hence, to facilitate adoption in practice, a method is needed that can handle uncertainty and disruptions and can be used in large warehouse systems.

Another consideration that has not been addressed in current collaborative picking methods, and has received limited attention in operations management in general, is workload fairness. While order pickers have to perform heavy labor, little attention has been paid to how this labor is distributed. Unfair labor distribution can increase the chances of injuries. Besides, it can cause mental strain. Namely, as noted by Pasparakis et al. (2021), human performance is affected by the mental state and their co-workers' work. Hence, unfair workload distribution may not only lead to some workers having increased physical strain, but it may also have adverse effects on the workers' mental state. This, in turn, can negatively impact the workers' well-being, as well as their long-term performance. In addition, having unfair workload distribution can cause some workers to exceed the legally allowed workload limits. Thus, another concern that is not considered in current methods is workload fairness.

In this study, we propose a Deep Reinforcement Learning (DRL) approach to optimize the allocation of human order pickers to picking tasks in collaborative order picking to tackle the mentioned shortcomings of current methods. Previously, DRL has been applied to solve dynamic decision-making problems with uncertainty in a variety of topics, such as the game of Go (Silver et al., 2016), robotic learning (Gu et al., 2017), and machine scheduling (Song et al., 2023). Continuing upon these successes, we show how DRL can be applied to create an online picker optimizer policy that can handle uncertainty and extend to larger warehouses and long picking horizons with thousands of picks.

Whereas regular DRL has been applied to many problems, multi-objective DRL methods have not been adequately tested in practical applications. We extend our solution approach to multi-objective DRL to address both performance and workload fairness and explore their trade-offs. We show which trade-offs can be achieved in various warehouse sizes and with varying picker/AMR-ratios using a discrete-event simulation that we developed. Our study is one of the first to apply and evaluate a multi-objective DRL method in a practical use case. Besides, modeling fairness as an explicit objective within multi-objective DRL and exploring its trade-offs has not been done before.

1.1 Company Background

We conducted this study in collaboration with Vanderlande. Vanderlande is one of the companies interested in collaborative order picking solutions. They are the global market leader for future-proof logistic process automation at airports. In addition, the company is a leading supplier of process automation solutions for warehouses and the parcel market.

The company focuses on optimizing its customers' business processes and competitive positions. Through close cooperation, it strives for the improvement of its operational activities and the expansion of its logistical achievements. Vanderlande's extensive portfolio of integrated solutions, consisting of innovative systems, intelligent software, and life-cycle services, results in the realization of fast, reliable, and efficient automation technology. To ensure that Vanderlande keeps its competitive position in the market, they need to continuously monitor developments in the field and explore the opportunities of new technological advances. This study is part of their continuous effort to develop innovative solutions.

1.2 Research Motivation

Vanderlande is working on the design and optimization of a novel collaborative picking concept to improve the order picking capacity in warehouses. A problem the customers of Vanderlande have experienced in their traditional picker-to-parts warehouses is that pickers spend a large portion of their time walking from and to drop-off locations. To alleviate this, the process has been modernized by using an initial collaborative picking solution. In this system, AMRs drive alongside a picker. These AMRs bring the items to the drop-off location while the picker can continue picking with a new AMR. Using this method, the picker can spend more time picking items and less time walking, increasing the overall capacity of the system.

The drawback of this method is that the picker is bound to a single AMR. More specifically, the picker's efficiency is bounded by the density of the items in the order the AMR is following. In low-density areas, walking between items to pick can still be a large portion of the picker's time. Therefore, another modernization step is performed where the picker can fill multiple AMRs at the same time. This results in a higher density of picks and, thus, a higher pick capacity overall.

In this setting, the AMRs are instructed to follow a fixed route depending on the order plan they have been given. The pickers, on the other hand, are guided by a picker optimizer but have more freedom to move while reaching the specified locations. The challenge is to optimize the route (i.e., sequence of locations) for each picker in the warehouse simultaneously such that a maximum pick capacity is reached. This allows for handling larger demand loads at similar costs or similar demand loads at lower costs for companies. The picker optimizer should be able to adapt to the uncertainty caused by random human behavior and other disturbances in the warehouse, such as congestion.

One seemingly easy way to increase the pick capacity would be to increase the number of AMRs in the system. However, this has several drawbacks. For example, AMRs are expensive and a high number of AMRs in the warehouse can lead to a higher congestion level in the system, resulting in more delays. Therefore, the actual goal is to optimize the pick rate performance per AMR by allocating the pickers to orders that must be picked efficiently.

To illustrate the scenario, a typical snapshot of the warehouse in operation is shown in Figure 1.1. This figure shows that the task allocation optimizer currently allocated a picker to the AMR at the bottom of the figure (A13), whereas there are two unattended AMRs at the top of the figure (A97 and A40). This shows one of the trade-offs that need to be considered: while a higher pick density can likely be achieved when picking from A97 and A40, the time walking there could lower the overall number of picks per hour.

The initial, straightforward objective of a picker optimizer is to have a solution that allocates pickers such that the pick rate is optimized. However, besides this pick rate, it is vital to acknowledge the human pickers in this scenario. Namely, the human pickers have to perform physical labor, in terms of walking and carrying items, which puts physical strain on them. As explained, uneven distribution of this workload over pickers can increase the chances of injuries and cause mental strain that reduces workers' performance. In addition, aligning with the increasing focus on corporate social responsibility, it is vital for organizations to maintain employees' well-being. Therefore, an important additional consideration is the workload distribution. Namely, we want a solution that fairly distributes the picker workload. Hence, in this study, we investigate the trade-offs between fairness and performance and to what extent we can get both good performance and good workload fairness using our proposed picker optimizer.

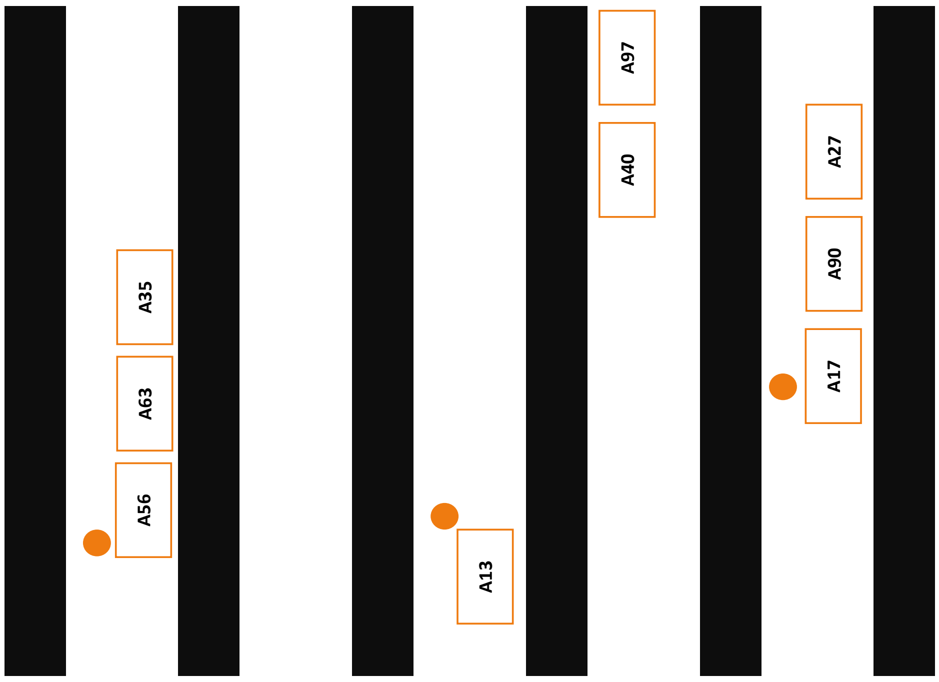


Figure 1.1: Snapshot of a warehouse. Pickers are depicted with orange dots and AMRs as white rectangles.

1.3 Research Questions

The main goal of this study is to develop a method to create a picker optimizer policy for collaborative picking warehouses. We want to explore the trade-offs that can be achieved between pick efficiency and workload fairness. Hence, our main research question is:

How can we solve the problem of sequentially allocating pickers to orders in a collaborative order picking system with performance considerations in terms of pick efficiency and workload fairness?

To answer this question, we consider the following sub-questions:

1. What existing methods exist in collaborative picking literature?
2. What methods in DRL exist for optimizing multi-objective problems?
3. How can fairness be modeled in optimization problems?
4. How to define workload fairness in this collaborative picking problem?
5. How to represent the collaborative picking system using discrete-event simulation?
6. How to design, implement, and evaluate policies that balance performance and fairness objectives?
7. How well does our solution perform in different warehouse settings with different ratios of pickers and AMRs?
8. How to get insights into the behavior and reasoning of the developed solutions?

We will answer the first three sub-questions in our literature review in Chapter 4. In Chapter 5, we will answer the fourth sub-question based on the literature review findings and stakeholders' domain knowledge. We will also discuss the fifth question in Chapter 5. Consequently, we will discuss the design, implementation, and evaluation of the policies in Chapters 5 and 6 to answer the sixth sub-question. In Chapter 7, we will discuss the results, and in Chapter 8, we will perform further analysis to understand the policies, answering the last two sub-questions. In Chapter 9, we will synthesize the results to draw conclusions and answer the main research question.

1.4 Contributions

This study has both practical and scientific contributions. The practical contributions for Vanderlande are:

- A discrete-event simulation model of the collaborative picking system, which can be used for further research. This includes a basic visualization method to verify and understand solution policies.
- A method to develop picker optimizer policies that can optimize the picking performance while balancing the workload fairly.
- A clear overview of the achievable trade-offs between workload fairness and performance in collaborative picking systems that decision-makers can use to make informed decisions about which policy to use.
- An extensive empirical study of the performance of the method compared to baseline methods under varying conditions.

We add to the existing literature in the following way:

- The first online optimization approach that handles stochastic, interactive systems in collaborative picking literature.
- The first method that integrates performance and human workload fairness in collaborative picking with both AMRs and human pickers traveling through the warehouse.
- The first machine learning-based solution in the operations management domain that considers human fairness alongside performance.
- The first study that applies multi-objective DRL to explicitly integrate a fairness objective next to a performance objective with different underlying metrics to generate a non-dominated set of policies outlining the trade-offs.
- A novel neural network architecture using aisle-embeddings to efficiently capture spatial information in warehouses in a less expensive way, and feature separation to effectively handle multiple feature groups.
- An extensive empirical evaluation of multi-objective DRL in a real-world problem.
- An empirical evaluation of a multi-objective DRL method designed for continuous action spaces when adapted to handle a discrete action space.

Chapter 2

Problem Statement

In this study, we developed a method that allocates human pickers to items that must be picked at various pick locations in a collaborative picking environment. The policies must optimize the picking performance while also achieving a fair workload distribution. The method must be capable of handling uncertainty caused by factors such as human travel speed, item unavailability, unexpected picking delays, and aisle congestion. To enhance practical adoption, the desired method can handle changes in the numbers of AMRs and pickers in the system, as these numbers can fluctuate in warehouses. Moreover, to increase applicability in real-life, solutions can also handle changes in the warehouse layout. Namely, sometimes parts of a warehouse may not be available due to maintenance or restocking, or some parts of the layout may be changed and products must be placed in different locations. The practicality of a solution would be limited if such changes significantly affect the usability of the method.

The process of the collaborative system is as follows. A warehouse management system assigns pickruns to AMRs. These pickruns are "shopping lists" for the AMRs that indicate what items must be picked at what locations and in which order. The pickruns are created based on the placed orders and the transport schedule. Once an AMR receives a pickrun, it moves toward its first pick destination. At this location, it waits until a human picker arrives and places the required items on the vehicle. Then, the AMR continues to its next destination, where the same action must occur. This pick can be done by either the same or any of the other human pickers. This process repeats until the AMR completes its pickrun. Then, it must go to a drop-off location where it is unloaded. After unloading, the warehouse management system assigns a new pickrun to the AMR, and the cycle restarts. This happens for many AMRs simultaneously.

In parallel, human pickers are distributed through the warehouse. These pickers are allocated by a picker optimizer. The pickers must walk to their assigned destination, where they meet with the AMRs, collect the items from the shelves, and load them onto the AMRs. Then, they request and receive a new destination and repeat the process. To ensure the productivity of the system, there must be more AMRs than human pickers in the system. This allows the creation of a "swarm" concept in which clusters of AMRs can be fulfilled by pickers. In an ideal system, pickers and AMRs must wait as little as possible for each other.

The initial target customer group of Vanderlande is supermarket fulfillment centers. In this study, we used one of their customers as a basis for the warehouse assumptions. This customer segment has some dissimilarities from other warehouse segments, although many things coincide and the processes are not fundamentally different. For example, these warehouses generally have a wide variety of products. Some products weigh just a kilogram, like boxes with crisps, while others can be ten or fifteen times as heavy, such as large packs with drinks. In addition, the target warehouse type affects the AMR pickruns. Specifically, in these warehouses, AMRs are relatively large but can only carry items for one or two unique customer orders. This means that the AMR pickruns often include a wide variety of products. Therefore, the pickruns generally cover a larger part of the warehouse, whereas, in other sectors, AMRs could be smaller with short pickruns. In Chapter 5, we will describe in more detail the simulation model that we used and the specific assumptions that we considered.

2.1 Mathematical Formulation

To concretize the problem, we define a mathematical program that illustrates the objectives, constraints, and decisions that can be made. We include the essential characteristics to clarify the core choices within our study. In our model, we partially borrowed the notation that Srinivas and Yu (2022) defined and adapted it to fit our use case.

The model that we consider has two objectives. For our study, we define one fairness objective and one performance objective. We define the fairness objective $F(W_1, W_2, \dots, W_{|\mathcal{K}|})$ where F is a fairness function that measures the fairness over the workloads W_k of pickers k. The workloads W_k measure the total mass of the lifted products by the pickers, as we will describe in Chapter 5. As our performance measure, we use the delivery completion time of the last fulfilled order C, which indicates the overall pick efficiency. In this formulation, we assume that we have several AMRs that all have to fulfill one pickrun. In practice, this is a repeating cycle in which AMRs are assigned to a new pickrun after finishing the previous one. However, this assignment is not within our scope, so we ignore this in the problem formulation for illustratory purposes. For a more general model, which includes AMRs fulfilling multiple pickruns, we refer to Appendix A.

Below, we will introduce the relevant sets, parameters, and variables. Then, we will define the mathematical model in Equations 2.1-2.21, after which we will explain the model.

Indices and Sets

$i \in \mathcal{N}$	Set of all items that must be picked. Note that multiple items can have the same location.
$r \in \mathcal{R}$	Set of AMRs.
$k \in \mathcal{K}$	Set of human pickers.

Parameters

M	Sufficiently large positive number.
$\tau^{K}_{i,i'}$	Travel time from location of item i to location of item i' by human pickers.
$\tau^{R}_{i,i'}$	Travel time from location of item i to location of item i' by AMRs.
$\tau^{o,R}_{r,i}$	Travel time from starting location of AMR r to location i.
$\tau^{o,K}_{k,i}$	Travel time from starting location of picker k to location i.
η^{L}_{i}	Time to place item i on an AMR.
$u^{R}_{i,i'}$	1 if AMR r collects item i before item i' in the same trip but not necessarily exactly before item i (i.e., relative precedence), 0 otherwise.
$a^{R}_{i,r}$	1 if AMR r must transport item i, 0 otherwise.
w_i	The workload value of item i.

Variables

Decision Variables

$A_{i,k}$	1 if human picker k is assigned to pick item i, 0 otherwise.
$U_{i,i'}$	1 if item i must be retrieved before item i' by the same picker, 0 otherwise.

Other Variables

$B^{K}_{i,k}$	Time at which picker k arrives at item i.

$F^K_{i,k}$	Time at which picker k is ready to leave the location of item i.
$B^R_{i,r}$	Time at which item i can be placed on AMR r.
$F^R_{i,r}$	Time at which item i has been placed on AMR r and the AMR can leave the location.
C^R_r	Completion time of the pickrun of AMR r.
C	Completion time of the last pickrun.
W_k	The total workload of picker k.

Model Formulation

$$\min C \tag{2.1}$$

$$\max F(W_1, W_2, \ldots, W_{|\mathcal{K}|}) \tag{2.2}$$

subject to

$$\sum_{k\in\mathcal{K}} A_{i,k} = 1 \qquad \forall i \in \mathcal{N} \tag{2.3}$$

$$A_{i,k} - A_{i',k} \leq 1 - (U_{i,i'} + U_{i',i}) \qquad \forall i, i' \in \mathcal{N}, i \neq i', k \in \mathcal{K} \tag{2.4}$$

$$A_{i,k} + A_{i',k} \leq 1 + (U_{i,i'} + U_{i',i}) \qquad \forall i, i' \in \mathcal{N}, i \neq i', k \in \mathcal{K} \tag{2.5}$$

$$B^K_{i.k} \geq \tau^{o,K}_{k,i} - M \cdot (1 - A_{i,k}) \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{2.6}$$

$$\sum_{k\in\mathcal{K}} B^K_{i.k} \geq \sum_{k\in\mathcal{K}} F^K_{i',k} + \tau^K_{i',i} \cdot U_{i',i} - M \cdot (1 - U_{i',i}) \qquad \forall i, i' \in \mathcal{N}, i \neq i' \tag{2.7}$$

$$F^K_{i,k} \geq F^R_{i,r} - M \cdot \left(2 - A_{i,k} - a^R_{i,r}\right) \qquad \forall i \in \mathcal{N}, k \in \mathcal{K}, r \in \mathcal{R} \tag{2.8}$$

$$\sum_{r\in\mathcal{R}} B^R_{i,r} \geq \left(\sum_{r\in\mathcal{R}} F^R_{i',r} + \tau^R_{i',i}\right) \cdot u^R_{i',i} \qquad \forall i, i' \in \mathcal{N}, i \neq i' \tag{2.9}$$

$$B^R_{i,r} \geq \tau^{o,R}_{r,i} \cdot a^R_{i,r} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R} \tag{2.10}$$

$$B^R_{i,r} \geq B^K_{i,k} - M \cdot (2 - A_{i,k} - a^R_{i,r}) \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, k \in \mathcal{K} \tag{2.11}$$

$$F^R_{i,r} = B^R_{i,r} + \eta^L_i \cdot a^R_{i,r} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, \tag{2.12}$$

$$C^R_r \geq F^R_{i,r} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R} \tag{2.13}$$

$$C \geq C^R_r \qquad \forall r \in \mathcal{R} \tag{2.14}$$

$$W_k = \sum_{i\in\mathcal{N}} w_i \cdot A_{i,k} \qquad \forall k \in \mathcal{K} \tag{2.15}$$

$$B^K_{i,k} \leq M \cdot A_{i,k} \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{2.16}$$

$$F^K_{i,k} \leq M \cdot A_{i,k} \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{2.17}$$

$$B^R_{i,r} \leq M \cdot a^R_{i,r} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R} \tag{2.18}$$

$$F^R_{i,r} \leq M \cdot a^R_{i,r} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R} \tag{2.19}$$

$$A_{i,k}, U_{i,i'} \in \{0, 1\} \qquad \forall i, i' \in \mathcal{N}, k \in \mathcal{K}, r \in \mathcal{R} \tag{2.20}$$

$$B^K_{i,k}, F^K_{i,k}, F^R_{i,r}, B^R_{i,r}, C^R_r, C \geq 0 \qquad \forall i \in \mathcal{N}, k \in \mathcal{K}, r \in \mathcal{R} \tag{2.21}$$

Constraint 2.3 ensures that each item is picked by just one picker. Constraints 2.4 and 2.5 define the relative order of two items that are picked by the same picker. They ensure that one item is ordered before the other if the same picker picks both items. Constraints 2.6 and 2.7 indicate that the time at which a picker can start to retrieve an item from a shelf is the maximum of the travel time from its starting position to the location or the time at which the picker left the previous location plus the travel time. In turn, a picker can only leave a pick location once the item has been placed on the associated AMR, as indicated by Constraint 2.8. Constraints 2.9, 2.10, and 2.11 describe the constraints on the time at which an AMR is ready for an item to be placed on it. Concretely, Constraint 2.9 describes that an AMR is only available after it has traveled from the previous pick location to the current one. Constraint 2.10 describes the start of the system, indicating that an AMR can only be ready to perform a pick after its initial travel time from its starting position to the pick location has passed. Constraint 2.11 relates the

AMR beginning time to the picker availability. Specifically, an AMR can only be loaded once a picker is at the pick location to pick the item. Then, an AMR has been loaded and can leave the location after the associated pick time has passed, which is indicated in Constraint 2.12.

Following these constraints that describe the system logic, Equations 2.13, 2.14, and 2.15 relate the system to the objectives. In Equation 2.13, we bound the completion time of an AMR pickrun to the time at which the AMR has finished its last pick. Then, in Equation 2.14, we define the efficiency objective value, being the completion time of the last fulfilled order. In Equation 2.15, we set the total workload of a picker equal to the sum of the workloads of each pick. Using a specific fairness measure F, these are combined into a fairness objective value in Equation 2.2.

Lastly, Equations 2.16-2.21 enforce proper functioning of the mathematical formulation while not describing system characteristics. Equations 2.16 and 2.17 ensure that the beginning and finishing times of picker actions related to an item are only set when the picker is assigned to pick this item. Similarly, Equations 2.18 and 2.19 enforce this for AMRs. Lastly, Equations 2.20 and 2.21 describe that the decision variables $A_{i,k}$ and $U_{i,i'}$ are binary and the time-related variables must be non-negative.

To highlight the task of our picker optimizer, we distinguished the decision variables from the other variables. These indicate that our picker optimizer solution will directly influence the picker variables. Concretely, in our solution, we will learn a policy that assigns the pickers to pick locations at each step. By doing so, the policy determines the assignment variables $A_{i,k}$ and $U_{i,i'}$. Based on these decisions and the occurrences in the environment, the other variables regarding start and finish times, completion times, and workload can be computed.

2.2 Assumptions and Scope

In this study, we focused on the allocation of pickers to orders that must be picked. Hence, we assumed that the batching and releasing strategies of the orders are fixed. Besides, the routes that AMRs follow through the warehouse are fixed. We will outline the assumed strategies in Section 5.1. We focused on one performance objective regarding the picking capacity and one fairness objective considering workload. Our performance objective was to minimize the total time to fulfill a predetermined set of picks. The fairness objective was to balance the lifting workload of the human pickers. We required a solution that ideally handles varying picker and AMR numbers. Besides, we explored how the optimizer adapts to different warehouse sizes.

In the above model formulation, a deterministic environment is defined. However, it is important to remember that we assumed a stochastic environment in our use case. The parameters such as travel times and item retrieval and placement times are stochastic. In turn, this also makes the other variables uncertain, and therefore, deterministic optimization methods are not preferred. Moreover, in our use case, we considered a scenario where many pickruns must be completed over the course of several hours. These pickruns are known in advance and assigned to AMRs online, as we will describe in Section 5.1. The goal is to complete all orders in the pre-specified set of pickruns as quickly as possible. Thus, the completion time of the last fulfilled pickrun indicates the performance.

Another relevant performance objective in order picking systems is tardiness (i.e., the number of picks that are too late). However, tardiness is primarily affected by the order batching and releasing strategies. In addition, tardiness is a less stringent requirement in our wholesale warehouse segment than in, for example, e-commerce. Hence, we did not consider this. An alternative workload specification that we could consider is walking distance. However, walking puts less strain on workers, is harder to notice physically by pickers, and is not subject to strict ergonomic regulations like lifting workloads. Therefore, we did not consider walking distances in fairness.

Besides picker allocation, batching, and sequencing, many factors influence the warehouse picking capacity. Factors such as the types of AMRs, the aisle layout, the product distribution, and having uni- or bi-directional aisle traffic can all influence performance. We did not aim to address these strategic choices.

Chapter 3

Background

This chapter outlines the fundamentals of DRL, multi-objective DRL, and learning on graphs. These are relevant concepts that are fundamental to the solution methodology that we propose in this study. We will first introduce the standard framework of Reinforcement Learning (RL) and how optimal policies are found in Section 3.1. Then, in Section 3.2, we will extend this framework to incorporate multi-objective decision-making. Lastly, we will explain the fundamentals of graph theory and graph neural networks in Section 3.3. For a more elaborate background, we refer to the cited literature.

3.1 Deep Reinforcement Learning

As mentioned in Chapter 2, DRL, which is a technique that combines RL with deep learning, offers solution techniques that can solve sequential decision-making problems in uncertain, interactive environments. In recent years, DRL has been used to solve many complex interactive problems, such as playing the game of Go (Silver et al., 2016), robotic learning (Gu et al., 2017), or machine scheduling (Song et al., 2023). In this section, we will first introduce the basic concepts of RL that are fundamental to any RL problem, based on the work by Sutton and Barto (2018). We will explain how agent-environment interaction works and how good solutions are defined. Then, we will provide a basic description of the general streams of DRL algorithms.

3.1.1 Agent-Environment Interaction

The core idea underlying RL, as opposed to supervised or unsupervised learning, is that no straightforward dataset with features and target values is used to train a model. Instead, an agent learns through agent-environment interaction. This interaction is illustrated in Figure 3.1. In this process, at time t, an agent is in a so-called state S_t, resembling the current situation, and takes an action A_t. Based on this action, the agent receives a reward R_{t+1} and continues toward a new state S_{t+1}. Then, this process can be repeated for time $t + 1$, and so on. Based on these interactions, the agent should learn which actions to take.

3.1.2 Markov Decision Process

The Markov Decision Process (MDP) provides a formalized framework to model the illustrated interaction process. It captures the information regarding states, actions, rewards, and transitions. Based on the definition by Silver (2015), an MDP can be defined as follows.

Definition 3.1. *An MDP is defined as a tuple* $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$

- $\mathcal{S}$ *is a set of states* s
- $\mathcal{A}$ *is a set of actions* a
- $T : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ *is the transition function, indicating the transition probability*

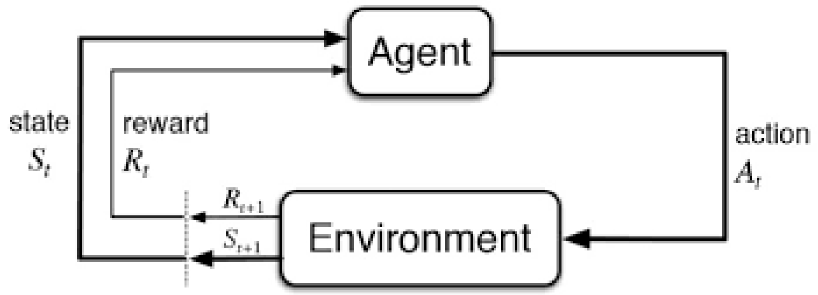


Figure 3.1: The agent–environment interaction in RL (Sutton & Barto, 2018, p. 48).

$\mathbb{P}(S_{t+1} = s'|S_t = s, A_t = a)$

- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is the reward function*
- $\gamma \in [0, 1]$ *is a discount factor, indicating how much future rewards should be discounted*

In this definition, we use a fundamental assumption. Namely, we assume the Markov Property. This property implies that the future of the process does not depend on any past state but only on the current one.

Definition 3.2. *A state S_t satisfies the Markov property iff:*

$$\mathbb{P}(S_{t+1}|S_t) = \mathbb{P}(S_{t+1}|S_t, S_{t-1}, \ldots, S_1)$$

Thus, when defining an MDP, it should be ensured that all necessary information regarding the past is captured in the current state.

3.1.3 Finding an Optimal Policy

As explained, an agent should learn which actions to take from its interaction in the MDP. To do so, we define its objective to maximize the expected cumulative future reward G_t. The most straightforward approach to finding the cumulative future reward is summing all obtained rewards.

$$G_t \coloneqq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

This can be suitable when a task is episodic, which means it naturally breaks into independent sub-sequences of a finite length having the last timestep T. However, when a task is not episodic or has many timesteps with high uncertainty of long-term rewards, it may be better to use the discounted cumulative future reward.

$$G_t \coloneqq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Here, γ is the discount factor in Definition 3.1 of the MDP. When γ approaches 1, the agent will focus more on the long-term rewards, whereas with a low γ it will focus more on immediate rewards.

Based on this definition of cumulative future rewards, we can formally describe what we consider an optimal policy. To do so, define the state-value function $v^{\pi}(s)$ of a state s while following a policy π.

$$v^{\pi}(s) \coloneqq \mathbb{E}_{\pi}[G_t|S_t = s] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right]$$

This function indicates the expected accumulated return when following a policy π after starting in state s. An essential property of this value function, which is frequently used in RL approaches, is that it can be decomposed into the expected immediate reward and the expected value function of the next time step. This is shown via a recursive equation called the Bellman equation.

$$\begin{aligned} v^{\pi}(s) :=& \mathbb{E}_{\pi}[G_t|S_t = s] \\ =& \mathbb{E}_{\pi}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ =& \mathbb{E}_{\pi}[R_{t+1} + \gamma v^{\pi}(S_{t+1})|S_t = s] \end{aligned}$$

Similar to the state-value function, we can define an action-value function $q^{\pi}(s, a)$. This function defines the expected cumulative return after starting in state s, executing action a, and following a policy π afterward.

$$q^{\pi}(s, a) := \mathbb{E}_{\pi}[G_t|S_t = s, A_t = a] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a\right]$$

Using these value functions, we can define what we consider an optimal policy. Namely, an optimal policy is a policy that has an expected cumulative return that is greater than or equal to the expected cumulative return of all other policies for all states. Formally, this can be defined as follows.

Definition 3.3. *Consider the partial ordering:*

$$\pi \geq \pi' \text{ if } v^{\pi}(s) \geq v^{\pi'}(s), \forall s \in \mathcal{S}$$

Then, an optimal policy π^* exists such that:

$$\pi^* \geq \pi, \forall_{\pi}$$

Such optimal policy achieves the optimal state-value function and action-value function. In high-dimensional, stochastic environments, achieving an optimal policy may be challenging. In such cases, the goal is to achieve a policy that is as good as possible.

3.1.4 Deep Reinforcement Learning Approaches

To find optimal policies in an MDP, a variety of RL algorithms have been developed. The algorithm that has been fundamental in both RL and as a basis for DRL is the so-called Q-learning algorithm (Watkins & Dayan, 1992). In Q-learning, an estimate $\hat{Q}(s, a)$ of the optimal action-value function is learned for each state-action pair. This is done by storing a so-called Q-table of $|\mathcal{S}|$ rows and $|\mathcal{A}|$ columns (with $|\cdot|$ indicating the cardinality of the sets) and using the following iterative update rule based on the latest experience tuple $\langle s, a, r, s'\rangle$ containing the state s, selected action a, obtained reward r and new state s'.

$$\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a))$$

Based on the Q-table, a policy, such as the epsilon-greedy policy, can be applied to select actions. This method works well on relatively small problem instances. However, it requires the storage of a value for each state-action pair. For problem instances with a large or unbounded state space, this table cannot be stored, and an enormous amount of interactions is needed to estimate the values accurately.

To overcome this, DRL, in which deep learning and RL are combined, has been proposed. We will give a short overview of the existing DRL methods as a background for the next chapter. For more details regarding the individual algorithms and their pros and cons, we refer to the amplitude of existing literature (e.g. Silver (2015) and cited papers).

The first DRL algorithm, which is the Deep Q-Network (DQN) algorithm (Mnih et al., 2015), uses a neural network that acts as a function approximator for the Q-table. Following this work, several algorithms have extended the DQN structure, like the dueling DQN (Z. Wang et al., 2015) and double DQN (Hasselt et al., 2016). These methods, in which the value function is approximated, are called value-based methods.

Aside from value-based methods, there are also policy-based and actor-critic methods. In policy-based methods, a policy is directly learned instead of learning a value function. Thus, a policy network outputs which action should be taken. It does so without estimating the action-value function and does not require a selection policy like the epsilon-greedy policy. Actor-critic methods also use a policy network for action selection but utilize a value network in the learning algorithm to combine the advantages of action-based and pure value-based methods. The most important methods that use a policy network are Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015).

To directly update a policy network π_θ with parameters θ, PPO uses a loss function based on the action probabilities of the network. To do so, the algorithm alternates between collecting samples and updating the policy using the empirical estimates from these samples. The loss function $\mathcal{L}^{CLIP}$ used to update the network is as follows.

$$\mathcal{L}^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) \right]$$

Here, $\hat{\mathbb{E}}_t$ indicates the empirical expectation based on the collected sample batch, $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ describes the ratio between the probabilities of the old and new policy of selecting an action a_t in state s_t at time t, and $\hat{A}_t$ is an estimator of the advantage function at time t, indicating how good the action taken at time t was. Thus, the loss tries to maximize the probability of taking good actions and minimize the probability of taking bad actions. To ensure the policy does not change too drastically and forgets what it has learned, the ratio is clipped using the hyperparameter ϵ, limiting the loss. In actor-critic variants of PPO, the advantage function $\hat{A}_t$ is also estimated by a neural network that is updated during the learning process. To handle the exploration-exploitation trade-off within the PPO algorithm, the loss function includes an entropy term. This term measures the spread of the probabilities. This is incorporated as follows.

$$\mathcal{L}(\theta) = \hat{\mathbb{E}}_t \left[\mathcal{L}^{CLIP}(\theta) + c_{ent} \cdot S[\pi_\theta](s_t) \right]$$

Here, c_{ent} is the entropy coefficient, which determines the weight of the entropy within the loss function, and $S[\pi_\theta](s_t)$ represents the entropy measure. By setting c_{ent} to a small value, the algorithm will focus more on exploitation instead of exploration when the clipping loss has been reduced.

Whereas value-based methods can only be applied in discrete action spaces, DDPG can be applied for continuous action spaces and PPO for both. A discrete action space indicates a finite set of actions, like going left and right. In contrast, a continuous action space indicates real-valued actions, such as the angle in a coordinate space that is any number between 0 and 360.

Lastly, next to value- and policy-based algorithms, there exist so-called model-based DRL algorithms. These learn a model of the environment using methods like Monte-Carlo Tree Search, while the previous model-free algorithms directly learn a value or policy network. An example of a model-based approach is AlphaGo (Silver et al., 2016). Model-based methods are generally more sample efficient, as they know a model of the process. However, they have to learn a model of the system that is often large, which can be very demanding and computationally complex for complicated tasks. In general, less research has been conducted on model-based DRL, and the model-free techniques are better developed for generic use.

3.2 Multi-Objective Reinforcement Learning

In regular RL and DRL methods, a single objective is optimized. However, in many real-life applications, like ours, multiple objectives must simultaneously be considered in optimization. Different approaches are used to handle these often conflicting objectives. This section will introduce the relevant background regarding multi-objective optimization in RL.

3.2.1 Multi-Objective Markov Decision Process

To handle multiple objectives in RL, we need a framework like the MDP that can handle multiple rewards. This multi-objective sequential decision-making problem can be formalized as the Multi-Objective Markov Decision Process (MOMDP), as defined by, amongst others, Hayes et al. (2022).

Definition 3.4. *An MOMDP is defined as a tuple* $\langle \mathcal{S}, \mathcal{A}, T, \boldsymbol{\mathcal{R}}, \gamma \rangle$

- $\mathcal{S}$ *is a set of states* s
- $\mathcal{A}$ *is a set of actions* a
- $T : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1]$ *is the transition function, indicating the transition probability* $\mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a)$
- $\boldsymbol{\mathcal{R}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ *is the reward function, with* $d \geq 2$ *the number of objectives*
- $\gamma \in [0,1]$ *is a discount factor, indicating how much future rewards should be discounted*

Thus, an MOMDP is equivalent to an MDP, except for the reward function. In an MOMDP, instead of a single scalar reward, a vectorized reward is obtained, with one element for each objective. Using this formulation, the multi-objective state-value function $\boldsymbol{v}^\pi(s)$ follows directly from the single-objective version $v^\pi(s)$.

$$\boldsymbol{v}^\pi(s) := \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k \boldsymbol{R}_{t+k+1} | S_t = s \right]$$

Using this multi-objective value function, one would ideally define an optimal policy similar to Definition 3.3. However, it is not possible to establish a complete ordering when having multiple objectives since one policy may be better for one objective while the other is better for another objective.

3.2.2 Optimal Policies in Multi-Objective Optimization

The most straightforward method for handling multiple objectives is to transform them into a single objective that captures the relative importance of each objective. This single objective depends on the utility function $u : \mathbb{R}^d \to \mathbb{R}$ of the user, which translates the rewards into a single reward. This function can be linear (i.e., a weighted sum of the two rewards) or non-linear. In this manner, the problem can be solved using traditional single-objective methods. However, this requires a known utility function. This may be possible when, for example, rewards relate directly to monetary costs. On the other hand, Hayes et al. (2022) and Roijers et al. (2013) show six distinct scenarios in which it is undesirable, impossible, or infeasible to use a single-objective approach. A single-objective approach is a good option only if a utility function is known, certain, and cannot change.

When not using a single-objective method, another way of determining which solutions are good is necessary. To this end, the Pareto Front has been crucial in multi-objective optimization. The Pareto Front is the set of non-dominated solutions. This means that for each solution (policy in an RL problem) on the Pareto Front, no other solution has a better value for all objectives. We adopt a similar formal definition as given by Hayes et al. (2022).

Definition 3.5. *Given a monotonically increasing utility function u, a set of policies Π, and the expected state-value function $\boldsymbol{v}^\pi := \mathbb{E}_{S_t}[\boldsymbol{v}^\pi(S_t)]$ over the distribution of states, we define the Pareto Front PF as:*

$$PF(\Pi) := \{\pi \in \Pi | \nexists \pi' \in \Pi : \boldsymbol{v}^{\pi'} \succ_p \boldsymbol{v}^\pi\}$$

Here, $\succ_p$ indicates the Pareto dominance:

$$\boldsymbol{v}^\pi \succ_p \boldsymbol{v}^{\pi'} \Leftrightarrow (\forall i : \boldsymbol{v}_i^\pi \geq \boldsymbol{v}_i^{\pi'}) \wedge (\exists i : \boldsymbol{v}_i^\pi > \boldsymbol{v}_i^{\pi'})$$

In general, multi-objective optimization methods should aim to find the Pareto Front. This allows end-users to gain a clear view of the trade-offs of each policy, enabling informed decision-making. Since multiple policies can lead to the same value functions, we can reduce our search to finding the Pareto Coverage Set, which is a subset of the policies in the Pareto Front, such that they cover all value functions from the Pareto Front. Hence, the final goal of multi-objective optimization methods is to find a non-dominated set of solutions (policies) that represents the Pareto Coverage Set as well as possible.

3.3 Learning on Graphs

Graphs are a fundamental part of our DRL modeling approach. Therefore, this section introduces the fundamentals of graph theory and graph neural networks. We largely adapted this background from Bacciu et al. (2020).

3.3.1 Graph Fundamentals

We formally define a graph in Definition 3.6.

Definition 3.6. *A graph G is defined as a tuple $\langle \mathcal{V}_G, \mathcal{E}_G, \mathcal{X}_G, \mathcal{A}_G \rangle$*

- *$\mathcal{V}_G$ is a set of nodes*
- *$\mathcal{E}_G = \{(u, v) | u, v \in \mathcal{V}_G\}$ is a set of edges*
- *$\mathcal{X}_G \subseteq \mathbb{R}^d$ is a set of node features, with $d \in \mathbb{N}$*
- *$\mathcal{A}_G \subseteq \mathbb{R}^{d'}$ is a set of edge features, with $d' \in \mathbb{N}$*

Thus, a graph $G = (\mathcal{V}_G, \mathcal{E}_G, \mathcal{X}_G, \mathcal{A}_G)$ is formed by a set of nodes $\mathcal{V}_G$, edges $\mathcal{E}_G$, the node information $\mathcal{X}_G$, and edge information $\mathcal{A}_G$. The nodes in a graph can represent people, locations, or any other entities. The edges represent the relations between the nodes. In an undirected graph, these edges are unordered, meaning $\mathcal{E}_G = \{\{u, v\} | u, v \in \mathcal{V}_G\}$. In a directed graph, the edges have an orientation from one node to the other, meaning $\mathcal{E}_G = \{(u, v) | u, v \in \mathcal{V}_G\}$. An undirected graph can also be considered a directed graph in which each edge $\{u, v\}$ is replaced by the directed edges (u, v) and (v, u). An efficient method of representing the edges is an adjacency matrix. This is a binary square matrix $A \in \{0, 1\}^{|\mathcal{V}_G| \times |\mathcal{V}_G|}$ in which 1 indicates that nodes are connected and 0 that there is no edge between the nodes. In the case of an undirected graph, this adjacency matrix is symmetric.

To increase the practical applicability of graphs, they are often enriched with node information $\mathcal{X}_G$ and edge information $\mathcal{A}_G$. For each node, the node feature vector $\mathbf{x}_v \in \mathcal{X}_G$ describes the information of the node. In a social network graph, for example, this can be information such as the age or gender of a person. Similarly, an edge feature vector $\mathbf{a}_{u,v} \in \mathcal{A}_G$ describes additional information regarding the relations between the nodes. In the social network graph example, this can be information such as how long people have known each other. One can choose to omit node or edge information in the graph. Intuitively, this means that all nodes or edges are considered equivalent.

The neighborhood of a node v describes the nodes connected to the node via a directed edge. Thus, the neighborhood is defined as $\mathcal{N}_v = \{u \in \mathcal{V}_G | (u, v) \in \mathcal{E}_G\}$. This can be expanded to a

k-hop neighborhood, the set of nodes that can be reached within k edges from the node. This is an important principle that is used in graph neural networks.

3.3.2 Graph Neural Networks

The goal of graph neural networks is to learn so-called node-embeddings for all nodes in a graph. The node-embeddings represent the state of a node from which value can be extracted. They can be used for further operations, such as passing them through feed-forward neural networks to create node values or aggregating them to find graph-embeddings.

The node-embeddings are generally calculated in an iterative way, using consecutive graph neural network layers. Thus, the node-embedding $\mathbf{h}_v^l$ at iteration l is used as input to compute the embedding $\mathbf{h}_v^{l+1}$ of iteration $l+1$. The initial state $\mathbf{h}_v^0$ is given by the node feature vector $\mathbf{x}_v$. Using K iterations, the final embedding $\mathbf{h}_v^K$ is used for further processing.

The core requirements of graph neural networks are that they can handle different topologies (i.e., different sizes and shapes) and that they are, depending on the goal, permutation invariant (i.e., when input is permuted, the output stays the same) or permutation equivariant (i.e., when input is permuted the output is permuted in the same way). This is achieved by locally computing at the node level. The most straightforward method is to use an invariant feed-forward network in which each node is passed through a simple multilayer perceptron without considering the other nodes. However, various layers have also been developed to incorporate the node neighborhoods. These methods apply permutation invariant functions to the nodes and their neighborhoods. These permutation invariant functions ensure that node representations can be learned regardless of the ordering of the nodes and the number of nodes.

All major graph neural networks use the concept of message passing to fulfill the previously mentioned requirements. In this concept, in each layer, information is passed between connected nodes and aggregated. The core formula for message passing layers is as follows.

$$\mathbf{h}_v^{l+1} = \phi^{l+1}\left([\mathbf{h}_v^l, \Psi(\{\psi^{l+1}(\mathbf{h}_u^l) | u \in \mathcal{N}_v\})\right)$$

Here, ψ^{l+1} is a function that is applied to the node-embeddings of each of the neighbors of a node. This constructs a “message” from each of these nodes. Ψ is a permutation invariant function, which can be a sum, mean, product, or any other function that does not depend on the ordering of the input. This combines the messages from the neighbors into an aggregated “message vector”. Lastly, ϕ^{l+1} is a function that takes the current node-embedding and the message vector and combines them into a new node representation. Thus, each node in the neighborhood sends a message to the considered node. These messages are combined into an aggregated message and then processed with the current node-embedding to create a new node-embedding. The functions ϕ^{l+1} and ψ^{l+1} can be any function, but in graph neural networks, often one or both functions are learned using multilayer perceptrons or a learnable weight matrix. By stacking k message passing layers, information from the k-hop neighborhood of a node can be captured. However, with more layers, more parameters must be learned, and information gets more diffused. Hence, the number of message passing layers that can be effectively used is limited.

Two examples of implementations of message passing networks are the Graph Convolutional Network (GCN) (Kipf & Welling, 2016) and Graph Isomorphism Network (GIN) (K. Xu et al., 2018). GCN implements a relatively simple message passing layer. The formula is as follows.

$$\mathbf{h}_v^{l+1} = \sigma\left(\mathbf{W}^{l+1} \sum_{u \in \mathcal{N}_v} \mathbf{L}_{u,v} \mathbf{h}_u^l\right)$$

Here σ represents a non-linear activation function, $\mathbf{L}$ is a normalized graph Laplacian, which normalizes the node features by the degrees of the nodes, and **W** is a weight matrix that is learned in training.

GIN offers a more flexible message passing layer at the cost of more learnable parameters. Namely, a multilayer perceptron combines the neighborhood messages with the local node-embedding. The formula is as follows.

$$\mathbf{h}_v^{l+1} = MLP^{l+1}\left((1+\epsilon^{l+1})\mathbf{h}_v^l + \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^l\right)$$

In this formula, MLP indicates the multilayer perceptron and ϵ is a trainable parameter specifying the weight of the current node-embedding in relation to the neighborhood message.

As explained, using the outcome embeddings of graph neural networks, either node information or a graph-embedding can be retrieved. To generate a graph-embedding, a final permutation invariant operation is required. Concretely, the following formula can be applied to generate a graph-embedding from the node-embeddings in layer l.

$$\mathbf{h}_G^l = \Psi\left(\{f(\mathbf{h}_v^l) | v \in \mathcal{V}_G\}\right)$$

Again, Ψ is a permutation invariant operator such as the sum, mean, or product. f can be a function that transforms the node-embeddings, but generally, the node-embeddings are aggregated without applying another function to them (i.e., f is the identity function).

Chapter 4

Literature Review

This chapter outlines the current work related to our study. To do so, we performed a semi-systematic literature review, as characterized by Snyder (2019). As Snyder (2019) explained, a semi-systematic literature review is a good option when literature from different streams of research must be combined to synthesize all information related to a research problem. This fits well with our problem, in which we must synthesize information from collaborative picking, DRL, and fairness.

In this chapter, we will first present the current works in collaborative picking. We will give an overview of the optimization techniques used and also show to what extent the human factor is considered and how workload has been defined. Then, we will discuss the current methods in multi-objective DRL. Consequently, we will show how DRL has been applied in problems related to collaborative picking. Lastly, we will discuss how fairness is defined and outline how fairness is currently incorporated in DRL and optimization.

4.1 Collaborative Picking

In this section, we will first discuss the existing collaborative picking methods on both optimization and the incorporation of the human factor. Then, we will shortly touch upon existing techniques to measure workload.

Collaborative order picking is a relatively new concept introduced only a few years ago. In their review of robotized warehousing, Azadeh et al. (2019) were the first to introduce collaborative order picking in operations management literature. Since then, various works have been published regarding optimizing collaborative order picking and human interactions in these scenarios.

In one of the first papers related to collaborative order picking, Lee and Murray (2019) evaluated warehouse layouts. In this paper, they considered a pick, place, and transport routing problem in which picker robots and transport robots collaboratively pick orders. This setting is similar to our problem if we replace the picker robots with human pickers. Another difference is that transport robots can carry items from many customers simultaneously, which is not the case in our project. This paper introduced a Mixed-Integer Linear Programming (MILP) model for this scenario. Then, they explored which designs are best by solving the program on small problem instances for different warehouse layouts. They did not assume any randomness. Additionally, no approach was proposed for guiding the robots in practice, and no heuristics were proposed to solve larger problem instances. Hence, the MILP formulation of the problem is valuable, but for order picking implementations, new methods are needed.

Zou et al. (2019) were the first to propose an optimization approach for collaborative order picking. In their paper, they considered a zone-based collaborative order picking problem. In their considered warehouse, each picker is assigned to a specific zone and handles orders within those zones, while Automated Guided Vehicles (AGVs) can traverse through zones. The paper proposed an approach to simultaneously optimize order batching, picker zone scheduling, and AGV routing. They defined an MILP model. To solve this, they proposed a two-stage heuristic. This heuristic first determines the order batching, after which the zone scheduling and AGV routing problems are solved using a heuristic rule with neighborhood search.

While Zou et al. (2019) assumed a zone-based picking strategy, they did not explore whether this is better than a non-zoned idea. To test this, Azadeh et al. (2020) performed a comparative study on zoning strategies in collaborative order picking. They focused on two main strategies, being no zoning and progressive zoning. They developed a queuing model to estimate the effect of different zoning strategies on the performance of their order picking system. Then, they suggested a Markov decision model to investigate how higher pick performance can be achieved by dynamically switching between strategies. They found that progressive zoning is helpful in e-commerce warehouses with small order sizes, and no zoning is suitable for store replenishment warehouses with large order sizes. A dynamic strategy can be used for heterogeneous order sizes. This aligns with the strategy of Vanderlande to use a concept without zoning in the supermarket fulfillment warehouses.

Ghelichi and Kilaru (2021) targeted decision-making regarding the creation of collaborative order picking warehouses. They developed an analytical model that helps determine which collaborative order picking solution in e-commerce should be applied in which case, based on the required number of pickers and AMRs. Here, they considered Last-Mile-Delivery and Meet-in-Aisle solutions. In addition to the theoretical analysis, they performed an experimental analysis based on a simulation environment. In general, collaborative solutions outperformed manual picking in facilities with high throughput rates. Meet-in-Aisle solutions generally performed better when the facility is big and the cluster size is relatively small. On the other hand, the Last-Mile-Delivery solution fits better for facilities that need to process high throughput rates. The Vanderlande use case considers the first scenario and, hence, the used collaborative picking approach with pickers and AMRs both moving through the warehouse fits well.

Following these initial works, several works have recently aimed at optimizing the performance in collaborative order picking problems. These works are the most related to our study. First, Löffler et al. (2022) studied picker routing in AGV-assisted picker-to-parts order picking in single-block (i.e., without cross-aisles), parallel-aisle warehouses. They developed an exact polynomial time algorithm to minimize the total traveled distance for cases with fixed picking sequences. This dynamic programming approach extends the work by Ratliff and Rosenthal (1983). In addition, they showed that the scenario in which sequences must be decided is NP-hard and offered a greedy heuristic. They proposed several MILP formulations and showed how they can be transformed into traveling salesman problem instances. They performed an experimental study and showed good performance of their methods, as opposed to traditional order picking.

Žulj et al. (2022) were the first in collaborative order picking to minimize tardiness. This paper formalized what they call the AMR-assisted order picking problem. In this setting, human pickers pick items within a specified region in a single-block warehouse and drop them off at handover locations. There, AMRs collect the items and transfer them to the central warehouse depot. They also formulated an MILP model. This model decides on the grouping of customer orders into batches, the assignment of these batches to AMRs, and the processing sequence in which these batches are handled by the order pickers and the AMRs. The objective is to minimize the total tardiness of all customer orders. To solve this problem, they developed a heuristic method using adaptive large neighborhood search (Ropke & Pisinger, 2006) and an adaptation of the NEH heuristic (Nawaz et al., 1983). This heuristic method performed well in experimental settings compared to exact solver benchmarks.

Srinivas and Yu (2022) also focused on minimizing tardiness. They considered a problem in which both pickers and AMRs can freely move through the warehouse. This collaborative human-robot order picking system is most similar to our use case. They integrated order batching, sequencing, and picker-robot routing in their method. Again, their method used an MILP model. They proposed a restarted simulated annealing meta-heuristic with adaptive neighborhood search to improve exploration and exploitation. This method showed near-optimal results for several problem instances.

The latest paper by Xie et al. (2022) regarding optimization in collaborative order picking extends the scope to warehouses with a mixed-shelf storing policy. For this approach, they considered collaborative order picking with picking zones for human pickers. They proposed two MILP formulations, using the three-index formulation and two-commodity network flow formulation known from the capacitated vehicle routing problem. Then, they suggested a variable neighborhood search heuristic to solve the problem.

In short, all these optimization approaches are offline methods. They declare a deterministic problem through an MILP formulation and use a heuristic to solve larger instances. They do not consider uncertainty, unexpected delays, or other sources of stochasticity. These omitted factors are precisely the elements that we require for the solution to be functional.

4.1.1 Human Factor in Collaborative Picking

Aside from these optimization approaches, some works have explored the human factor within several scenarios related to collaborative picking.

First, Pasparakis et al. (2021) noted that human workers are often impacted by their coworkers and their performance. Hence, they investigated how human pickers are influenced by collaborative picking robots. They tested the effect of two strategies where humans continuously walk alongside the robot. These two approaches are the human leading and human following way. The paper found that a human-leading approach allows superior collaborative order picking productivity compared to human following. In contrast, a human-following method allows for greater collaborative order picking accuracy than a human-leading setup.

Second, M. Zhang et al. (2021) developed a simulation model to inspect the human energy expenditure, as well as the cost per pick and average throughput in a hybrid order picking system. To measure the energy expenditure, the work by Garg et al. (1978) was used. This system, however, was not a collaborative order picking system. Instead, in this hybrid system, human and robot pickers independently pick orders in the warehouse.

Lorson et al. (2022) explored the interactions between humans and robots in warehousing. The article aims at helping operations management researchers to identify the potential effects of human behavior. To this end, they developed a theoretical framework of the types of interactions and their implications. They advocate for better integration of human factors in operations management research and human-robot systems, as this is underrepresented in current literature.

In collaborative picking solutions, only one method has been proposed that explicitly aims at the human factor. This method, proposed by Niu et al. (2021), uses a multi-agent RL approach in a robotic mobile fulfillment system. In this system, robots perform all picking tasks in the warehouse and drop items off at working stations where humans handle the items. The work aims to assign the robots to one of the working stations such that the performance is good, but human discomfort is not too high. All other decisions, such as choosing which item to pick, were excluded, and they considered deterministic behavior. To this end, a reward function was specified based on a discomfort function defined by Larco et al. (2017). Then, an assignment policy was learned using single-objective RL. The same approach was used by Niu and Schulte (2021), but using an objective of having sufficient breaks instead of discomfort. Thus, this is one of the few methods that handle a human factor simultaneously with an objective. However, the setting is different, and a rather simplistic single-objective method in a deterministic environment was used. Besides, only two human workstations were considered.

To summarize this section, existing work in collaborative order picking has focused on several aspects. The current optimization solutions use mathematical programming formulations and solve those using heuristic algorithms. These methods are offline and handle deterministic scenarios. They cannot efficiently adapt to unexpected, random behavior, as a complete solution is created in one go. In addition, the human factor has barely been considered. One method was proposed to handle human discomfort simultaneously with a performance objective. This, however, was a significantly different setup than the setup in other papers and our use case. Besides, a simple single-objective approach was used, which allows no insights into trade-offs. Hence, a gap exists for methods that use an online approach that handles stochasticity and environment interactions. Besides, including human aspects next to the performance must be explored more elaborately.

4.1.2 Workload Measurement

To achieve a fair workload distribution, we need a measure of workload. The most trivial measure would be the summed or average weight of the items a picker carries. However, other measures exist.

Gajšek et al. (2021) created an overview of the ergonomics measures used in order picking literature. The most frequently used measure is the OWAS measure (Karhu et al., 1977). This measure identifies the most common back postures, arm postures, leg postures, and load mass categories. It requires an observer to identify these postures. Hence, it is mainly useful when real-world experiments can be performed. In addition, the frequently used REBA measure (Hignett & McAtamney, 2000) also relies on observations in real-world experiments. Similarly, the RULA (McAtamney & Corlett, 1993) measure is used in order picking. This method relies on posture observations combined with weight information about the items. The last measure named by Gajšek et al. (2021) is the NIOSH lifting index (Waters et al., 1993). This measure does not rely on many observations. Instead, it uses a formula based on, amongst others, the vertical and horizontal distance of a lift and the weight of the item. This allows for better application in scenarios without frequent observations.

Larco et al. (2017) developed a measure of discomfort during order picking. Using Borg's CR-10 scale (Borg, 1982), they collected discomfort experiences from order pickers in two warehouses. Then, they used a linear regression model to determine the relation between several categories of pick quantity, mass, and volume. The resulting function can be used to determine the discomfort of picks. They applied this technique to two warehouses to develop two scales with concrete coefficients. However, both of these warehouses contained very small and light items compared to the warehouses of the customers of Vanderlande. Namely, the largest item category had a volume of at least 5 dm^3 and a mass of 3 kg, while these are considered small items in our considered warehouses.

4.2 Multi-Objective Deep Reinforcement Learning Methods

Since the introduction of DRL, several authors have developed methods for multi-objective DRL. These methods follow different streams. In this section, we provide an overview of the existing approaches. Table 4.1 summarizes the methods.

First, some methods use a single-policy approach. These approaches transform the multi-objective problem into a single-objective problem. One way to do so when having an unknown

Table 4.1: An overview of existing multi-objective DRL methods.

Paper	Approach	DRL Type	Short Description
Beeks et al. (2022)	Single	PPO	Bayesian optimization for reward shaping.
Abdolmaleki et al. (2020)	Single	Actor-Critic	Method that enables bounding the influence of an objective.
T. T. Nguyen et al. (2020)	Multi	DQN	Use TLO for non-linear utility. Multi-threading for multi-policy efficiency.
Mossalam et al. (2016)	Multi	DQN	Outer-loop with Optimistic Linear Support to determine which weights to pick.
K. Li et al. (2021)	Multi	Actor-Critic	Outer-loop with parameter copying. Combinatorial optimization focus.
Reymond and Nowe (2019)	Multi	DQN	Inner-loop. Poor performance in complex tasks.
F. Yang et al. (2022)	Single	DQN	Actions sampled from Pareto front. For multi-objectivization of single-objective problems.
Tajmajer (2018)	Multi	Dueling DQN	Modular structure, with Q-values aggregated using weighted sum.
Abels et al. (2019)	Meta	DQN	Conditioned network with Diverse Experience Replay in training.
R. Yang et al. (2019)	Meta	Double DQN	Conditioned network, using envelope updates and homotopy optimization
Hu et al. (2022)	Meta	Dueling DQN	Extend work by R. Yang et al. (2019) using dueling network, Noisynet, and soft target update.
Dornheim (2022)	Meta	DQN	Uses TLO in meta-policy approach.
Nian et al. (2020)	Meta	Actor-Critic	Adapts Abels et al. (2019) for partially observable MOMDP.
X. Chen et al. (2019)	Meta	PPO	Three-phased approach with adaptation, meta-policy training, and fine-tuning phase.
J. Xu et al. (2020)	Multi	PPO	Use evolutionary strategy to guide learning in promising directions.
F. Y. Liu and Qian (2021)	Meta	PPO	Combined three-phased meta-policy by X. Chen et al. (2019) with guided learning by J. Xu et al. (2020)
Abdelfattah et al. (2021)	Multi	DDPG	Two-Stage approach for learning in non-stationary environments with continuous action space.

utility function is to use reward shaping. In this approach, used by Beeks et al. (2022), a surrogate function was used to find the weights that should be applied in a weighted sum of the objectives. By doing so, weights can be found such that good results are achieved for both objectives. The disadvantage of this method is that only one point in the solution space is found. Hence, the Pareto Front is not explored, and no insights into the trade-off between objectives can be obtained to make an informed decision. Thus, in many scenarios, this method might not suffice. Another single-policy method was suggested by Abdolmaleki et al. (2020). They proposed a method that enables setting objective preferences for objectives of different scales more easily. In their actor-critic approach, they set a bound on the expected influence of each objective on the policy adjustment. This enables faster and easier exploration of suitable policy weights.

The second group of methods comprises the multi-policy approaches. These multi-policy approaches can be addressed in several ways. First, a naive method is to learn a policy from scratch for each utility function using single-policy methods. T. T. Nguyen et al. (2020) offer a framework for this, which can handle both linear and non-linear utility functions. To improve execution times, they offer multi-threading. Non-linear utility functions are handled using Thresholded Lexicographic Ordering (TLO). This is a method that allows the optimization of one objective while thresholding the other objectives. TLO enables finding different solutions than linear scalarization. However, it requires knowledge of how to specify the thresholds. Besides, Issabekov and Vamplew (2012) and T. T. Nguyen et al. (2020) showed that TLO outperforms linear scalarization when there is a single intrinsic objective, but with multiple intrinsic objectives, it generally performs worse.

Another multi-policy approach is the outer-loop approach, in which different policies are trained by looping over different weight combinations. Mossalam et al. (2016) were first to apply this in a DRL method, to train multiple DQNs that together form a coverage set. Instead of naively looping through linear scalarization weights, they used Optimistic Linear Support to determine which weights to use. In Optimistic Linear Support, weights are picked using the concept of corner weights. These weights are at the corner of the convex upper surface formed by the intermediate coverage set. The corner weight with the highest optimistic bound for improvement is selected at each iteration. Their experiments tested the impact of copying the DQN of the closest weight vector in the intermediate coverage set. They found that partially copying gives the best result, as it includes existing information but still allows sufficient freedom for exploration. K. Li et al. (2021) also used an outer-loop approach. They used an actor-critic algorithm with simple looping and partial weight copying. Their method was focused on solving a multi-objective traveling salesman problem. They constructed complete solutions in one forward pass, which is possible since there is no stochasticity or interaction in the problem. This deviates from most RL problems and is likely less suitable for the collaborative picking problem.

Opposed to outer-loop methods, in classical multi-objective RL techniques, there has also been a stream of inner-loop methods. These methods modify the internal workings of single-objective algorithms to use set operators and directly identify and save multiple policies in parallel (Roijers et al., 2013). Only one DRL paper (Reymond & Nowe, 2019) has proposed an inner-loop method. They proposed a Pareto-DQN algorithm that directly estimates the Pareto Front. However, the method did not work well for a high-dimensional, complex environment. F. Yang et al. (2022) proposed a method related to the inner-loop and single-policy categories. They adapted Q-learning to Q-learning based on the Pareto Front set. The Q-network estimates the rewards for each objective for each action. Then, instead of an epsilon-greedy policy, the "greedy" action consists of picking a random action on the Pareto Front that can be estimated by the Q-values. Besides, they adjusted the Q-learning update rule to use the expectation of Q in the Pareto front set for every objective. This method is aimed at scenarios where a single-objective problem is changed to a multi-objective problem to construct an easier new task. Due to the random policy over the entire Pareto Front, it does not seem suitable for tasks with competing objectives.

Besides the looping techniques based on popular classical RL techniques, different ideas have been proposed in DRL. Tajmajer (2018) proposed a modular learning technique. They aim to learn in parallel multiple DQNs, one for each objective. Then, to select an action, a scalar Q-value is calculated using a weighted sum based on the input weights of a user. This allows decomposing the problem into sub-problems and offers a method for changing utility preferences after training. To improve the scalarization, they proposed the addition of learning decision values in the networks. These decision values dynamically indicate which objective value is more important for the decision based on the input state. Including these decision values improved the robustness of the method to different preferences. However, due to the additional noise, the baseline performance is worse. A limiting factor of this method is that the Q-values are learned assuming a specific policy, but this policy can be different when new weights are used. This is not captured in the networks, leading to biased Q-value estimates.

Then, a new stream of techniques using “meta-policies” has gained attention in recent years. These techniques learn a single neural network, which can easily adapt to policies for different utility functions. Abels et al. (2019) were the first to introduce such a method, focusing on a dynamic weight setting. They proposed to condition the (dueling) DQN on the weights by adding these weights as input alongside the state. Then, training is performed using episodes with different weights. To ensure that the model does not only adjust to the most recently used weights, they proposed Diverse Experience Replay, which ensures that experiences of a variety of weight vectors are stored. Their experiments showed good performance for various weight settings, both with frequently and infrequently changing weights. R. Yang et al. (2019) proposed a successive meta-policy method. They used the concept of envelope updates instead of Diverse Experience Replay. This means that for each experience tuple, a vectorized reward is stored. Then, using a similar training scheme as Hindsight Experience Replay (Andrychowicz et al., 2017), one can sample multiple weights and optimize based on this variety of weights simultaneously. They theoretically showed that this leads to faster convergence. Additionally, they used so-called “homotopy optimization,” shifting between two loss functions during the learning phase. An empirical evaluation showed that their method performed better than Abels et al. (2019) and Mossalam et al. (2016) on four problem sets. This method has been improved further by Hu et al. (2022). They empirically showed improved learning stability and performance. They achieved this using a dueling DQN, a Noisynet method, and a soft target function update. The Noisynet method adds random perturbations to parameters, enabling persistent and complex permutations to agent strategies. This should improve the exploration-exploitation abilities compared to the epsilon-greedy policy. The generalized meta-policy approach has also been adapted for non-linear functions. Dornheim (2022) used a TLO-based approach. Using a multi-headed DQN that is conditioned on the thresholds and TLO action selection, they achieved policies that extend those on the Convex Coverage Set. The method contains fewer optimization tricks but shows a working concept. Lastly, Nian et al. (2020) adopted the ideas by Abels et al. (2019) and adjusted them to handle partially observable MDPs. They developed a recurrent conditioned actor-critic network, in which the actor-critic training is inspired by Heess et al. (2015). This concept is combined with the Diverse Experience Replay by Abels et al. (2019).

All the above multi-policy methods are value-based methods using a DQN or one of its variants. To handle continuous action spaces, some authors have proposed approaches using policy networks. X. Chen et al. (2019) proposed a meta-policy approach using PPO. This approach consists of three phases. First, in an adaptation phase, several policies for different preference vectors are updated based on the meta-policy. Then, in the meta-policy training phase, the meta-policy is updated using the aggregated data generated in the adaptation phase. Lastly, after the meta-policy has been trained, in the fine-tuning phase, the meta-policy is updated for a few iterations for each desired weight preference. They empirically showed improved performance over training separate policies from scratch. J. Xu et al. (2020) adopted a PPO-based method incorporating an evolutionary method to guide learning in the most promising direction. In each generation of evolution, stored data from previous iterations are used to fit a prediction model that helps to find the pairs of policies and scalarization weights that will improve the solution the most. Each selected policy-weight pair is then improved to produce offspring policies. These are used to create a new generation of policies. The final generation is divided into policy families by clustering, and policy parameters within each family are interpolated to produce a continuous approximation of the Pareto front.

F. Y. Liu and Qian (2021) combined the strengths of X. Chen et al. (2019) and J. Xu et al. (2020). They trained a meta-policy using the general three-phased approach by X. Chen et al. (2019). However, instead of randomly picking the preference vectors in each task, they use a prediction model for the expected improvement based on the idea by J. Xu et al. (2020). Their results show a better approximated Pareto set as well as faster and better adjustments to new preferences, compared to some benchmarks, including the method proposed by X. Chen et al. (2019). Compared to J. Xu et al. (2020), their performance is only better in terms of adaptability, which is the inherent strength of the meta-policy approach. If this adaptability is not required, the multi-policy approach by J. Xu et al. (2020) may be better as it could be more robust for complicated problems. However, this has not been elaborately tested.

Lastly, Abdelfattah et al. (2021) proposed a method that deals with continuous action spaces in non-stationary environments. They used a two-stage approach, utilizing DDPG. In the first stage, a set of generic skills (e.g., in a robot setting, Move Forward, Turn Left, and Turn Around) are learned. In the second stage, the learned skills are used in a hierarchical version of DDPG to produce a policy coverage set for the MOMDP. An intrinsically motivated RL algorithm is used to select which objective preferences to explore, and a policy bootstrapping mechanism is used to adapt to changes in the dynamics of the environment. The policy-based and DQN-based methods have not been compared since one group focuses on discrete action space while the other focuses on continuous action space. However, since PPO can also be used for discrete action spaces, it can still be considered in such scenarios.

In short, works on multi-objective DRL have spread along several categories. First, single-policy approaches transform the objectives into a single objective. This is a good option in some cases but provides no insights into any trade-offs and does not allow for any adaptation of preferences afterward. Then, multi-policy approaches establish a Pareto Coverage Set by training multiple policies. To aid this training, they use techniques such as weight copying and Optimistic Linear Support. Lastly, meta-policy approaches use a single network, adaptable to weight preferences given as input next to the state information. These works have been developed for both value-based and policy-based solutions. For value-based learning, these meta-policy approaches are state-of-the-art. For policy-based learning, the multi-policy method by J. Xu et al. (2020) and the meta-policy method by F. Y. Liu and Qian (2021) appear to be state-of-the-art. The methods using DQN and PPO have not been compared to each other as the PPO-based methods have only been applied to problems with continuous action space, while the DQN-based method can only be applied to problems with discrete action space. In general, the multi-objective DRL methods have been tested on typical benchmark DRL problems. However, they have not been elaborately used in applied DRL studies.

4.3 Deep Reinforcement Learning for Related Problems

This section outlines the usage of DRL in several related problems to provide insights into the feasibility of DRL for our collaborative picking problem and to explore potential methodologies. These related problems are online bipartite matching, vehicle routing, and vehicle dispatching problems.

4.3.1 Online Bipartite Matching

Online bipartite matching is a sequential resource allocation problem introduced by Karp et al. (1990). In this problem, a fixed set U of entities must be matched with at most one entity in an unknown incoming stream V upon arrival of the entities in V. These matches all have a cost or reward, and the goal is to establish a matching such that the final result after all arrivals is optimal.

Alomrani et al. (2021) proposed a DRL approach to solve the online bipartite matching problem. In their work, they modeled the state as a bipartite graph. They then tested several graph neural network architectures and features to investigate the best-performing structure. They found that the best performance was achieved using an encoder-decoder graph neural network with message passing. Adding historical information regarding the arrival history improved performance further. In addition, they found that using a node-wise invariant feed-forward structure also performed well when supplied with additional history information, suggesting that we can still achieve good performance without message passing as long as we use descriptive features of the system.

Our use case can also be seen as a matching problem, with possible picks being matched with human pickers that become available online. Hence, lessons can be taken from this method. On the other hand, some significant differences are not included in the modeling for the bipartite matching problem. Firstly, the bipartite matching problem does not consider any spatial relations between nodes, while in a collaborative picking system, many entities interact with each other at different distances. Secondly, in the bipartite matching problem, there is an unknown node arrival based on a hidden pattern. Oppositely, in our use case, arrivals of AMRs and pickers are interdependent. Besides, orders become available through pickruns that are known in the system. Hence, there are more interactions and patterns that may be taken into account. Lastly, in our use case, both sets U and V can be considered streams since both orders and pickers become available over time. Overall, we can take general lessons from this approach, such as the suitability of an invariant feed-forward network and incorporating an allocation-type modeling technique. However, this method does not cover all aspects of our problem.

4.3.2 Vehicle Routing Problem

The vehicle routing problem, first introduced by Dantzig and Ramser (1959), is an optimization problem that aims to find the optimal routing of a set of vehicles to serve a given set of customer locations. This relates to the collaborative picking problem, as we want to optimally allocate a set of pickers to a set of picking locations that must be visited sequentially. In recent years, several DRL solutions have been developed to solve the vehicle routing problem.

Raza et al. (2022) reviewed the recent advances of DRL solutions for the vehicle routing problem. These works generally use attention networks or encoder-decoder structures. Here, many models construct entire tours to solve the complete optimization instance. This limits the possibilities of these methods to be used in online settings. Yu et al. (2019) did propose an online method. However, they do not consider any constraints regarding time windows or condition-based availability of locations. These availability restrictions are crucial parts of our collaborative picking system.

Hence, current work in DRL for the vehicle routing problem is not developed sufficiently to handle online optimization with similar constraints as the collaborative picking problem. Besides, the problem characteristics are quite different, despite seeming similar at first glance. For example, constructing entire tours is nearly impossible in the collaborative picking problem due to the high interdependency of available picks. Additionally, collaborative picking systems run for much longer, and human pickers do not need to form a tour with begin and end. Therefore, DRL methods for vehicle routing problems cannot be directly transferred for this use case.

4.3.3 Dispatching Problem

Vehicle dispatching problems deal with the movement of a fleet of vehicles over a specific region. The goal is to dispatch vehicles geographically so incoming demand can be fulfilled efficiently (Y. Liu, Wu, et al., 2022). In recent years, it has gained importance due to the rise of online ride-hailing platforms and can also be applied to “classical” taxi networks. Like our use case, interdependent actors are allocated to handle online incoming demand patterns. Hence, several DRL methods have been proposed recently.

Both model-based and model-free methods exist. For example, Jin et al. (2019) combined a parallel ranking problem with a multi-agent RL formulation to solve a dispatching problem in a model-based way. In contrast, Jiao et al. (2020) learned value functions using dual policy evaluation. Model-free methods can be split into multi-agent and single-agent approaches. Works by Guo et al. (2020) and Wen et al. (2017) treat the problem as a multi-agent problem in which each vehicle is considered an agent and a DQN-network was trained to find the optimal policy. Oppositely, Holler et al. (2019) and Mao et al. (2020) used DQN and PPO to train a centralized fleet controller. This strategy aims at optimizing the dispatching policy at a system level. Y. Liu, Wu, et al. (2022) extended the latest model-free, single-agent works and were the first to offer a methodology that scales to complicated-real world scenarios. In their work, they made a split between the dispatching decision and the order-matching decision. Thus, they learned a policy using DRL that dispatches vehicles to regions. Then, within a region, vehicles are matched to incoming orders using a predefined method. The dispatcher policy receives requests from vehicles to be dispatched and assigns those vehicles to new locations. They modeled the state space as a grid, with each node representing a region, with its current number of vehicles, orders, and the potential income generated from those orders. Then, vehicles are assigned using a convolutional neural network and a pruned action space.

Although the pruning method and specific features of the method are case-specific and hard to transfer, we can take some lessons from this. For example, a single-agent dispatching policy that assigns incoming requests could be applied to our use case. Additionally, ensuring a restricted action space with potential options can aid learning. Oppositely, the general regional dispatching with a local prespecified matching method may not be optimal in our case. Instead, with the smaller, more specific area formed by a warehouse, it seems better to incorporate more exact decision-making within the DRL policy.

4.4 Fairness

With the rising use of Artificial Intelligence (AI) in decision-making, it has become increasingly important that decisions are made fairly. To this end, fair machine learning and fair RL have received increasing attention in recent years. In this chapter, we will first outline how fairness can be defined. Then, we will provide an overview of the current works in DRL that address fairness. This allows us to scope our work within the existing literature. Besides, it helps to define our fairness criteria and to identify potentially suitable methods for our work. Lastly, we will show how fairness is incorporated into optimization in general.

4.4.1 Defining Fairness

In fair AI research, most attention has been directed toward supervised machine learning. Several extensive surveys have been published on this topic by, amongst others, Barocas et al. (2019), Corbett-Davies and Goel (2018), Gajane and Pechenizkiy (2017), and Mehrabi et al. (2019). The main goal of fair machine learning is to prevent bias caused by so-called protected attributes. Protected attributes are attributes that can not be used for decision-making, such as race, gender, and religion. As Gajane and Pechenizkiy (2017) outlined, various notions of fairness exist in literature, and various measures and subcategories can be distinguished for each fairness notion. The most trivial notion is fairness through unawareness, which states that an algorithm is fair if protected attributes are not explicitly used in the prediction process. However, this is shown to have limited bias-mitigation ability because proxies for the protected attributes may be implicitly encoded in other attributes (Pedreshi et al., 2008). As a result, more explicit notions are used. The main concepts are individual fairness (i.e., similar individuals receive similar outputs), group fairness (i.e., approximately equal probability of an outcome for individuals across groups), and counterfactual fairness (i.e., output remains the same when the protected attribute is switched). As can be seen, fair machine learning mainly focuses on societal fairness in the sense of protected attributes. In our case, these protected attributes are not explicitly present and relevant. Hence, we consider other fairness measures.

Aside from fairness in supervised machine learning, Gajane et al. (2022) provide a survey on fairness in RL. In this survey, they denote two streams. The first stream is what they call societal fairness. This stream naturally follows from fair machine learning and focuses on preventing societal bias based on protected attributes. The other stream relates to non-societal fairness. These works mainly frame fairness as a way of adhering to defined constraints in allocation tasks or sequential decision-making problems. Example problems are resource allocation in computer networks or traffic flow control. This latter stream, however, is not investigated as expansively as the former.

For our study, we need a fairness measure between the two notions. Namely, we are dealing with humans that need to experience a fair distribution of rewards among all workers. However, we do not have protected attributes that influence our decision-making. This idea of having a "socially fair" algorithm has recently been studied in RL by Mandal and Gan (2022), who focused on three frequently occurring socially fair measures. The most straightforward fairness measure adopts the minimum welfare principle, also called max-min fairness or egalitarian fairness. This measure assumes that the fairest solution is the one in which the worst value by any actor is maximized.

Definition 4.1. *Given the values $x_i, \forall i \in \{1, \dots, n\}$ of n actors, the best minimum welfare value MW is given by:*

$$MW := \max_{\pi} \min_{i \in \{1,\dots,n\}} x_i$$

An extension of this measure is the generalized Gini function. This function can be adapted to different fairness notions using its parameters. Other fairness notions that can be achieved using this function are regularized max-min fairness, utilitarian fairness (i.e., maximize aggregate welfare), and leximin fairness (i.e., first maximize minimum value, then second lowest and so on) (Siddique et al., 2020). This fairness function is defined as follows.

Definition 4.2. *Given the values $x_i, \forall i \in \{1, \dots, n\}$ of n actors such that $x_1 \leq x_2 \leq \dots \leq x_n$ and a weight vector $\boldsymbol{w} = \langle w_1, w_2, \dots, w_n \rangle$ such that $w_i \geq 0$ for all i and $\sum_{i=1}^{n} w_i = 1$, the generalized Gini function $GGF_{\boldsymbol{w}}$ is given by:*

$$GGF_{\boldsymbol{w}} := \sum_{i=1}^{n} w_i x_i$$

Another fairness measure is the Nash social welfare measure, as defined in Definition 4.3. Since this is a multiplicative measure, any low rewards cause a punishment of the measurement value. It is similar to the leximin fairness principle, but a few values can influence the measure relatively heavily.

Definition 4.3. *Given the values $x_i, \forall i \in \{1, \dots, n\}$ of n actors, the Nash social welfare NSW is given by:*

$$NSW := \left(\prod_{i=1}^{n} x_i \right)^{1/n}$$

An additional fairness measure that was not used by Mandal and Gan (2022) but is very frequently used in literature is Jain's fairness index (Jain et al., 1984). This metric describes the throughput fairness over different users and is frequently used in resource allocation problems. As Jain et al. (1984) described, it has several desirable properties. Namely, it is applicable to any number of users, scale-independent, bounded, and continuous. Additionally, it directly relates to fairness and has an intuitive link to the human perspective on fairness. It is defined as follows.

Definition 4.4. *Given the values $x_i, \forall i \in \{1, \ldots, n\}$ of n actors, the Jain's fairness index JFI is given by:*

$$JFI := \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot \sum_{i=1}^{n} {x_i}^2} = \frac{1}{1 + \widehat{c_v}^2}$$

where $\widehat{c_v}$ is the sample coefficient of variation.

In addition to these measures, many other fairness criteria exist, of which some extend the named measures, and some are based on other principles. However, these are targeted at specific use cases with unique characteristics. For example, Quality of Experience measures can be used for variables with an interval scale instead of a ratio scale (Hoßfeld et al., 2017). Hence, we do not discuss those measures.

In short, fair machine learning, and RL specifically, has focused on various fairness notions. The most prevalent notion in fair machine learning is societal fairness that prevents unjustified bias. Conversely, we have non-societal fairness, which focuses on adhering to certain constraints in allocation tasks, mainly regarding computer networks or traffic flow. In our work, we must touch upon a notion of social fairness that forms a middle ground. There are several fairness measures based on different principles. Measures like the generalized Gini function, Nash social welfare measure, and Jain's fairness measure all have a solid theoretical and practical foundation and can be suitable for our use case. No measures explicitly incorporate the performance on an additional performance objective.

4.4.2 Fair Deep Reinforcement Learning

As mentioned in the previous section, Gajane et al. (2022) outlined the existing work in fair RL. What is immediately evident is that most works focus on traditional reinforcement learning. This allows for proofs of bounds on the performance and fairness level. However, there are also some DRL works that consider fairness. These studies focus on several fairness considerations.

C. Li et al. (2020) and Raeis and Leon-Garcia (2021) applied DRL for fair traffic light control. C. Li et al. (2020) used a single-objective approach to optimize a weighted reward function of the average and maximum waiting times. Raeis and Leon-Garcia (2021) proposed a similar approach, but they used two custom fairness functions based on throughput and waiting time instead of the maximum waiting time. Another relatively frequently studied topic is fair communication coverage in UAV control. These studies generally use complex, often non-linear, custom reward functions that combine fairness and performance, solving them using single-objective DRL. Authors like D. Chen et al. (2021), C. H. Liu et al. (2018), Qi et al. (2020), and Qin et al. (2021) incorporated Jain's fairness index into their functions, while others such as Ding et al. (2020) and Nemer et al. (2022) crafted their own fairness criteria. Generally, the fairness criteria are based on the same underlying measure (e.g., throughput) as the performance criteria. Tong et al. (2021) studied a use case of dynamic spectrum allocation. They also used a single-objective DRL approach to balancing the Quality of Experience mean and fairness using a custom reward function. Zhu and Oh (2018) proposed a method for fairness in multi-type resource allocation. They incorporated the Gini coefficient, a specific version of the generalized Gini index, with their custom reward function alongside the performance objective.

Aside from these papers that focus on specific problems, Siddique et al. (2020) proposed a generic method to learn fair policies in DRL. They proposed to use the generalized Gini function to determine a fair policy over multiple objectives. They modeled the problem as an MOMDP. In this model, each reward should have the same scale, and the goal of the algorithm is to distribute those rewards according to the generalized Gini function fairly. Thus, one could say that, although a multi-objective model is used, the problem has one objective. This objective is to achieve a fair distribution of rewards among individuals according to the generalized Gini function.

Concluding, fair DRL has received limited attention in the literature. The main works focus on non-societal fairness. Several methods have used social fairness measures, such as Jain's fairness measure or the Gini function. The methods all use single-policy DRL approaches, solving a specific reward function that integrates the fairness aspect. Additionally, in these works, the underlying measures for fairness and performance criteria are generally the same or similar. Oppositely, in our use case, we have a fairness metric based on an inherently different measure than our performance metric. Besides, we want to explore the trade-off between the objectives and allow for adaptation to different preferences. For this setting, a gap in current fair DRL exists. Lastly, no studies use fair DRL in the operations management domain.

4.4.3 Fairness in Optimization

To gain insights into the current state of fairness considerations in optimization problems and how fairness and optimization are combined, we reviewed existing works in this intersection. In Table 4.2, we provide an overview of the existing literature. This overview lets us understand which fairness measures are prevalent in the intersection between fairness and optimization. Besides, we analyzed which type of solution approaches are used, whether they use a single-solution approach, or whether they aim to find multiple (non-dominated) solutions for different trade-offs. In addition, we gathered insights into the application domains and whether the methods offer general solution strategies or problem-specific approaches.

Our overview shows that max-min fairness is the most commonly used fairness measure in optimization problems. This is related to several factors. Firstly, it is a relatively straightforward measure that is easy to understand and interpret. Secondly, due to its easy computation, it can be embedded in mathematical techniques such as MILP in a relatively straightforward way. Thirdly, the measure fits the considered problem domains well. Many studies considering max-min fairness consider resource allocation or scheduling problems where machines or devices are used, and the algorithm does not influence the total workload. Hence, by optimizing max-min fairness, the entire distribution can be brought to a stable level without needing to consider the overall load of the system.

Aside from max-min fairness, we find a relatively even spread over the used fairness measures. Jain's fairness index, the Nash social welfare measure, and the generalized Gini index are all used in several problems. Besides, α-fairness, which is a measure related to max-min fairness, and quality of service fairness are considered in some studies. Lastly, several studies consider a custom fairness measure or a variance-based metric. In general, there does not seem to be a clear pattern for which measures are used in which problem domains. The choice is often arbitrary and based on what is considered relevant by the authors or what fits the optimization method best.

Considering the problem domains that are investigated, we find that many of the problem domains are similar. Namely, most methods aim at variants of resource allocation, load balancing, and network optimization problems. This is as expected since these are prevalent problems in combinatorial optimization literature and they inherently consider workload distribution over resources. Other considered domains are recommender systems, supply chain optimization, and the vehicle routing problem. All in all, there are few use cases in which fairness for humans is considered. In addition, most studies consider specific problem instances and do not consider generalizable methods. Some exceptions are the works by J. Jiang and Lu (2019) on fairness in multi-agent RL, Busa-Fekete et al. (2017) on multi-objective bandits, and Alabi et al. (2018), Escoffier et al. (2013), and V. H. Nguyen and Weng (2017) on combinatorial optimization problems.

Lastly, in terms of solution approaches, we find two main observations. First, most methods focus on deterministic optimization. Second, most methods use a single-solution approach, although in recent works solving problems in a multi-objective way is becoming more prevalent. Many problems are modeled using an MILP formulation. Especially in the older studies, the problems are mostly solved using more classical optimization methods such as using an MILP solver, approximation algorithms, or lagrangian optimization. More recently, evolutionary algorithms and meta-heuristics such as the genetic algorithm or ant colony optimization have been used. Using these evolutionary approaches, more focus has been put on finding a non-dominated set of solutions to explore the trade-off between fairness and another objective. This increased interest reflects the increasing attention paid to fairness by decision-makers, as it is not just considered a nice-to-have extra, but its effects and relation with performance must be understood.

In short, fairness is getting increased attention in optimization literature. Many methods still focus on relatively standard combinatorial optimization problems, but some different applications have also been studied. Max-min fairness is a prevalent measure that fits many of these problems well. However, different measures are used in various applications based on the problem characteristics. In recent years, more focus has been put on incorporating fairness in a multi-objective way, such that a solution set reflecting the trade-offs can be explored. With the increasing importance of fairness in decision-making, this seems to be the way forward. For future research, many opportunities exist in developing general guidelines and methodologies to incorporate fairness in optimization. Besides, a more comprehensive range of problem domains must be tackled. Lastly, most current methods focus on scenarios wherein fairness and performance have the same underlying measure, such as throughput. Methods in which the underlying fairness criterion differs from the performance criterion are still scarce. Here, our use case differs from the current methods and can help expand the literature on these problems.

Table 4.2: Overview of existing methods considering fairness in optimization problems. The table contains the considered fairness measure, applied solution approach, whether the method generates a single or multiple (non-dominated) solutions, and the application domain.

Paper	Measure	Approach	Solution	Domain
Kleinberg et al. (2001)	Max-min fairness	Approximation algorithm	Single	Load balancing
Harks (2005)	Proportional and max-min fairness	Lagrangian optimization	Single	Bandwidth allocation
Pioro (2007)	Max-min fairness	Seq. lexicographic optimization	Single	Bandwidth allocation
Ishida et al. (2006)	Variance	Nonlinear program	Single	Multipath network flow
Pishdad et al. (2010)	Quality of service fairness	Polynomial time approximation scheme	Single	Job scheduling
Koppen et al. (2010)	Max-min fairness	Multi-objective evolutionary algorithms	Multi	Network congestion control
Meng and Khoo (2010)	Custom fairness measure	Genetic algorithm	Multi	Ramp metering
Devarajan et al. (2012)	Jain's fairness index	Custom algorithm	Single	Power allocation
Tangpattanakul et al. (2012)	Maximum difference	Genetic algorithm	Multi	Satellite Scheduling
Stolletz and Brunner (2012)	Custom fairness constraints	MILP	Single	Workforce scheduling
Escoffier et al. (2013)	α-fairness	Polynomial time approximation algorithm	Single	Multi-agent combinatorial optimization
Amaldi et al. (2013)	Max-min fairness	MILP	Single	Network optimization
Bertin et al. (2014)	Custom fairness measure	Lagrangian optimization	Single	Job scheduling
Yue and You (2014)	Nash bargaining fairness	Nonlinear program	Single	Supply chain optimzation
Yaacoub and Dawy (2014)	Max-min and quality of service fairness	Custom algorithm	Single	Radio resource management
Dely et al. (2015)	Max-min fairness	MILP	Single	Bandwidth allocation
Partov et al. (2015)	Custom fairness measure	Primal-dual algorithm	Single	Network optimization
Sawik (2015)	Custom fairness measure	Stochastic MILP	Single	Supply chain optimization
L. Xu et al. (2015)	Jain's fairness index	Hybrid ant colony optimization	Single	Resource allocation
Z. Li et al. (2016)	Max-min fairness	ϵ-constraint method	Multi	Network traffic offloading
Busa-Fekete et al. (2017)	Generalized Gini Index	Online gradient descent	Single	Multi-objective bandits
X. Liu et al. (2017)	Max-min fairness	Evolutionary algorithms	Single	Load balancing
V. H. Nguyen and Weng (2017)	Generalized Gini Index	Primal-dual algorithm	Single	Classic combinatorial optimization
Ye et al. (2017)	Max-min fairness	Two-stage algorithm	Single	Task allocation in transportation
Alabi et al. (2018)	Multiple convex group-fairness measures	Polynomial-time reduction method	Single	General multi-objective optimization
Doi et al. (2018)	Custom and max-min fairness	Decomposition-based metaheuristic	Single	Crew scheduling

Limmer and Dietrich (2018)	Custom fairness measure	Genetic Algorithm	Multi	Dynamic pricing
Arribas et al. (2019)	α-fairness	Heuristic non-convex optimizer	Single	Network optimization
Diao et al. (2019)	Max-min fairness	Iterative algorithm	Single	Data allocation and trajectory optimization
J. Jiang and Lu (2019)	Custom variance-based measure	Hierarchical multi-agent RL	Single	Multi-agent RL
Zhao (2019)	Max-min and quality of service fairness	Alternating optimization algorithm	Single	Wireless network design
Clausen et al. (2020)	Max-min and leximin fairness, and variance	Genetic algorithm	Single	Resource allocation
Jagtenberg and Mason (2020)	Nash social welfare	MILP and local search	Single	Ambulance facility location problem
Kermany et al. (2020)	Custom fairness metric	Genetic algorithm	Multi	Recommender System
Z. Zhang et al. (2020)	Max-min fairness	Multi-objective local search	Multi	Vehicle routing problem
Z. Li et al. (2021)	Max-min fairness	MILP	Single	Emergency resource location problem
Lu and Wang (2021)	Max-min fairness	Alternating optimization algorithm	Single	Network optimization
Malencia et al. (2021)	Max-min fairness	Supermodular algorithm	Single	Multi-agent task allocation
Munguía-López and Ponce-Ortega (2021)	Nash social welfare and max-min fairness	MILP	Single	Vaccine allocation
Purushothaman and Nagarajan (2021)	Jain's fairness index	Evolutionary algorithm with neural network	Single	Resource allocation
Rahmattalabi et al. (2021)	Multiple group-fairness measures	MILP	Single	Influence maximization
Tang et al. (2021)	Gini coefficient	Genetic algorithm	Multi	Water resource allocation
S.-Z. Zhou et al. (2021)	Variance	Ant colony system algorithm	Multi	Crew scheduling
Zimmer et al. (2021)	Max-min and proportional fairness and Generalized Gini Index	multi-agent RL algorithm	Single	General multi-agent RL
Arribas et al. (2022)	α-fairness	Extremal optimization	Single	Network Optimization
Fan et al. (2022)	Nash social welfare	Q-learning adaptation	Single	Multi-objective classic RL
F. Li et al. (2022)	Custom fairness measure	Genetic algorithm	Multi	Multi-workflow scheduling
Y. Liu, Huangfu, et al. (2022)	Quality of service fairness	Proximal stochastic gradient descent	Single	UAV placement
Kuai et al. (2022)	Max-min fairness	Offline PPO	Single	Virtual network scheduling
Sadiq et al. (2022)	Custom fairness measure	Non-linear marine predator algorithm	Single	Power allocation
Y. Wang et al. (2022)	Maximum difference	Genetic algorithm adaptation	Multi	Virtual power plant profit allocation
Gong and Guo (2023)	Gini coefficient adaptation	Custom genetic approach	Multi	Influence maximization
Y. Jiang et al. (2023)	Custom fairness measure	Genetic algorithm with large neighborhood search	Multi	Airport gate assignment
Wu et al. (2023)	Custom fairness measure	Multiple gradient descent	Multi	Recommender System

4.5 Synthesis and Research Gap

In this literature review, we first outlined the existing works in collaborative picking. Several aspects of collaborative picking have been studied, like picking optimization and the impact of human-robot interaction. Existing works use deterministic methods, like MILP formulation with heuristic solution approaches, to optimize picking and batching. However, in real-life, disturbances may take place. Thus, we identify a gap for an online solution approach that can handle stochastic environments in collaborative picking while also extending to long time horizons. In addition, we found that little to no attention is paid to handling the human workload during optimization, despite human workers having a crucial role in the system.

DRL offers a framework that can meet this requirement. Regular DRL techniques, however, can only optimize based on a single reward function. Oppositely, in our problem, we aim to optimize the picking performance while maintaining a fair workload distribution and exploring the trade-offs. To do so, we aim to use a multi-objective DRL approach. Therefore, we need multi-objective DRL techniques. Exploring the existing multi-objective DRL literature showed several solution categories, being single-policy, multi-policy, and meta-policy strategies. The meta-policy approaches seem to be state-of-the-art for value-based learning, while a multi-policy (J. Xu et al., 2020) or meta-policy (F. Y. Liu & Qian, 2021) method is best for policy-based learning.

To establish the feasibility of DRL, we found several similar problems to collaborative picking in which DRL has been applied. These problems, being online bipartite graph matching, vehicle routing problems, and dispatching problems, showed that DRL has the potential to be valuable for collaborative picking. Besides, they offered some potential modeling concepts. On the other hand, the problems all have different properties that prevent us from applying the methods directly.

To define our fair workload objective, we need two measurements. First, we need a workload measurement. We found that the NIOSH lifting index (Waters et al., 1993) and a discomfort measure (Larco et al., 2017) are most suitable for our use case. However, both have their deficiencies that limit the direct application. Secondly, we must define fairness. In Section 4.4, we discussed several measures, the generalized Gini function, Nash social welfare measure, and Jain's fairness measure, that all offer valid metrics for our use case. We found no measures that explicitly include an additional performance objective.

Continuing, we explored the existing fair DRL methods. The existing works mainly use single-policy approaches with handcrafted reward functions. No works use a multi- or meta-policy strategy to explore the trade-offs between a fairness objective and a performance objective based on different underlying properties. Thus, a gap exists to incorporate multi-objective DRL with fair DRL. Besides, the underlying subject of fairness in these methods is mostly the same as the subject of performance, while in our case, these differ from each other.

Lastly, we outlined the current works that consider fairness in optimization. We found that, similar to the DRL approaches, single-solution methods are most frequently used. However, more focus has been put on multi-objective considerations and exploring trade-offs in recent years. Most works focus on specific use cases, and the selected fairness measures are mostly arbitrarily picked based on the specific problem characteristics. The studied problem domains are relatively common combinatorial optimization problems. There is still much work to be done in expanding the application domains besides further exploring the trade-offs between fairness and performance.

Concluding, existing collaborative picking works do not consider stochastic, interactive scenarios and lack consideration of human workload. To handle this, multi-objective DRL offers a promising framework in which an interactive picker allocation policy can be learned, and a fairness objective can be included next to a performance objective. We identified the state-of-the-art multi-objective DRL techniques, several fairness measures that can be used, and how fairness is currently considered in optimization studies. There exists a gap in the current collaborative picking literature for online solution approaches that handle stochastic, interactive systems. In addition, current collaborative picking methods insufficiently consider the human workload and fairness. In DRL research, fairness has also received limited attention, and there are no existing works that combine multi-objective DRL and fair DRL to explore the trade-offs between performance and fairness. We position our work within these interdisciplinary gaps. We aim to stimulate these research tracks further to enable the adoption of multi-objective DRL, as well as the integration of fair and multi-objective optimization techniques and the consideration of the human factor in operations management.

Chapter 5

Methodology

In this chapter, we focus on the methods that we used to solve the problem of allocating pickers to orders in collaborative order picking. This methodology consists of two parts. First, in Section 5.1, we will describe our developed simulation model. This simulation model is vital for both developing and evaluating solutions. Second, in Section 5.2, we will outline our DRL approach. Specifically, in Section 5.2.1, we will define the MOMDP. Consequently, in Section 5.2.2, we will explain the learning algorithms we used. Lastly, we will introduce the architecture of our DRL agent in Section 5.2.3.

5.1 Simulation Model

To represent the collaborative picking system, we used discrete-event simulation. Discrete-event simulation models the evolution of a system over time using instantaneous changes at separate times, called events (Law, 2015). Between these events, a specific amount of time can take place. This method abstracts away from the exact real-life processes, which increases speed and reduces unnecessary complexity but allows for a detailed simulation by including all the critical events.

To implement the simulation, we used Python 3.9 with the discrete-event simulation package SimPy (Matloff, 2008). To facilitate DRL, we wrapped the simulation model within the OpenAI Gym interface (Brockman et al., 2016). We had product data and order picking data from a few days of one of the customers of Vanderlande. As the order picking data was limited, we relied mainly on stakeholder information to verify the suitability and assumptions of the simulation model. The simulation model is split into three main entities: the warehouse, pickers, and AMRs. In Section 5.1.1, we will outline the warehouse specifications and assumptions. Then, in Sections 5.1.2 and 5.1.3, we will discuss the picker and AMR processes, respectively.

5.1.1 Warehouse

In this study, we considered a traditional warehouse layout. Namely, we assumed warehouses with vertical, parallel aisles with storage racks on both sides of the aisles. At the top and the bottom, two horizontal cross-aisles connect the vertical aisles. For AMRs, the vertical aisles are unidirectional, with the travel direction alternating between consecutive aisles. On the other hand, pickers can move in both directions in each aisle. Additionally, in the horizontal cross-aisles, pickers and AMRs can both move in each direction. The distance between adjacent pick locations within an aisle is 1.4 meters, while the distance to move to the other side of the aisle is 1 meter. The travel distance between two aisles is 6 meters. Figure 5.1 clarifies these warehouse parameters.

To enable efficient calculations and to model the warehouse layout, we used a graph structure. Here, the nodes represent locations at which entities can be. The edges represent how entities

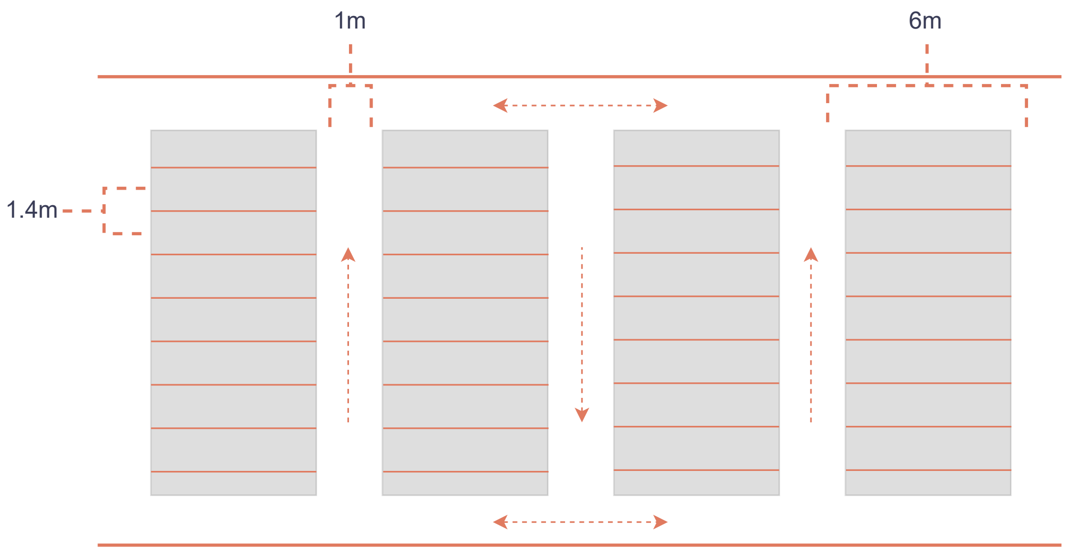


Figure 5.1: Illustration of the considered warehouse parameters. The image represents a top view from a small section of a warehouse. The dotted arrows represent the allowed AMR movement directions, and the grey areas represent storage racks between the aisles.

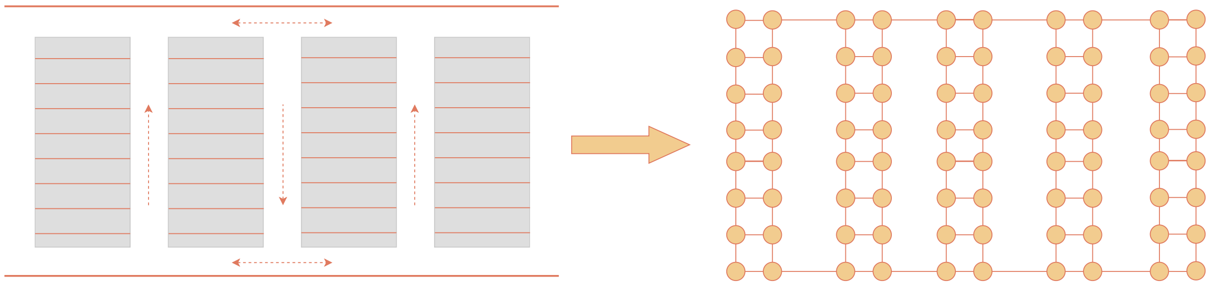


Figure 5.2: Illustration of the undirected graph representation of a warehouse. The circles indicate nodes and the connections between the circles are edges.

can move within the warehouse and what the distances between these locations are. We illustrate this graph representation in Figure 5.2. The resulting adjacency and distance matrices can be used to calculate distances and routes within any warehouse layout efficiently. In our use case, we used a directed graph to represent the AMR travel possibilities, while an undirected graph was used for the pickers, which can move in any direction within the warehouse. Different warehouse structures and travel direction rules can be implemented by switching the graph structure and, therefore, the adjacency and distance matrices.

A pre-generated set of pickruns must be collected by the pickers and AMRs to fulfill an episode. Thus, simulation episodes start with the pickrun generation and end when all items from all pickruns have been picked. These pickruns contain the list of locations that must be visited, with the number of items that must be picked at each location. The pickrun optimization is a problem on its own. As this is outside our scope, we used a basic approach. Namely, to generate these pickruns, for each episode, we randomly selected a set of pick locations. We determined the pickrun lengths using uniform sampling between 15 and 25 locations. We selected these lengths based on stakeholder knowledge and the pickrun lengths we found in the available data. These pickruns must be collected by the AMRs via an S-shaped/traversal routing policy (i.e., from the left to right side of the warehouse via aisles with alternating directions; Petersen, 1997), which is the policy that Vanderlande intends to use within their collaborative picking concept. Hence, we sorted the locations by aisle and then by how early the locations are within their respective aisle, with the aisle entries being on the opposite end of the aisle for consecutive aisles. The picking frequencies at the locations were randomly sampled using the empirical distribution of pick frequencies from the available data (see Appendix B). For the pickrun-AMR assignment, we used a trivial method, with the first pickrun in the queue being assigned to an AMR that becomes available.

To start a simulation run, we used a diverse starting method. In diverse starting, all AMRs are assigned to a pickrun. These pickruns are cut off using a random uniform selection. In this way, the system starts with the AMRs randomly spread through the warehouse. Similarly, the pickers are randomly allocated to destinations spread throughout the warehouse during instantiation. We did so based on expert knowledge, as initialization procedures to create distributed initial states are common.

To generate a product distribution through the warehouse, we randomly instantiated product locations based on the actual products and product categories in the warehouse of the Vanderlande customer. To do so, for each product category, we gathered the distribution of how many items of the category are clustered together. Then, to fill a warehouse with product locations, we randomly sampled a product category based on the relative frequencies of the categories. Consequently, we sampled how many products must be grouped for this product category based on the empirical distribution. Finally, real-world products of these categories were assigned to these locations. This was done repeatedly until each location contained a specific product with its weight and volume. The distributions of the products and weights can be found in Appendix B.

We determined the expected pick time of an order line on a pickrun based on the product characteristics and the number of items that must be picked. To do so, we used an internal method from Vanderlande that was developed using the empirical product and pick time data. We do not fully disclose this method due to confidentiality reasons. This method combines the product volume and weight with the number of item pairs and single items that must be picked. Using several empirically tested linear functions that use these two product characteristics and the number of items that must be collected, the expected pick time t_{pick} can be calculated for each pick. To create the actual pick times, we sampled a value from a Gaussian distribution with $\mu = t_{\text{pick}}$ and $\sigma = 0.1 \cdot t_{\text{pick}}$.

Since sampling the product characteristics and the number of items that must be picked occurs independently, we verified whether the resulting expected pick times are similar to those calculated from the real order distribution, which contained 100,833 order lines. The histograms of 100,000 sampled picking times through our method and those from the data show a sufficiently similar distribution for our purpose. The histograms can be found in Appendix B. Besides, the means and standard deviations of the sampled pick times ($\mu = 11.3$, $\sigma = 10.3$) and of the actual order pick times ($\mu = 12.3$, $\sigma = 10.8$) are also satisfactory similar.

5.1.2 Picker Process

The picker process describes the picker's logic and how it interacts with the optimizer and AMRs. Figure 5.3 shows an overview of this picker process.

In the simulation, we modeled each transition of one location to the other location by a picker or AMR in the warehouse as an event. This allows us to maintain a detailed overview of the current state of the system with regard to the locations of all pickers and AMRs at any time. The picker process starts with a picker being allocated to a destination. Once the picker receives its destination, it follows the shortest path to the destination. We set the average picker speed to 1.25 m/s. At the start of each path to a new destination of a picker, we randomly set the speed using a Gaussian distribution with $\mu = 1.25$ and $\sigma = 0.15$. This mimics the uncertainty in real-world picker speeds. After a timeout of distance/picker speed seconds, the movement

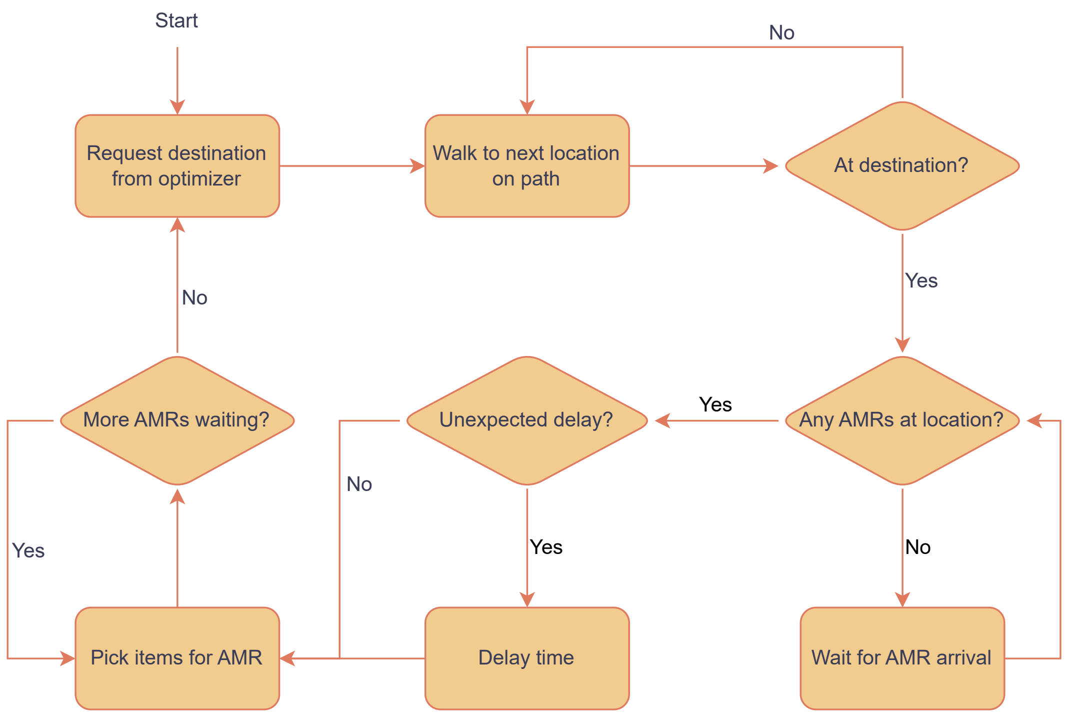


Figure 5.3: Overview of the picker process in the simulation model.

event toward the following location takes place. This is repeated until the picker reaches its destination.

When a picker reaches the destination, it checks if any AMR is waiting there. If no AMR is waiting at the location, the picker waits until any AMR arrives there. When an AMR is waiting at the location or an AMR arrives, the picking takes place. This picking is represented using a timeout event. The picking time is sampled from a Gaussian distribution as explained in Subsection 5.1.1. However, in real-world warehouses, picking does not always happen perfectly. Therefore, in consultation with business stakeholders, we included a random picking interruption. Namely, a picking delay is included every once in a while to mimic any uncertainty caused by pickers. This delay can represent pickers having a short break or having to reshuffle items on the AMR, items being hard to retrieve from the shelve, and so on. We set the frequency of this unexpected delay occurring for each picker using a Poisson random variable with $\lambda = 50$, indicating that, on average, a picker has an unexpected delay once per 50 picks. We used a Poisson distribution as it has been frequently used to model event occurrences, like accidents (Nicholson & Wong, 1993) or power failures (Y. Zhou et al., 2006). The distribution fits well when events are independent, which we can assume since a disruption, stock-out, or error at one location generally does not affect those at the next locations. The disruption time is sampled from a Gaussian distribution with $\mu = 60$ and $\sigma = 7.5$ seconds.

When a picker has finished picking the items for an AMR, it checks if any other AMR is waiting at the location. If so, it picks the items for this AMR. Then, if no AMRs are left to be served at the location, the picker must request a new location from the optimizer, and the cycle repeats.

5.1.3 AMR Process

The AMR process establishes the behavior of the AMRs within the system and how they interact with each other and the human pickers. Figure 5.4 outlines the process logic of the AMRs in the simulation.

The AMR process starts with an AMR being assigned to fulfill a pickrun. Then, like for pickers,

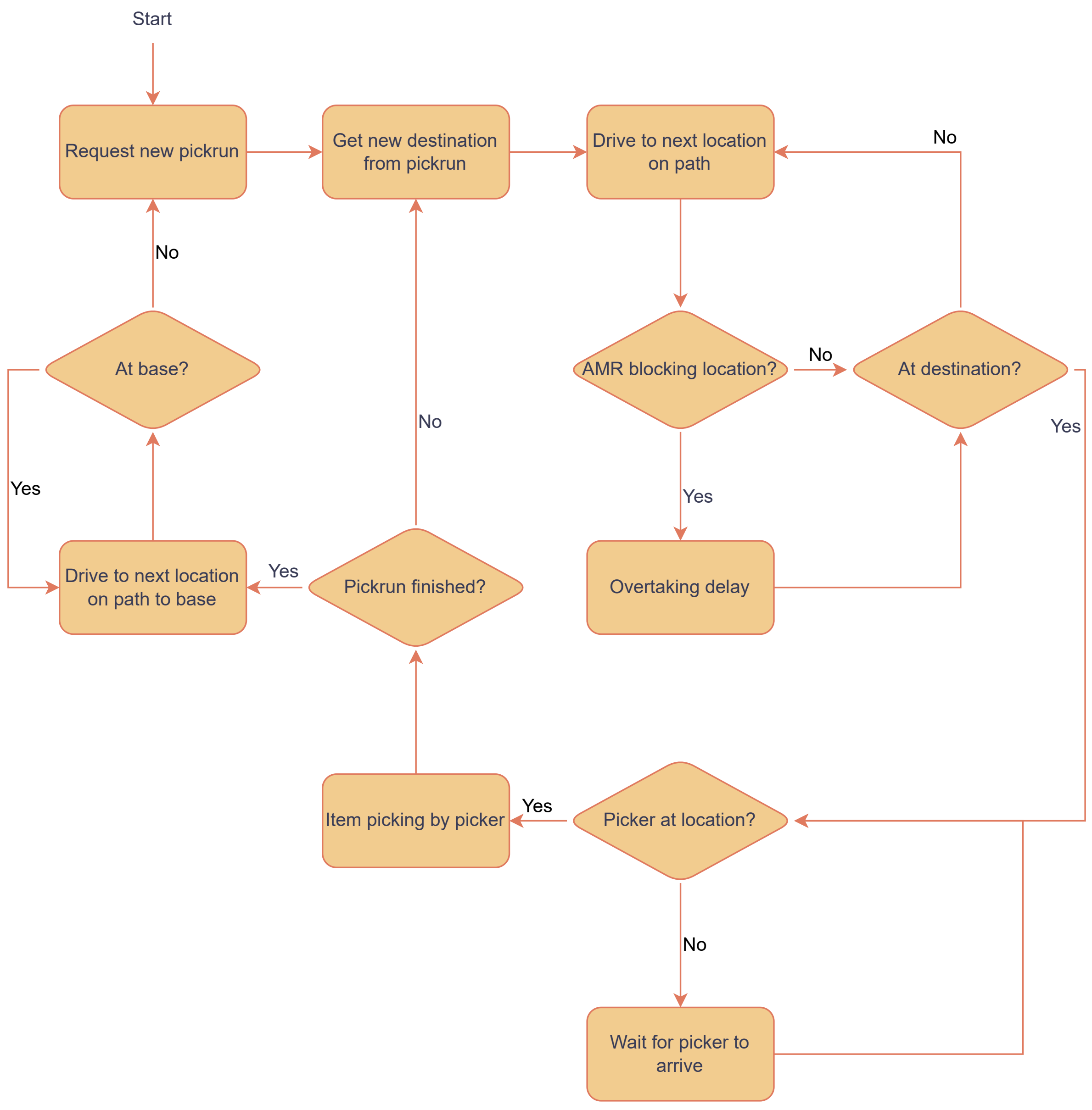


Figure 5.4: Overview of the AMR process in the simulation model.

an event is used for each transition to the following location on the AMR path toward the first destination in the pickrun. For AMRs, these events occur after a timeout of distance/AMR speed seconds. We set the average AMR speed to 1.5 m/s. Similar to the picker speed, at each tour toward a new destination, a random speed is selected from a Gaussian distribution with $\mu = 1.5$ and $\sigma = 0.15$ m/s. Like the pickers, the AMR continues its movement until it has reached its destination.

However, whereas pickers can walk easily through the warehouse, the AMRs can encounter congestion. Namely, when another AMR is standing in the path of the vehicle, it has to do an overtaking maneuver. We model this by adding an overtaking timeout whenever an AMR has to overtake another AMR. Since overtaking is a slow procedure for AMRs, this time penalty is relatively large. We used a random Gaussian sampling method with $\mu = 15$ and $\sigma = 2.5$ seconds for the overtaking penalty. This penalty indicates the importance of preventing congestion.

When an AMR reaches its destination, it waits for the human picker to arrive if it is not yet there. When the human picker arrives, a picking timeout occurs, representing the picker picking the items for the AMR. This is modeled as discussed in Subsections 5.1.1 and 5.1.2.

Once an AMR has been loaded at a pick location, the process is repeated, and the AMR moves to the next location on its pickrun. This occurs until all locations in the pickrun have been visited. Then, in a real-world warehouse, the AMR moves to a drop-off location outside the picking area to unload its items. As we do not consider this unloading, we do not include it in our simulation. Instead, the AMR must return to the base location at the bottom left of the warehouse, where it can start a new pickrun. As the AMRs at the drop-off location are simply out of the system, not including this process in the simulation does not affect the picking efficiency results. By up- or down-scaling the number of AMRs, we can still capture the number of AMRs that are actually in the picking system.

In summary, the simulation model represents multiple interactive processes between pickers, AMRs, and the picker allocation optimizer. There are several sources of uncertainty. Specifically, the picker and AMR speeds are stochastic, as well as the picking times at the pick locations. In addition, random disruptions can occur during item picking, creating relatively long delays. Lastly, congestion causes additional travel delays for the AMRs related to overtaking procedures.

5.2 Deep Reinforcement Learning Approach

To develop a DRL approach, two steps are needed. First, in Section 5.2.1, we will describe how we defined the MOMDP. Then, Sections 5.2.2 and 5.2.3 will outline the used learning algorithms and agent architectures.

5.2.1 Markov Decision Process

To model the agent-environment interface, the first decision is to determine the general role of the DRL agent. We defined our agent as the general picker allocation optimizer. This allows us to learn a single policy that optimizes the combined allocation of all human pickers and the system as a whole. Another option would be to model the solution as a multi-agent system where pickers are agents. However, these methods can suffer from problems such as non-stationarity and difficulty in obtaining global optimum solutions due to independent actors with partial information (Canese et al., 2021). In our case, we have full information on the entire system. With such full information, a centralized policy works better for optimizing system-level performance than a distributed solution, as shown in, for example, the related dispatching problem as described in Subsection 4.3.3.

The agent must act within a continuous cycle in which it receives requests and allocates pickers to new destinations. The cycle is illustrated in Figure 5.5. In this cycle, the pickers and AMRs fulfill their respective order picking processes. Whenever a human picker finishes its task at its destination, it places a request for a new destination. The picker optimizer receives the system state and allocates the picker to a new destination. Then, the cycle continues until any picker places a new request. Note that, as the process is stochastic, multiple pickers will never place a request at the exact same moment. Therefore, the picker optimizer uses the natural order of incoming requests to allocate pickers one by one.

Within this framework, we define the MOMDP below. Specifically, we define the state space, action space, and reward function. The transition function is formed by the warehouse system in which the agent interacts. One transition step is formed by the picking process between two consecutive allocation requests for the optimizer agent. In our case, this picking process is represented by the simulation model that we defined. An episode in our system is one warehouse simulation in which a pre-generated set of pickruns is wholly picked.

State Space

To model the state space, we need a representation that can capture spatial information, handle different numbers of human pickers and AMRs, and adjust to different warehouse layouts.

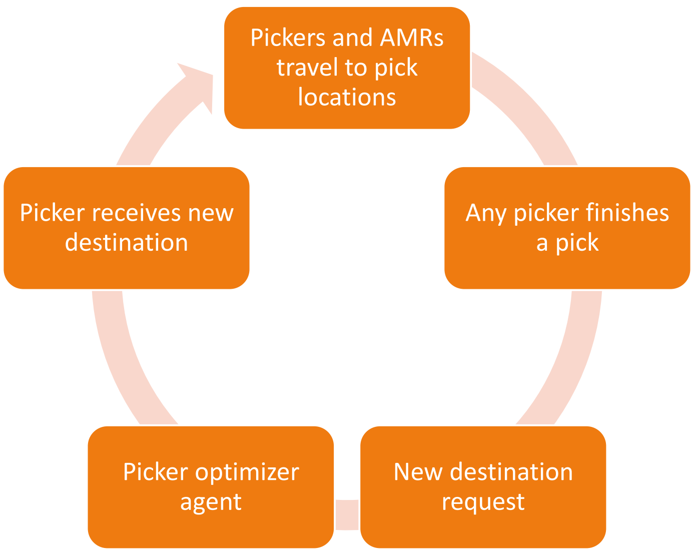

Pickers and AMRs travel to pick locations
Any picker finishes a pick
New destination request
Picker optimizer agent
Picker receives new destination

Figure 5.5: Illustration of the cycle within which the picker optimizer agent must act.

Therefore, we used a graph representation for the state space. In this graph, the node set $\mathcal{V}_G$ consists of all possible locations within the warehouse. All adjacent locations between which pickers can move without passing through another location are connected with edges. We used an undirected graph, so the edge set $\mathcal{E}_G$ consists of unordered edges. This graph structure coincides with the undirected graph we used internally in the simulation model, as shown in Figure 5.2. We captured all relevant information in the node features $\mathcal{X}_G$ and did not use any edge features $\mathcal{A}_G$, as the connections between warehouse locations are generally homogeneous.

The node features we used can be split into two categories: efficiency-related features and workload fairness features. Below we first describe the efficiency-related and then the fairness-related features.

Efficiency Features The efficiency features that we used consist of several subcategories. These subcategories are current picker information, AMR information, other picker information, node location information, and node neighborhood information.

The current picker information describes the positioning of the nodes in relation to the controlled picker for which an allocation decision must be made. We used the following two node features to capture this:

- Current picker location indicator: Binary value that is 1 if the controlled picker is currently at this node and 0 otherwise.
- Distance from picker: The distance in meters of the picker to this node by following the warehouse paths. The distance can be retrieved from the pre-computed distance matrix.

The AMR information describes the positioning and next destinations of the AMRs with respect to the nodes. To do so, we used seven node features:

- AMR location indicator: Binary value that is 1 if an AMR is currently at this node and 0 otherwise.
- Number of AMRs to node: The number of AMRs that are currently going toward this node for picking.
- AMR destination distance: If any AMR has this node as destination, the distance in meters that this AMR must still travel. If no AMR goes to this node, the value is -10.

If multiple AMRs, the minimum of the distances. We used values of -10 as it is hard to define a general maximum. On the other hand, the distance values will never be below 0, so -10 always works and adds a margin to differentiate from 0, which is the minimum possible distance.

- Expected time until next AMR destination: Similar to the previous feature, but considers if an AMR is going to the node for the next destination in its pickrun. This feature includes the expected travel time until the current destination, the expected pick time at the destination, and the expected travel time until the next destination. If no AMR is going to this node for the next pick in the pickrun, the value is -10. If multiple AMRs, the minimum is selected.
- Expected time until two-step ahead AMR destination: Similar to the previous feature, but considers if an AMR is going to the node for the two-step ahead destination in its pickrun. This feature includes the expected travel time until the current destination, the expected pick time at the destination, the expected travel time until the next destination, the expected pick time at the next destination, and the expected travel time between the next and two-step ahead destination. If no AMR goes to this node for its two-step ahead destination, the value is -10. If multiple AMRs, the minimum is selected.
- Number of AMRs to aisle: The total number of AMRs going to a destination within the same aisle as the considered node.
- Number of waiting AMRs in the aisle: The total number of AMRs currently stopped at their destination in the same aisle as the considered node.

We used the expected times until the one- and two-step ahead AMR destinations instead of the distance because the picking times can greatly influence the process flows. For example, if the expected time of a pick is 10 seconds, this is equivalent to the time in which an AMR can move 15 meters. Oppositely, if the expected time of a pick is 30 seconds, this is equivalent to the time of moving 45 meters. With the expected times, this effect is considered in the feature representation, although we must remember it is no perfect prediction due to randomness, congestion, and picker availability.

Similar to the AMR information, we also included picker information in the state space. We used five related node features to capture the positioning of the pickers within the system:

- Picker location indicator: Binary value that is 1 if any picker other than the picker being allocated is currently at this node and 0 otherwise.
- Picker destination distance: If any picker has this node as destination, the distance in meters that this picker must still travel. If no picker is going to this node, the value is -10. If multiple pickers, the minimum distance is used.
- Number of pickers to aisle: The total number of pickers going to a destination within the same aisle as the considered node.
- Minimum distance of other pickers: The minimum distance of any other picker to this location. Here, the distance is the distance of the picker to its current destination plus the distance from its current destination to the considered node.
- Minimum expected time of other pickers: Similar to the feature above, but considers not only the distance but also the expected pick time. Here the expected time is the expected travel time of the picker to its destination plus the expected travel time from the current destination to the considered node and the expected pick time at the current destination.

The node location information describes the regions in which the nodes are located within the warehouse. We used two features to describe these locations:

- Aisle distance from origin: Describes how far toward the end of the warehouse the aisle is in which the node is located, scaled by the warehouse size. Thus, it is calculated using the aisle distance from the start of the warehouse divided by the total width of the warehouse.
- Node depth within aisle: Describes how far toward the beginning or end of the aisle a node is located. It is calculated by taking the depth in meters within an aisle and dividing it by the total aisle length. Here the depth indicates the distance from the AMR entrance of the aisle.

These node location features are mainly included as they may be useful for policies to consider the routing of AMRs. Namely, if two nodes are within the same aisle, it may be beneficial to first pick the one at the aisle entry since the AMRs will afterward continue its route toward the other node at the aisle end. Similarly, if two nodes are in consecutive aisles, picking the node in the aisle closer to the start could be beneficial.

Lastly, the node neighborhood features are selected to capture the picking process occurring around the nodes. These may help capture high- or low-density pick areas. For this subcategory, we used the following features:

- Minimum distance next AMR destination: If any AMRs are going to this node, take the minimum of the distances in meters to the following destinations of these AMRs. If no AMR is going to this node or this was the last node on the AMR pickrun, the value is 0.
- Second smallest distance next AMR destination: If multiple AMRs are going to this node, take the second smallest value of the distances in meters to the next destinations of these AMRs. If less than two AMRs are going to this node or this was the last node on the AMR pickrun, the value is 0.
- Minimum distance two-step ahead AMR destination: If any AMRs are going to this node, take the minimum of the distances in meters to the two-step ahead destinations of these AMRs. If no AMR is going to this node or no two destinations are left in the AMR pickrun of at least one of the AMRs, the value is 0.
- Second smallest distance two-step ahead AMR destination: If multiple AMRs are going to this node, take the second smallest value of the distances in meters to the two-step ahead destinations of these AMRs. If less than two AMRs are going to this node or no two destinations are left on the AMR pickrun of at least two of the AMRs, the value is 0.
- Distance of closest other picker destination: The minimum of the distances from this node to the other nodes that are currently the destination of any of the pickers.
- Distance of closest unserved AMR destination: The distance in meters between this node and the closest other node that is the destination of an AMR and where no picker is already going.
- Distance of second closest unserved AMR destination: The distance in meters between this node and the second closest other node that is the destination of an AMR and where no picker is already going.

Workload Fairness Features The workload fairness features we used can also be split into two types. First, the node-specific features describe the workload characteristics at the nodes. Second, we included several "distributional" features that describe the current distribution of picker workloads. Although the distributional features are general and not node-specific, we included them as node features to facilitate the node-wise computations used in graph learning. Thus, these features contain the same value for each node.

We modeled the following node-specific features:

- Workload of picker at node: The total workload mass in kilograms that the picker at this node has handled so far subtracted by the mean workload of all pickers. If multiple pickers are at the node, which is rare, one is picked at random.
- Workload of picker going to node: The total workload mass in kilograms that the picker with this node as destination has handled so far subtracted by the mean workload of all pickers. If multiple pickers go to the node, which is rare, one is picked at random.
- Item workload at node: The mass in kilograms of a single item stored at this node.
- Waiting AMR workload: The total mass in kilograms that must be picked at this node to fulfill the picks of all AMRs that are currently waiting there.
- Destination AMR workload: The total mass in kilograms that must be picked at this node to fulfill the picks of all AMRs that are currently going toward this node but are not yet waiting there.
- Closest picker workload: The current total experienced workload of the closest picker to this node in terms of expected arrival time, subtracted by the mean picker workload.
- Second closest picker workload: The current total experienced workload of the second closest picker to this node in terms of expected arrival time, subtracted by the mean picker workload.

The distributional features are as follows:

- Current picker workload: The total experienced workload in kilograms of the picker currently being allocated, subtracted by the mean experienced picker workloads.
- Minimum workload: The minimum experienced workload in kilograms by any of the pickers, subtracted by the mean experienced picker workloads.
- 25th percentile workload: The 25th percentile workload in kilograms of all the picker workloads, subtracted by the mean experienced picker workloads.
- 75th percentile workload: The 75th percentile workload in kilograms of all the picker workloads, subtracted by the mean experienced picker workloads.
- Maximum workload: The maximum experienced workload in kilograms by any of the pickers, subtracted by the mean experienced picker workloads.

In the features related to experienced picker workloads, we corrected the picker workloads by subtracting the mean experienced workload. This helped to maintain stable features representing the distribution of workloads. Without correction, all workload values would continue to increase for the entire episode. This reduces learning stability and, hence, the potential performance of DRL agents.

In short, the state space consists of a graph representation, with nodes representing the warehouse locations and undirected edges representing how entities can move between these locations. For each node, we included 35 node features, split into 23 efficiency features and 12 fairness features. We did not include edge features.

Action Space

The action space of the problem is a discrete action space that consists of the nodes in the graph. Namely, a policy should assign a picker that places an allocation request to a single node, representing the new destination of the human picker. However, many locations are unsuitable options since many locations do not have AMRs going toward them. Therefore, we used a truncated action space. The truncated action space at any timestep consists of all locations that are the current or the next destination in the pickrun of any AMR and where no other human picker is already going. In the rare case that all other pickers are assigned to a future AMR destination, the action space is limited to only current AMR destinations to ensure that the system continues to fulfill picks. We did not allow locations that are two or more steps further in the AMR pickruns as they are far into the future, and the fulfillment of earlier picks and unexpected occurrences heavily influence the performance of these actions. Thus, the size of the action space is variable. However, the maximum size is achieved when all the AMR destinations and next destinations are unique locations. Then, the action space has a size of $2 \times \text{nr. of AMRs} - (\text{nr. of pickers} - 1)$.

Reward Function

The reward function is crucial for effective learning and must reflect the eventual objectives of the policy well. In our MOMDP, we propose one reward signal for efficiency and one for workload fairness.

The efficiency objective of our solution is to maximize the number of picks by the system in a certain amount of time. Hence, a policy must minimize the total time to perform a specific number of picks. To stimulate this, we used a penalty on the passed time. More specifically, at each RL step, the penalty is the difference between the total elapsed time in seconds until the current step and the total elapsed time until the previous step. As explained before, a transition constitutes the period between the previous picker allocation and the current picker allocation request. Thus, two consecutive steps are two consecutive picker allocation actions of the agent. Formally, the reward at step t is as follows, with τ_t indicating the system time at step t.

$$R_t^{\text{efficiency}} = \tau_{t-1} - \tau_t$$

By summing all rewards, the total efficiency reward of a full episode indicates the negative of the time that has passed between the first allocation and the fulfillment of the last pick. This directly maps to the objective of minimizing the total time.

The workload fairness reward must aim to maximize the fairness of the workload distribution over pickers. As explained in Section 4.1.2, several workload measures exist in the literature. However, these cannot be applied in our study. The traditional ergonomics measures used in order picking literature all require very detailed measurements and observations to estimate ergonomic strain. The measure by Larco et al. (2017) offers a good option for collaborative picking as they focus on product volume and mass. However, the estimated coefficients do not apply to our case. Therefore, in consultation with business stakeholders, we decided on using the total lifted product mass in kilograms as the workload measure.

To measure the fairness of the workloads, we outlined several options in Section 4.4. Since the policy cannot influence the total workload of the system, we did not opt for measures such as the generalized Gini function or Nash social welfare. Namely, these measures incorporate a balance between total workload and individual workloads, while the former is not affected by our decisions. Besides, we did not opt for the max-min fairness as this does not sufficiently consider the full distribution of the workloads. In consultation with business stakeholders, we selected the standard deviation of the workloads as our objective measure. The standard deviation is a widely known measure considering the full spread of all workloads. Besides, as noted in Section 4.4, it is not uncommon in optimization literature to use variance-based metrics, and the commonly used Jain's fairness index is also based on the standard deviation.

To transform the objective of minimizing the workload standard deviation, we used a similar approach as for the efficiency reward. Namely, the reward at each step is based on the increase or decrease of the standard deviation of the total carried product masses between the previous and current steps. So, at step t, the fairness reward is as follows, with σ indicating the standard deviation, $W_{k,t}$ indicating the total lifted mass by picker k until step t, and $|\mathcal{K}|$ the number of human pickers in the system.

$$R_t^{\text{fairness}} = \sigma(W_{1,t-1}, \dots, W_{|\mathcal{K}|,t-1}) - \sigma(W_{1,t}, \dots, W_{|\mathcal{K}|,t})$$

Hence, by summing the fairness rewards, the total fairness reward at the end of an episode amounts to the negative standard deviation of the picker workloads. Again, this relates directly to our objective of minimizing this standard deviation.

To combine the two rewards, the output vector of the reward function in the MOMDP at each step t is as follows.

$$\mathbf{R}_t = \begin{pmatrix} R_t^{\text{efficiency}} \\ R_t^{\text{fairness}} \end{pmatrix}$$

5.2.2 Learning Algorithm

In our study, we considered both single-objective learning for efficiency and multi-objective learning for balancing efficiency and workload fairness.

For single-objective learning, we used PPO with the clipped loss function described in Chapter 3. As discussed in Chapter 3, PPO is a policy-based method in which the policy network directly outputs the actions that should be taken. We used PPO for two main reasons. First, it is a state-of-the-art algorithm frequently shown to achieve good policies for various problems compared to other learning algorithms. Second, PPO can utilize parallel environments to collect experiences and use GPUs to efficiently make parallel decisions. This significantly reduces the training time needed to gather many experiences. Since we need many experience samples to gather the relatively long-term relations between picker allocation actions, this greatly increases the runtime.

Since we consider long episodes representing real-world picking processes, we used the actor-critic version of PPO. Actor-critic variants allow for updating the agent using the estimates of the critic without having to wait for full episode completions. This allows for faster learning using fewer samples and reduces the variance of the policy gradient (Silver, 2015). Thus, we used an actor network to take actions and a critic network that estimates the advantage function. Algorithm 5.1 gives an overview of the PPO algorithm we used. For the details of each step, we refer to the original paper by Schulman et al. (2017).

Algorithm 5.1 Pseudo-code for the PPO learning algorithm.

```
Input: Number of iterations $N$, initial actor parameters $\theta_0$, initial critic parameters $\phi_0$.
  $i \leftarrow 0$
  while $i < N$ do
      Collect trajectories by running policy $\pi_i = \pi(\theta_i)$ in parallel environments.
      Compute advantage estimates $\hat{A}_t$ using critic network $V_i = V(\phi_i)$.
      Update policy $\theta_k$ to $\theta_{k+1}$ via gradient descent on PPO loss $\mathcal{L}(\theta_k)$.
      Update critic $\phi_k$ to $\phi_{k+1}$ via gradient descent on mean-squared error loss.
  end while
```

For multi-objective learning, we built on top of the PPO algorithm. That is, we used the Prediction-Guided Multi-Objective Reinforcement Learning (PGMORL) algorithm that was proposed by J. Xu et al. (2020). This is one of the two state-of-the-art policy-based multi-objective RL algorithms. PGMORL is a multi-policy algorithm that produces a non-dominated set of policies. Hence, it allows us to explore the trade-offs between performance and fairness.

We did not opt for the meta-policy approach by F. Y. Liu and Qian (2021) as we do not desire to manually change the specific reward weights to any given value. PGMORL allows us to present a non-dominated set that showcases the trade-offs while not having the added difficulty of meta-learning. Due to the reliance of PGMORL on policy-based algorithms, we can transfer the PPO algorithm with the learning parameters and modeling approach from the single-objective method almost directly to the multi-objective algorithm. In the original paper, policies with continuous actions were tested. However, we apply our learning policies with the modeled discrete action space.

To understand the relevant aspects of the PGMORL algorithm, we outline the key steps below. For the detailed procedures, we refer to the paper by J. Xu et al. (2020). Algorithm 5.2 shows an overview of the method.

Algorithm 5.2 PGMORL algorithm.

```
Input: Number of parallel tasks $n$, number of warm-up iterations $m_w$, number of task iterations
    $m_t$, number of generations $M$.
  Initialize population $\mathcal{P}$, Pareto archive $EP$, and RL history $\mathcal{R}$.
  ▷ Warm-up Phase
  Generate initial task set $\mathcal{T} = \{\pi_j, \boldsymbol{\omega}_j\}_{j=1}^{n}$ using random policies $\pi_j$ and evenly distributed
  weight vectors $\boldsymbol{\omega}_j$.
  for task $(\pi_j, \boldsymbol{\omega}_j) \in \mathcal{T}$ do
      Run PPO for $m_w$ iterations.
      Collect result policy $\pi'_j$ and intermediate policies in $\mathcal{P}'$
      Store eval. rewards of old, new, and intermediate policies with weights $\boldsymbol{\omega}_j$ in $\mathcal{R}$
  end for
  Update $\mathcal{P}$ and $EP$ with $\mathcal{P}'$.
  ▷ Evolutionary Phase
  for generation $\leftarrow 1, 2, \ldots, M$ do
      Fit improvement prediction models for each policy in $\mathcal{P}$ using data in $\mathcal{R}$
      Select new task set $\mathcal{T} = \{\pi_j, \boldsymbol{\omega}_j\}_{j=1}^{n}$ based on improvement predictions.
      for task $(\pi_j, \boldsymbol{\omega}_j) \in \mathcal{T}$ do
          Run PPO for $m_w$ iterations.
          Collect result policy $\pi'_j$ in $\mathcal{P}'$
          Store eval. rewards of old, new, and intermediate policies with weights $\boldsymbol{\omega}_j$ in $\mathcal{R}$
      end for
      Update $\mathcal{P}$ and $EP$ with $\mathcal{P}'$.
  end for
```

In PGMORL, the core concept is to learn DRL policies using PPO training with a weighted-sum reward function $R_t = \boldsymbol{\omega}^T \mathbf{R}_t$, with $\boldsymbol{\omega}$ a weight vector and $\mathbf{R}_t$ the reward vector at time t. The algorithm must steer learning toward the weight vectors expected to stimulate policies that improve the current non-dominated set of solutions.

To do so, the algorithm starts with a warm-up phase. In this phase, n tasks are initialized. A task j consists of a policy π_j and a weight vector $\boldsymbol{\omega}_j$. The initial tasks consist of randomly initialized policy networks and evenly distributed weight vectors. For example, with six tasks and a two-dimensional reward function, the initial weight vectors are $(0, 1), (0.2, 0.8), \ldots, (0.8, 0.2), (1, 0)$. These initial tasks are trained using PPO for m_w warm-up iterations. The trained policies, intermediate policies, and their evaluation rewards are stored in a population. The population consists of both non-dominated and dominated policies. Based on the evaluation rewards, the intermediate Pareto archive is also updated to contain the non-dominated solutions. Thus, the warm-up phase outputs several baseline policies for different objective preferences.

Then, in the evolutionary phase, PGMORL uses improvement predictions to define new tasks that find better policies and improve the non-dominated set. To do so, at each generation, for each policy in the population $\mathcal{P}$, a prediction model is made to predict the rewards that can be achieved if the policy is trained using a specific weight vector. This four-parameter hyperbolic model for each policy and objective function is trained based on the data stored in history R. Only data samples in the neighborhood of the policy are used to fit the model. Using this prediction model, tasks (i.e., policies combined with a weight vector) are selected such that the predicted new non-dominated set improves the most, based on the hypervolume and sparsity.

Once these tasks have been selected, PPO training is done for m_w iterations, and the results are stored. Then, the evolutionary cycle repeats. To enable PPO learning without having to retrain a critic policy for each new weight combination for each policy, the critic is adapted to estimate a separate value for each of the objectives. In regular training, a single estimate of the weighted-sum reward would be used.

The final output of PGMORL is a set of non-dominated policies. These policies outline which trade-offs can be achieved between the objectives. This allows decision-makers to make an informed choice between several policies based on their preferences.

5.2.3 Deep Reinforcement Learning Agent

To learn the policies, PPO and PGMORL need an actor and critic network. The actor network interacts with the environment and performs the actual allocation decisions. The critic network is only used during training to evaluate the quality of selected actions. To do so, the actor must output the probabilities of each action. The critic network typically outputs one value to estimate the performance based on the whole input. For PGMORL, one value per objective is needed. In this section, we outline the network architectures used for our actor and critic network for single-objective and multi-objective learning.

Aisle-Embedding

As explained in Section 3.3, graph neural networks typically use a message passing principle to aggregate neighborhood information over nodes. However, capturing a large neighborhood is difficult as this requires many message passing steps and, hence, network layers. This leads to very deep networks with many parameters that are difficult and slow to learn. Oppositely, when smaller networks are chosen, smaller neighborhoods can be captured. In the warehouse scenario, we consider that nodes related to each other can be far apart. Namely, aisles can be 30 or 40 nodes deep, while nodes within an aisle are still frequently related to each other. To capture regional node dependencies and circumvent the limitations of message passing networks, we used a custom network architecture.

We refer to this architecture as an aisle-embedding network. The aisle-embedding network combines the idea of permutation invariant aggregation from graph networks with our knowledge of warehouse structures. Namely, we know that aisles form natural regions of nodes that are related within a warehouse. By aggregating the embeddings of the nodes within an aisle, we create an aisle-embedding that captures the regional information. Then, we combine the node-embedding with the aisle-embedding to calculate the final node values the actor uses to output the action probabilities. Formally, the aisle-embedding of an aisle A is calculated as follows, with $\boldsymbol{h}_v^l$ indicating the node embeddings at layer l and $\mathcal{V}_A$ the set of nodes within an aisle A.

$$\boldsymbol{h}_A^l = \Psi\left(\{\boldsymbol{h}_v^l | v \in \mathcal{V}_A\}\right)$$

Here, we used the mean as the permutation invariant function Ψ. Figure 5.6 illustrates the overall aisle-embedding network architecture. In the architecture, nodes are passed through an invariant feed-forward encoder, which applies a multilayer perceptron to each individual node without message passing. Then, the aisle-embedding is calculated for each aisle. This is

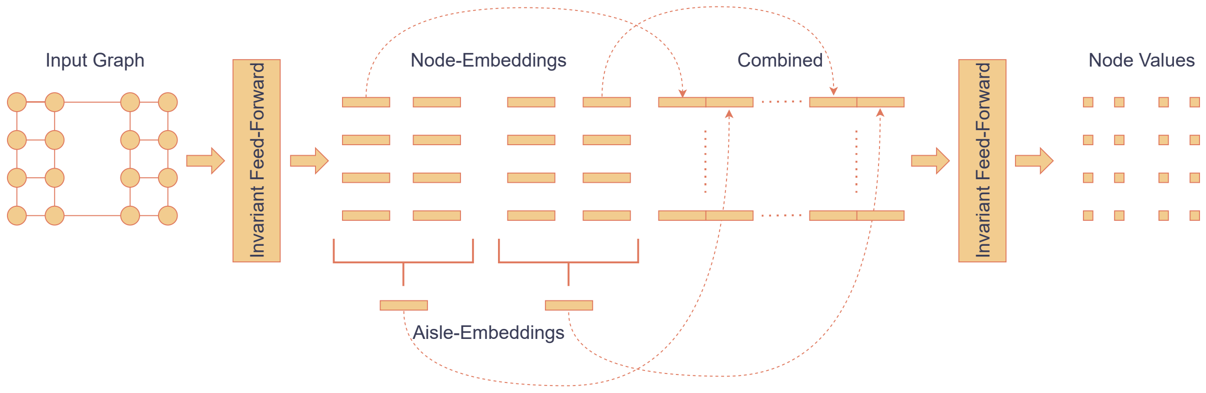


Figure 5.6: Illustration of the aisle-embedding architecture.

done using the node sets of the aisles, which are disjoint subsets of the total set of nodes $\mathcal{V}_G$. Afterward, the node-embedding and associated aisle-embedding of each node are stacked. This is then passed into a last set of invariant feed-forward layers to calculate the final node values. Thus, the final node value is calculated using the following function, with ψ_{actor} and ϕ_{actor} two feed-forward neural networks and $\mathcal{V}_{\text{aisle}(v)}$ the set of nodes within the aisle of node v.

$$Actor(v) = \phi_{\text{actor}}\left(\left[\psi_{\text{actor}}(\mathbf{x}_v), \text{AVG}(\{\psi_{\text{actor}}\left(\mathbf{x}_u)|u \in \mathcal{V}_{\text{aisle}(v)}\}\right)\right]\right)$$

To create the final action probabilities, the nodes are masked based on the truncated action space requirements by setting the values of invalid nodes to negative infinity. Lastly, the softmax function transforms the node values into action probabilities.

For the critic network, we used a simpler architecture, which we show in Figure 5.7. We used this simpler architecture as we found in preliminary tests that this network can approximate the value function well. As all node information is aggregated in the network, the aisle-embeddings are not needed to represent the value of the full graph accurately. The critic network applies an invariant feed-forward encoder. Then, the graph-embedding is calculated by summing the node-embeddings. Lastly, a final layer reduces the graph embedding to a single output value. Hence, the critic value for a graph G is calculated as follows, with ϕ_{critic} and ψ_{critic} two trainable networks.

$$Critic(G) = \phi_{\text{critic}}\left(\sum(\{\psi_{\text{critic}}(\mathbf{x}_v)|v \in \mathcal{V}_G\})\right)$$

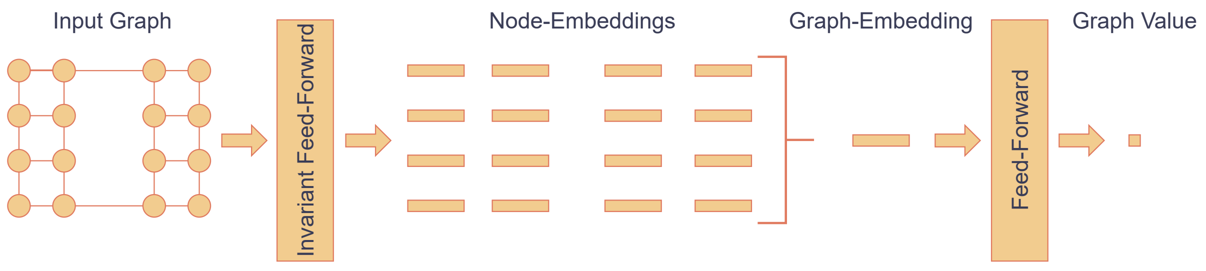


Figure 5.7: Illustration of the critic network architecture for single-objective learning.

Feature Separation

Existing works in multi-objective DRL focus on the learning algorithms. They use simple feed-forward neural networks in which all features are combined with fully-connected layers. They evaluate their methods on benchmark problems to test the algorithm performance without consideration of the network architecture. In our case, we have two distinct sets of features that relate to two different objectives. When combining these features in fully-connected layers, the

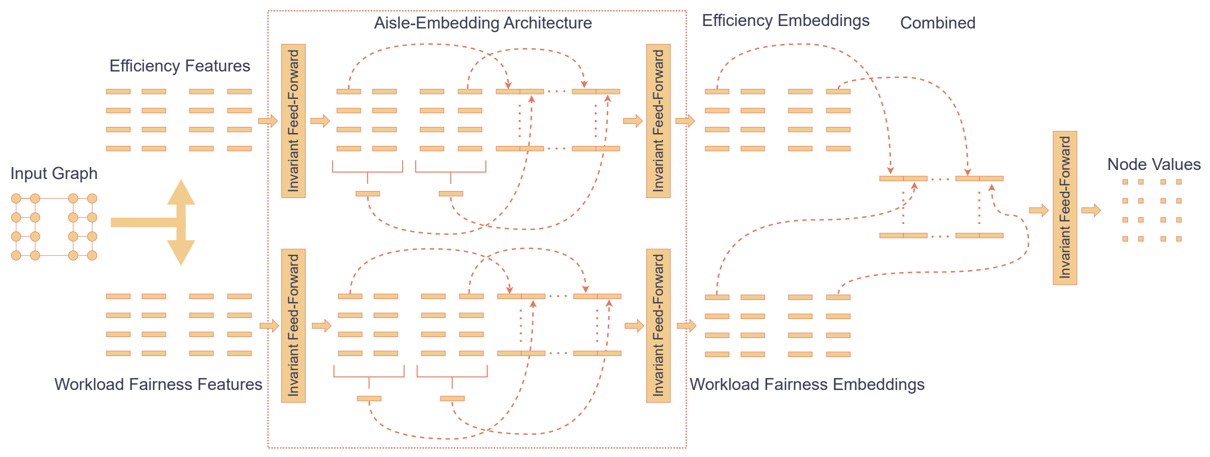


Figure 5.8: Illustration of the actor for multi-objective learning, combining feature separation with the aisle-embedding architecture.

parameter space becomes larger, while the features of the different categories do not have a clear connection. This can cause the network to learn noisy relations and create unstable learning.

To alleviate this, for multi-objective learning, we used a network architecture in which two feature categories are separated and treated independently for several layers before their high-level embeddings are combined into shared layers. This enables learning embeddings related to both feature categories without noise while the shared final layers capture the interactions between the fairness and efficiency objectives. Figure 5.8 outlines the feature separation architecture for the actor network. Combining the aisle-embedding structure with feature separation, the multi-objective actor network we used can be formulated as follows.

$$Actor_{\text{MO}}(v) = \gamma_{\text{actor}}\left([Emb_{\text{MO,fair}}(v), Emb_{\text{MO,effic}}(v)]\right)$$

Here, γ_{actor} is the feed-forward neural network that combines the embeddings of the feature categories, and $Emb^{\text{cat}}_{\text{MO}}$ represents the aisle-embedding network for a certain feature category and $\mathbf{x}^{\text{cat}}_v$ the feature vector of a category for node v.

$$Emb^{\text{cat}}_{\text{MO}}(v) = \phi^{\text{cat}}_{\text{actor}}\left(\left[\psi^{\text{cat}}_{\text{actor}}(\mathbf{x}^{\text{cat}}_v), \text{AVG}(\{\psi^{\text{cat}}_{\text{actor}}\left(\mathbf{x}^{\text{cat}}_u\right) | u \in \mathcal{V}_{\text{aisle}(v)}\})\right]\right)$$

Thus, using the aisle-embedding structure, the network creates a high-level embedding for both the fairness and efficiency features. Then, these are combined into a final set of node-wise feed-forward layers to create the final node value.

For the multi-objective critic, which is outlined in Figure 5.9, we combined the feature separation principle with the invariant feed-forward architecture. In this structure, the feature categories are first passed through separate feed-forward architectures. Then, the embeddings are combined into a final layer before being aggregated. Like in the single-objective network, the final graph-embedding is reduced to the output by a last network layer. The multi-objective critic outputs two values, one per reward. The multi-objective critic can be formalized as follows.

$$Critic_{\text{MO}}(G) = \gamma_{\text{critic}}\left(\sum\left(\left\{\phi_{\text{critic}}\left(\left[\psi^{\text{effic}}_{\text{critic}}(\mathbf{x}^{\text{effic}}_v), \psi^{\text{fair}}_{\text{critic}}(\mathbf{x}^{\text{fair}}_v)\right]\right) | v \in \mathcal{V}_G\right\}\right)\right)$$

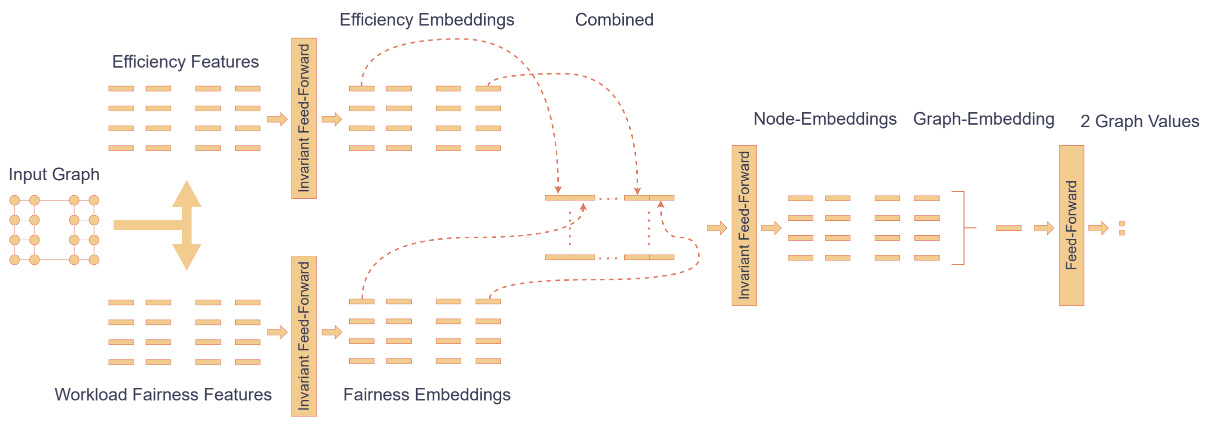


Figure 5.9: Illustration of the critic for multi-objective learning, using feature separation.

Chapter 6

Experiment Setup

This chapter outlines our setup to train and evaluate our solution. In Sections 6.1 and 6.2, we will introduce the warehouse instances that we used in the experiments and the benchmark methods, respectively. Consequently, in Section 6.3, we will discuss the details of the single-objective efficiency optimization experiments. We tested the single-objective optimization as this offers us insights into the most efficient achievable policies. These results can be used as a reference to compare with the multi-objective solutions. This allows us to find the "price of fairness". Lastly, in Section 6.4, we will present the specifics of the multi-objective optimization experiments.

6.1 Warehouse Settings

To train and test our methods, we used various warehouse setups. We picked a wide range of sizes, numbers of pickers, and numbers of AMRs in consultation with business stakeholders. These different warehouses were used to evaluate learning performance on problems of different scales as well as how the learned policies scale to different environments. Table 6.1 gives an overview of the warehouses.

The smallest warehouse we considered (XS) was only used for evaluation in a deterministic setting, as we will discuss in Section 6.3.3. This system only contained 4 pickers and 7 AMRs while having 7 aisles with 7 pick locations per side. Our study's smallest stochastic warehouse type (S) had 10 aisles, with 10 pick locations on both sides. Thus, it had a total of 200 pick locations. We used 10 human pickers and 25 AMRs. The medium warehouse type (M) contained 15 aisles of 15 pick locations on each side, totaling 450 pick locations. With 20 human pickers and 50 AMRs, this warehouse system was about twice as large as type S. The large warehouse type (L) consisted of 25 aisles with 25 pick locations per side, which is a total of 1250 pick locations. The default number of pickers and AMRs we used for this warehouse type was 30 human pickers and 90 AMRs. Finally, the largest warehouse system (XL) that we considered, consisted of 35 aisles with 40 pick locations per side. In this system, we used 60 pickers and 180 AMRs. Unless stated otherwise, we refer to these warehouse settings for each type in the remainder of this report. The simulation model with the mentioned settings explained in Section 5.1 was used for all experiments. For each warehouse type, to complete one episode, we set a predefined number of picks that had to be completed. These picks, distributed over many pickruns, had to be completely fulfilled to end an episode. We set these numbers such that an episode simulated a few hours of the warehouse process, which is roughly equivalent to the length of a typical work shift. For warehouse S this number is 5000 picks; for warehouse types M and L 7500; and for warehouse XL 15000. As XS was only used for deterministic evaluation of small instances, we did not set such a large number. We will explain these episode lengths in Section 6.3.3.

Table 6.1: Overview of the warehouse types considered in the experiments.

Warehouse Type	Aisles	Aisle Depth	Locations	Pickers	AMRs	Picks
XS	7	7	98	4	7	≤ 100
S	10	10	200	10	25	5000
M	15	15	450	20	50	7500
L	25	25	1250	30	90	7500
XL	35	40	2800	60	180	15000

6.2 Benchmark Methods

To compare the quality of our policies, we used two benchmark methods. First, the greedy baseline considers a trivial strategy. In this method, we always assign a picker to the closest available location where an AMR is going, and no other picker is already going. This relies on the assumption that assigning a picker to the closest possible option allows this picker to fulfill a new pick the fastest. However, the method ignores the overall system efficiency.

The second benchmark that we used is the method that is currently being used by Vanderlande to evaluate collaborative picking systems and was developed based on simulation tests. We will refer to this as the VI Benchmark. This rule-based method considers the distance of potential picks and also tries to spread the pickers across aisles. It uses a slightly different interface than our proposed picker optimizer. Namely, within aisles, pickers act more independently while they are assigned to a new aisle once they reach the end of an aisle. More concretely, each picker is located within an aisle. Within this aisle, a picker checks 10 locations ahead or backward whether any AMRs are waiting for a human picker to perform a pick. If so, the picker moves to the closest waiting AMR. If, during this walk, a picker encounters another AMR that is waiting, it will first pick the items for this encountered AMR. If there are no AMRs within the checked zone, the picker moves a step toward the end of the aisle (i.e., in the allowed AMR travel direction). Here, the picker checks its zone again, and so on. Once a picker reaches the end of an aisle, he needs to be assigned to a new aisle. This occurs by assigning a cost to each aisle. This cost is as follows.

$$\text{Aisle Cost} = \text{Nr. of aisles difference} - \text{Nr. waiting AMRs}$$

The aisle with the lowest cost is selected. Consequently, the picker moves to this aisle, where it repeats its process of checking for waiting AMRs and picking items for these AMRs.

As there are no current online optimization methods in collaborative picking or related problems that we can apply, we do not consider other benchmark methods. In addition, we do not use specific multi-objective benchmarks. Other popular multi-objective optimization methods, such as evolutionary methods, construct full episode solutions in one go based on complete information. Hence, these do not satisfy our solution requirements and, therefore, do not provide valid benchmarks. In Section 6.4.3, we will explain how we evaluated the multi-objective DRL solutions based on the picking time and workload standard deviation objectives.

6.3 Efficiency Optimization

The first part of our experiments considered single-objective optimization of the picking efficiency. In these experiments, we tested the possible performance we can achieve without focusing on workload fairness. Besides, we tested how well the learned policies scale to different warehouse environments and different amounts of pickers and AMRs in the systems. For these experiments, we only included the efficiency features in the state representations and only used the efficiency reward function. We evaluated all policies on the total time to complete an episode in seconds.

6.3.1 Network Architectures

To learn the single-objective efficiency policies, we used the actor network with the aisle-embedding structure and the critic network with invariant feed-forward encoder as outlined in Section 5.2.3. We implemented all networks in our study using Pytorch (Paszke et al., 2019) and Pytorch Geometric (Fey & Lenssen, 2019).

In the actor network architecture, to generate the node-embeddings, we used two fully-connected layers with 64 neurons and the Leaky ReLU activation function, followed by a fully-connected layer with 16 neurons. We used $\alpha = 0.01$ for all Leaky ReLU activation functions in our models. The output was used to create the aisle-embeddings, and the node- and aisle-embeddings were stacked to get node representation vectors of length 32. To create the final node values from the vectors, we used two fully-connected layers with Leaky ReLU activation and 64 and 16 neurons, respectively, followed by a last fully-connected layer with one neuron. Then, the invalid nodes were masked by setting their values to negative infinity, and the softmax function was used to generate the action probabilities.

In our critic networks, we used three fully-connected layers with the Leaky ReLU activation function to create the node-embeddings. For the first two layers, we used 64 neurons, while the third layer had 16 neurons. Then, after aggregating the node-embeddings to form the graph-embedding, we used one fully-connected layer with one neuron to get the value estimate.

6.3.2 Learning Algorithm

As explained in Section 5.2.2, we used PPO for single-objective learning. We adopted the Tianshou package (Weng et al., 2022) for our PPO implementation.

We used 64 parallel environments to collect samples. Per PPO iteration, we set the number of collected experience tuples per environment to 400. Thus, full environment episodes span across multiple iterations. This number is relatively large as the experiences must capture sufficiently long sequences from various warehouse environments to reliably derive long-term rewards. For the loss function, we used a clipping parameter ϵ of 0.2 and set the entropy coefficient c_{ent} to 0.01 after some preliminary tests. To update the network, we used the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 5×10^{-4}. Per PPO iteration, we performed three epochs with a batch size of 128. We set the discount factor γ to 0.995. Per training run, we trained for 150 epochs for warehouse types XS and S, 200 epochs for types M and L, and 400 epochs for type XL which showed convergence. Lastly, during training, we sampled the actions of the policies based on the output probabilities. This facilitates exploration as opposed to greedily selecting the action with the highest value. Oppositely, we always picked the actions with the highest action probability for evaluation, as these are considered the best actions by the agent, and, in evaluation, we want to maximize performance. In some DRL use cases, multiple solutions can be generated using sampling, after which the best is selected. However, in our use case, the picker allocation policy must immediately decide upon an allocation and cannot evaluate multiple options. We used this selection strategy for both single-objective and multi-objective evaluations in all experiments. All single-objective PPO training runs were performed on a computing instance with an Intel Xeon Platinum 8360Y processor using 16 CPU cores and an NVIDIA A100 GPU.

6.3.3 Experiment Description

Performance Evaluation on Fixed Warehouse Sizes

In our first main experiment, we tested the performance of our DRL methods on stochastic warehouses with uncertainty and congestion, as explained in the simulation description in Section 5.1. This experiment aims to test how much efficiency improvement we can achieve in warehouses with the same system parameters as our training instances. To do so, we trained one policy for each of the warehouse types S, M, L, and XL, using the previously described settings. We trained these policies using randomly initiated warehouse instances within the defined parameters. Thus, each training warehouse has a unique set of pickruns and a randomly initiated product spread. In all remaining experiments, we also used warehouses with randomly initiated pickruns and product allocations

For evaluation, we tested each policy on 100 random instances of its associated warehouse type. Thus, we tested the policy trained on type S on 100 warehouses of type S, and likewise for the other warehouse types. We used 100 evaluation instances for this experiment and all further experiments. Again, all evaluation instances in this and further experiments have a unique set of pickruns and allocation of products through the warehouse. We compared the performance of these policies with the benchmark methods.

Picker/AMR Transferability

For practical applicability, the developed policies must be capable of handling various amounts of pickers and AMRs in the system. To test how well the policies can adapt, we tested each policy on a variety of picker and AMR numbers. For these numbers, we changed the amounts of pickers and AMRs, as well as the ratio between them. We used the policies we trained in the previous experiment with the settings from Table 6.1. Thus, these policies were trained on a fixed amount of pickers and AMRs. Again, we compared the performance with the greedy and VI Benchmark solutions over 100 evaluation episodes. We tested the following picker/AMR combinations.

- Warehouse type S: 7/15, 10/20, 10/30, 15/25, 15/30, 15/35.
- Warehouse type M: 15/35, 20/40, 20/60, 30/50, 30/60, 30/70.
- Warehouse type L: 25/60, 30/70, 30/100, 40/90, 40/100, 40/110.
- Warehouse type XL: 50/120, 60/140, 60/200, 80/180, 80/200, 80/220.

Warehouse Size Transferability

Aside from evaluating the transferability to picker and AMR numbers, we also evaluated how well policies adapt to different warehouse sizes. If policies adapt well to different warehouse sizes, this allows easier distribution of policies to new warehouses or when warehouse environments are downsized or expanded. To do so, we again used the trained policies from the performance evaluation experiment. Then, we tested the performance of each policy on 100 evaluation episodes for each warehouse type S, M, L, and XL. We compared how well the policy performance transferred compared to the policies that were trained for each specific warehouse size.

Deterministic Instance Evaluation

The fourth experiment we performed was comparing our DRL agent with the optimal solution that can be found with complete information in a deterministic environment. This allows us to understand how close the policies can get to optimal solutions. In this experiment, we tested several instances of warehouse type XS with fully deterministic settings. We used fixed picking times of 7.5 seconds, fixed picker and AMR speeds of 1.25 m/s and 1.5 m/s, respectively, no overtaking penalties, and no random disruptions.

We solved these instances by implementing the MILP model from Equations 2.1-2.21, without the workload fairness considerations (i.e., without Equations 2.2 and 2.15). We implemented and solved the MILP instances using the Gurobi solver (Gurobi Optimization, LLC, 2023). We used their indicator constraints option to solve the constraints with big-M notation as efficiently as possible. For each instance, we ran the Gurobi solver for 20 hours on a computer with an AMD Rome 7H12 CPU instance with 64 CPU cores.

The instances we used all contained one pickrun per AMR that had to be completed. For each instance, we sampled random pickruns of lengths between 9 and 14 items. We tested two different instance types. First, we tested instances with diverse starting positions. For diverse starting position instances, we cut off the sampled pickruns using random uniform selection to ensure that AMRs are spread through the warehouse. Second, we tested instances without diverse starting positions. In these instances, all AMRs start a full pickrun, meaning they are initialized closer to each other at the beginning of the warehouse.

We trained one DRL agent for the diverse starting scenarios and one for the non-diverse starting scenarios. In training, we used random warehouse instantiations with the same overall warehouse parameters. Thus, the DRL policies were not explicitly trained for the specific testing instances but for generalizable performance in warehouse instances with the same settings. We evaluated the DRL policies on each evaluation instance. In addition, we evaluated the greedy and VI Benchmark methods on these instances.

Architecture Comparison

To further show the quality of our proposed methodology, we compared our method with three other neural network architectures, being an invariant feed-forward neural network with node-wise forward passing of information and no information exchange, a GCN network, and a GIN network.

For the invariant feed-forward actor network, we used two fully-connected layers with Leaky ReLU activation and 64 neurons, followed by a fully-connected layer with 16 neurons and Leaky ReLU and a last layer with one neuron that represents the node value, which is masked and passed through the softmax function with all nodes. We used the same critic as described in Section 6.3.1.

For the GCN actor, we used four consecutive GCN layers with 64 output channels and Leaky ReLU activation function, followed by two fully-connected feed-forward layers of 64 and 16 neurons with Leaky ReLU, and a last fully-connected layer with one neuron. The GCN critic also had four consecutive GCN layers with 64 output channels and Leaky ReLU activation function, followed by two fully-connected feed-forward layers of 64 and 16 neurons with Leaky ReLU. These were followed by the summation aggregation and one final linear layer with one neuron to output the graph value.

The GIN networks had the same structure as the GCN networks with GIN layers instead of GCN layers. For each GIN layer, we used a multilayer perceptron with two fully-connected layers of 64 neurons with Leaky ReLU activation. All policies were trained with the learning parameters from Section 6.3.2. We performed the training and evaluation procedure as described for the performance evaluation experiment for warehouse sizes S, M, and L for each of the architectures to compare their performance.

Aside from the performance, we also tested the inference time of each architecture. Namely, the picker optimizer must be deployed in real-time with potentially several allocation requests per second at busy moments. Therefore, inference time must not be too long for the DRL agents to be applied. For each architecture, we measured the inference time of a thousand actions on each warehouse type. We did so on a laptop with an Intel Core i7-9850H processor and an NVIDIA Quadro T2000 GPU, as we ideally want the picker optimizer to be capable of running without extreme hardware requirements.

6.4 Multi-Objective Optimization

After evaluating the performance of DRL on the single-objective problem of optimizing efficiency, we evaluated the performance of the multi-objective algorithms. For these experiments, we used both the efficiency and workload fairness features in our state representation and incorporated both reward functions. The following sections outline the architecture specifics, learning algorithm settings, and experiments we used to train and evaluate our multi-objective solutions.

6.4.1 Network Architectures

For multi-objective learning, we implemented the architectures described in Section 5.2.3. In the actor network, to create the efficiency and fairness embeddings, we used two aisle-embedding architectures with the same structure, with one handling the efficiency features and the other handling the fairness features. This architecture is the same as we used in the single-objective critic, except for the last fully-connected layer with one neuron. Instead, we stacked the efficiency and fairness embeddings of 16 dimensions to create the combined node-embeddings of 32 dimensions. These node-embeddings were transformed to node values by a fully-connected layer with 16 channels and Leaky ReLU activation followed by a final layer with one neuron. Again, these final node values were masked, and the softmax function was applied to get the action probabilities.

In the critic network, we used the same principle of applying the single-objective architecture to both the efficiency and workload fairness features. Thus, we use the same three fully-connected layers. Then, the resulting embeddings with 16 layers were stacked to form a 32-dimensional embedding. These 32-dimensional embeddings were passed through a 16-neuron fully-connected layer having Leaky ReLU activation and aggregated using summation to get the aisle embedding. Then, we used one final linear layer of 2 neurons to get the value estimate for the two reward functions.

6.4.2 Learning Algorithm

For our PGMORL implementation, we adapted the original code from J. Xu et al. (2020) to handle graph states and a discrete action space and enable integration with our simulation interface. Besides, we integrated Dask (Dask Development Team, 2016) into the code to enable nested multiprocessing for the PPO tasks and the parallel simulation environments within the PPO instances, allowing more efficient training and, therefore, reduced training times. We ran PGMORL for warehouse types S, M, and L. We did not include warehouse type XL due to the high computational costs.

For the PPO tasks, we used the hyperparameters described in Section 6.3.2. As explained by J. Xu et al. (2020), having more parallel tasks is likely to increase the quality of the found non-dominated set. However, it also increases computational costs. Therefore, we used 6 tasks, similar to the default value in the original paper. For warehouse type S, we set the number of warm-up iterations m_w to 80 and the number of task iterations between evolutionary steps m_t to 12. The number of warm-up iterations must be sufficiently high such that distinguishable policies with appropriate performance can be used as a base for the further evolution of different policies. For warehouse types M and L, we set m_w and m_t to 128 and 16, respectively, as the larger warehouses generally require more learning iterations to update. For warehouse S, we collected 7 million steps per task before termination, while for warehouse types M and L, we used 7.5 million steps per task before termination. We determined these numbers of steps by inspecting the generally required number of update iterations for convergence. We performed 20 evaluation episodes once every 6 PPO iterations for type S and once every 8 iterations for types M and L. These evaluation results are used by PGMORL to determine the non-dominated set and perform the evolutionary prediction. Lastly, for multi-objective training, we scaled the efficiency rewards down to get both reward functions to similar scales. Although not strictly necessary, this aids in finding better weight vectors oppositely to when rewards are of different magnitudes. Namely, the initial policy set will have a better spread in behavior, leading to quicker identification of valuable weight combinations. For all other algorithm settings, we used the default values from the original implementation by J. Xu et al. (2020).

We ran the PGMORL implementations on a machine with a 32-core Intel Xeon Platinum 8360Y processor and an NVIDIA A100 GPU.

6.4.3 Experiment Description

For the multi-objective evaluation, we followed a similar structure as the single-objective experiments. Namely, we first tested the performance on similar warehouses, followed by an analysis of the transferability to different warehouse sizes and numbers of pickers and AMRs. Lastly, we also performed an architecture comparison.

Performance Evaluation on Fixed Warehouse Sizes

For performance evaluation, we ran the PGMORL algorithm for the warehouse types S, M, and L. This resulted in a set of non-dominated policies for each warehouse type. We gathered these policies and ran 100 evaluation episodes for each of these policies on the same warehouse type as they were trained on. These policies each form a front that shows the achievable trade-offs between fairness and performance. We assessed the quality of this front by inspecting the performance of the policies in terms of total picking time and the standard deviation of the workloads. We compared these with the greedy method, VI baseline, the pure efficiency policies from the single-objective optimization, and pure fairness policies.

The pure fairness policies were trained using the network architecture and learning parameters of the single-objective optimization setup with just the workload fairness features and reward function. One slight difference in the training setup is that we used a learning rate of 5×10^{-5}, and for warehouse type S we replaced the layers of 64 neurons with layers of 32 neurons.

Picker/AMR Transferability

To test transferability for picker/AMR numbers of multi-objective policies, we applied a similar procedure as for the single-objective optimization. Namely, we used the policies trained in the performance evaluation experiment. We evaluated each of these policies on 100 evaluation episodes for different numbers of pickers and AMRs than they were trained on within the same warehouse size and compared the results with the performances of the greedy method, VI benchmark, pure efficiency policy, and pure fairness policy. Because there are many policies to be evaluated, we only tested warehouse sizes S and L and the more extreme pick/AMR ratios for these experiments. If these sizes and more extreme picker/AMR numbers perform well, in combination with the single-objective results, we still have sufficient proof that the less extreme numbers are also handled well. Accordingly, we used the following picker/AMR numbers.

- Warehouse type S: 7/15, 10/30, 15/35.
- Warehouse type L: 25/60, 30/100, 40/110.

Warehouse Size Transferability

We also evaluated the warehouse size transferability similarly to the single-objective problem. Namely, we evaluated the trained multi-objective policies for warehouse types S, M, and L on all warehouse types S, M, L, and XL. Like in the other multi-objective experiments, we compared the formed front with the pure efficiency and fairness policies, as well the greedy baseline and VI Benchmark.

Architecture Comparison

To outline the value of our proposed network architectures for multi-objective DRL, we also compared different multi-objective architectures. Due to the large computational requirements of PGMORL, we did not perform the architecture comparison on full PGMORL training. Instead, we trained and evaluated the architectures on varying weight vectors. Since PGMORL internally uses weighted sums for learning, having good performance when training for pre-specified weighted sums indicates the performance level that can be achieved by PGMORL.

We compared our multi-objective architecture with three other architectures. First, we compared the performance with the single-objective aisle-embedding architecture from the single-objective experiments. Second, we used the invariant feed-forward architecture that we also used as a comparison for the single-objective performance. Third, we tested an invariant feed-forward architecture with feature separation. This network applies the structure of our multi-objective architecture, but instead of aisle-embeddings, we used an invariant feed-forward structure to create the efficiency and workload fairness embeddings. This invariant feed-forward structure consisted of two fully-connected layers with 64 neurons and a Leaky ReLU activation function followed by one fully-connected layer with 16 neurons.

We trained these networks using the PPO settings described in Section 6.3.2. We trained and evaluated these networks for three different weight vectors to create a weighted-sum reward of the scaled-down efficiency reward and the workload fairness reward. These weight vector settings were $(0.1, 0.9)$, $(0.5, 0.5)$, and $(0.9, 0.1)$. The warehouse types that we used for this experiment were types S and M.

Similar to the single-objective architectures, we also tested the inference time of the multi-objective architectures. We used the same approach as explained in Section 6.3.3 for the single-objective experiment.

Chapter 7

Results

In this chapter, we outline the results and implications of our experiments. In Section 7.1, we will explain the results of the single-objective experiments for efficiency optimization. Consequently, in Section 7.2, we will consider the results of the multi-objective optimization experiments.

7.1 Efficiency Optimization

7.1.1 Performance Evaluation on Fixed Warehouse Sizes

We present the performance evaluation results in Table 7.1. Note that the values in this and all other tables indicate the total picking time to complete an episode in seconds, and thus, the lower the values, the better. This table shows that the DRL policies outperformed the greedy and VI benchmark policies by a clear margin for all warehouse sizes. For the smallest warehouse size, the performance improvement over the VI benchmark was 14.9% percent, while for the larger warehouse sizes DRL achieved over 30% faster completion times, with improvements of 31.7% and 33.6% for warehouses L and XL, respectively. The greedy baseline performed slightly worse than the VI Benchmark, although the differences were just a few percent.

Table 7.1: Performance evaluation of DRL, greedy, and VI Benchmark policies on picking efficiency. The values indicate the average picking time in seconds over 100 evaluation episodes, with ± indicating the width of the 95%-confidence intervals. The % indicates the percentage improvement over the VI Benchmark, with a positive percentage indicating an improvement and, thus, lower picking times. The bold markings indicate the best performance values per warehouse size.

	DRL		Greedy		VI Benchmark
Warehouse	Picking Time	%	Picking Time	%	Picking Time
S	**8586 ± 62**	**14.9**	10619 ± 59	−5.3	10087 ± 58
M	**8425 ± 46**	**21.0**	11023 ± 58	−3.3	10669 ± 41
L	**6540 ± 37**	**31.7**	9823 ± 33	−2.7	9569 ± 61
XL	**9010 ± 21**	**33.6**	13972 ± 44	−3.0	13570 ± 72

These findings demonstrate that the DRL policies perform well as picker optimizer agents in collaborative order picking warehouses. The outcomes of this experiment show that the DRL policies can achieve good efficiency in realistically-sized warehouse instances with randomness, congestion, and unexpected interruptions.

7.1.2 Picker/AMR Transferability

The previous results showed that DRL policies could improve the system efficiency by a clear margin. For these DRL policies to be helpful in practice, they must continue this efficiency improvement when the numbers of pickers and AMRs in the system change, as real-world warehouses do not always have the same amounts of workers in the system. Table 7.2 shows the results of the transferability analysis of the DRL policies to the different picker and AMRs numbers.

In these numbers, a similar result is visible that the DRL approach outperformed both greedy and the VI Benchmark for each combination of pickers and AMRs in each warehouse size. Again, the performance improvement over the VI Benchmark was the largest for the larger warehouses. Namely, for warehouse type S we found improvements between 13.0% and 24.7%, for warehouse type M between 16.0% and 26.2%, and for warehouses L and XL between 32.5% and 41.5% over the VI benchmark. Remarkably, the relative improvement of the picking times was better for most picker/AMR ratios than the improvements in Table 7.1 on the ratios we trained on. On warehouse types S and M, the advantage was only smaller for the ratios 10/30 and 20/60, respectively. These are ratios with a relatively low number of pickers and, in comparison, many AMRs. For all other combinations, the percentage improvement over the VI Benchmark was roughly equal or better. This shows that, whereas the VI Benchmark efficiency deteriorates when the crowdedness levels in the warehouse become either small or larger, the DRL policy continues to achieve good results. Thus, the DRL policy can adapt to extremer warehouse occupation levels more efficiently.

These results can be used in two ways. First, the completion times can be improved using the DRL policies to handle larger capacities. Second, one can achieve equal performance while reducing the number of used resources to save costs. For example, the results of warehouse type XL show that the DRL policy could achieve roughly similar picking efficiency with 50 pickers and 120 AMRs as the VI Benchmark achieved with 80 pickers and 220 AMRs.

In several cases, the greedy baseline performed slightly better than the VI Benchmark, with the best of the two alternating for different settings. Especially for the larger warehouse size with extremer picker/AMR numbers, the greedy policy seemed more suitable. However, the greedy baseline, like the VI benchmark, did not get close to the DRL performance for any problem instance.

7.1.3 Warehouse Size Transferability

We saw that policies adapt well to different numbers of pickers and AMRs. This section discusses how well they transferred to other warehouse sizes. These results are outlined in Table 7.3.

The results in this table reveal that the policies adapted well to different warehouse sizes. We see that the policy trained on warehouse type S achieved an average total pick time of 6877 seconds on type L compared to the 6540 seconds reached by the policy trained on warehouse L. Thus, while being developed for a warehouse with over 6 times fewer pick locations and roughly 3 times as little pickers and AMRs, it only performed about 5% worse. Similarly, the policies also scaled down well to smaller warehouses. The policy of warehouse type L achieved an average completion time of 8875 seconds compared to the 8586 seconds of policy S. This is a performance difference of just over 3%. Remarkably, policy L (8567 seconds) outperformed policy XL (9010) on all instance sizes. Policy L achieved an improvement of 36.9% over the VI benchmark, compared to the 33.6% improvement of policy XL. This indicates that training for increasingly larger warehouse sizes is not necessary to get good performance on those warehouse sizes. In larger warehouse sizes, the action space is bigger, and therefore, learning can be slower and harder to fine-tune to get the last percentage improvements. Learning for many more iterations might eventually bring better results, but this is not guaranteed and the learning is substantially

Table 7.2: Performance of DRL policies trained for efficiency on warehouse types S, M, and L, given varying other combinations of the number of pickers and AMRs within their respective warehouse sizes. The values indicate the picking time in seconds. The ± indicates the 95%-confidence interval. The % indicates the percentage improvement over the VI Benchmark, with a positive percentage indicating an improvement and, thus, lower times. The bold markings indicate the best performance values per warehouse setting.

(a) Warehouse type S.

	DRL		Greedy		VI Benchmark
Pickers/AMRs	Picking Time	%	Picking Time	%	Picking Time
7/15	**12825 ± 83**	**17.1**	15166 ± 74	2.0	15472 ± 87
10/20	**9206 ± 51**	**19.4**	11274 ± 69	1.3	11420 ± 56
10/30	**8221 ± 54**	**13.0**	10283 ± 60	−8.8	9447 ± 52
15/25	**6737 ± 42**	**21.5**	7994 ± 40	6.9	8583 ± 36
15/30	**5930 ± 34**	**24.7**	7804 ± 55	1.0	7879 ± 46
15/35	**5938 ± 35**	**16.6**	7550 ± 44	−6.0	7121 ± 38

(b) Warehouse type M.

	DRL		Greedy		VI Benchmark
Pickers/AMRs	Picking Time	%	Picking Time	%	Picking Time
15/35	**11263 ± 50**	**21.2**	14240 ± 66	0.4	14297 ± 72
20/40	**9139 ± 46**	**23.5**	11331 ± 56	5.2	11952 ± 41
20/60	**7965 ± 48**	**16.0**	10569 ± 51	−11.4	9489 ± 53
30/50	**6795 ± 34**	**26.2**	8189 ± 46	11.0	9206 ± 34
30/60	**6293 ± 38**	**22.7**	7789 ± 31	4.3	8136 ± 50
30/70	**5944 ± 37**	**20.6**	7620 ± 29	−1.7	7490 ± 41

(c) Warehouse type L.

	DRL		Greedy		VI Benchmark
Pickers/AMRs	Picking Time	%	Picking Time	%	Picking Time
25/60	**8566 ± 34**	**36.6**	11852 ± 31	12.3	13512 ± 65
30/70	**7209 ± 26**	**37.7**	10120 ± 35	12.5	11563 ± 58
30/100	**6354 ± 65**	**32.5**	9980 ± 44	−6.0	9410 ± 62
40/90	**5659 ± 29**	**36.1**	7962 ± 29	10.1	8859 ± 68
40/100	**5279 ± 50**	**34.6**	8141 ± 44	3.4	8424 ± 52
40/110	**5059 ± 18**	**37.3**	7605 ± 27	5.8	8076 ± 45

(d) Warehouse type XL.

	DRL		Greedy		VI Benchmark
Pickers/AMRs	Picking Time	%	Picking Time	%	Picking Time
50/120	**12028 ± 23**	**40.2**	16816 ± 32	16.5	20142 ± 112
60/140	**10150 ± 20**	**40.7**	14312 ± 27	16.4	17118 ± 101
60/200	**9009 ± 44**	**35.6**	14293 ± 88	−2.2	13979 ± 87
80/180	**8106 ± 77**	**38.9**	11343 ± 30	14.6	13275 ± 83
80/200	**8011 ± 59**	**36.8**	11765 ± 52	6.4	12571 ± 91
80/220	**6947 ± 19**	**41.5**	10799 ± 40	9.1	11877 ± 84

Table 7.3: Performance of DRL policies trained on specific warehouse sizes when evaluated on a variety of warehouse sizes. The values indicate the picking time in seconds. The ± indicates the 95%-confidence interval. Policy X indicates the DRL policy trained on warehouse type X. The bold markings indicate the best performance values per warehouse size.

Warehouse	Policy S	Policy M	Policy L	Policy XL	Greedy	VI Benchmark
S	**8586 ± 62**	9190 ± 53	8875 ± 58	8986 ± 51	10619 ± 59	10087 ± 58
M	**7931 ± 42**	8425 ± 46	8064 ± 41	8220 ± 37	11023 ± 58	10669 ± 41
L	6877 ± 31	7190 ± 42	**6540 ± 37**	6877 ± 23	9823 ± 33	9569 ± 61
XL	9478 ± 20	11275 ± 33	**8567 ± 24**	9010 ± 21	13972 ± 44	13570 ± 72

slower, as we already trained the XL policy for twice as many steps as those for types M and L. The XL policy did transfer well to other warehouse sizes though, which again indicates the good transferability of policies. In addition to the comparative performances between each other, all DRL policies maintained a clear advantage over the greedy and VI benchmark results.

Thus, overall, the policies adapted well to different warehouse sizes. This enables easier deployment of policies to varying warehouses. Also, when a warehouse layout is (temporarily) changed, the policies can maintain good performance without needing to retrain and redeploy new policies. In addition, it is advantageous for the training process itself since one can train and evaluate different settings quicker on smaller warehouse instances and then scale the learned policies to larger warehouses.

In general, the experiments showed that, using our DRL method, we can achieve picker optimizer policies that increase the efficiency of collaborative picking systems compared to the greedy baseline and VI benchmark. For larger warehouses, we can achieve improvements in pick time of over 35%. The policies can also adapt to varying warehouse sizes while losing little performance.

7.1.4 Deterministic Instance Evaluation

Table 7.4 shows an overview of the results of the deterministic instance evaluation. The first thing that stands out is that the solver could not prove optimality within 20 hours. This can be seen by the MILP gap, which indicates the gap between a lower bound on the best possible time and the best solution that was found, not being 0%. This indicates the complexity of the problem, even in these minimalistic, deterministic instances.

In these results, we also find that the DRL solution got very close to the best found MILP solution in all cases. It even achieved better results for 5 instances. The most significant deviation in total picking time from the best MILP solution was just 21 seconds (227 vs. 206), indicating that DRL policies can consistently achieve good results. In addition, the DRL agents outperformed the greedy and VI benchmark methods for each instance. Compared to the greedy baseline, the improvement was generally not large. However, with such small instances, no congestion, and the results being so close to the MILP results, we did not expect a large deviation from the greedy method. That is, the greedy method optimizes in the short run without much consideration of other pickers, leading to fast initial picks for the pickers. In such short episodes, the long-term consequences cannot be affected too much as episodes end relatively quickly. This limits the extent to which the greedy solution can get into poor states. In addition, the greedy method experiences the converse effects of congestion less due to the lack of overtaking penalties. The VI benchmark results were worse than greedy and DRL. This makes sense as this method was developed to spread the pickers more evenly through the warehouse, while this may be less beneficial in short episodes without congestion effects. All in all, the deterministic instance results show that we can achieve good, near-optimal solutions using DRL that match the performance of the best solutions that were found using a solver with complete information

Table 7.4: Comparison of DRL, greedy, and VI Benchmark performance versus the best solutions found by the MILP solver for multiple small warehouse instances without randomness and uncertainty. The values indicate the total picking time in seconds for the specific problem instance. The MILP gap indicates the percentage gap between the lower bound estimate of the solver and the best solution that was found. The bold markings indicate the best performance values per problem instance.

(a) Instances of type XS with diverse starting.

Instance	DRL	Greedy	VI Benchmark	MILP	MILP gap (%)
1	154	154	355	**149**	17.8
2	**187**	190	397	**187**	6.0
3	155	167	299	**149**	12.2
4	**206**	248	269	212	17.5
5	227	236	277	**206**	15.9

(b) Instances of type XS without diverse starting.

Instance	DRL	Greedy	VI Benchmark	MILP	MILP gap (%)
1	**244**	262	355	**244**	28.2
2	**249**	253	297	271	28.1
3	**265**	272	299	267	29.3
4	**240**	257	269	245	22.8
5	**251**	255	277	260	30.9

of the problem instances. Therefore, these findings provide additional proof of the value of DRL in collaborative picking.

7.1.5 Architecture Comparison

Having established the main single-objective results of our method, in this section, we will present the architecture comparison.

Performance Comparison

Table 7.5 shows the performance of the different agent architectures we evaluated. These results demonstrate that our proposed aisle-embedding (AISLE-EMB) actor performed best on all warehouse sizes. Oppositely, the GIN and GCN structures both performed poorly compared to the aisle-embedding and invariant feed-forward (INV-FF) networks. Especially for the two larger warehouses, the performance difference is clear. For these scenarios, GIN and GCN did not outperform the VI Benchmark and greedy baseline scores shown in Table 7.1. Thus, the message passing networks cannot sufficiently extract useful regional information. Instead, the extra parameters introduced noise into the learning process, limiting their performance.

Table 7.5: Performance comparison of policies with different network architectures, focused on picking efficiency. The values indicate the average picking time in seconds over 100 evaluation episodes, with ± indicating the width of the 95%-confidence intervals. The bold markings indicate the best performance values per warehouse size.

Warehouse	INV-FF	AISLE-EMB	GIN	GCN
S	8689 ± 58	$\mathbf{8586 \pm 62}$	8869 ± 55	11677 ± 67
M	8628 ± 40	$\mathbf{8425 \pm 46}$	14151 ± 75	13851 ± 65
L	6602 ± 29	$\mathbf{6540 \pm 37}$	11723 ± 76	14419 ± 88

The difference with the invariant feed-forward network is smaller. Even though the aisle-embedding actor outperformed it on each warehouse type, the difference was within a few percent. This difference may be so slight because we used multiple node features that already describe regional information. Still, the aisle-embedding architecture increases performance for single-objective optimization over all other network structures.

Inference Time Comparison

We present the inference times of each of the architectures in Table 7.6. We find that the invariant feed-forward network achieved the quickest inference times, followed by the aisle-embedding structure. The GIN was about twice as slow as the invariant feed-forward network, and, in turn, the GCN was about twice as slow as the GIN actor. With all inference times being just a few milliseconds, all networks are suitable for deployment in real-life as performing multiple actions per second is no problem. We also find that the inference times scale slowly with increasing warehouse sizes, so it should not be an issue even for much larger warehouses.

Table 7.6: Inference time of DRL policies with different network architectures. The numbers represent the average inference time per action in milliseconds over 1000 collected actions, with ± indicating the 95%-confidence intervals. The bold markings indicate the best inference times per warehouse size.

Warehouse	INV-FF	AISLE-EMB	GIN	GCN
S	$\mathbf{1.44 \pm 0.03}$	2.16 ± 0.03	3.10 ± 0.03	6.02 ± 0.05
M	$\mathbf{1.53 \pm 0.03}$	2.22 ± 0.03	3.25 ± 0.03	6.28 ± 0.05
L	$\mathbf{1.53 \pm 0.03}$	2.41 ± 0.03	3.41 ± 0.04	6.65 ± 0.05
XL	$\mathbf{1.73 \pm 0.03}$	2.57 ± 0.04	3.77 ± 0.04	7.19 ± 0.06

7.2 Multi-Objective Optimization

In the previous section, we showed that DRL could achieve excellent results in terms of pick efficiency when performing single-objective learning for this objective. In this section, we will outline the results of the multi-objective learning experiments. We will discuss the quality and transferability of the policies that balance pick performance with workload fairness and which trade-offs can be achieved. We will consider the total order completion times in seconds and the workload standard deviation over pickers in kg.

7.2.1 Performance Evaluation on Fixed Warehouse Sizes

Table 7.7 and Figures 7.1, 7.2 and 7.3 show an overview of the results of the multi-objective learning experiments for warehouse sizes S, M, and L. Here, for both the picking times and workload standard deviations, lower values are better. For sizes S and L, PGMORL found a set of 6 non-dominated policies, whereas, for size M, 8 non-dominated policies were found. Note that the non-domination criterion was tested in the algorithm using 20 evaluation episodes. Hence, with the elaborate evaluation, it may be that in some cases a policy was slightly dominated by another policy that was very close in terms of performance.

In Figure 7.1, the non-dominated set of multi-objective policies forms a clear front toward the bottom left. The policies show a trade-off with a relatively sharp “angle.” This shows that we can decrease the workload standard deviation a lot before we sacrifice much pick efficiency or decrease the picking time by a lot before the workload fairness deteriorates. A policy that stands out is policy S3, which is represented by the dot in the bottom left of the front. This policy achieved both good completion times and good workload fairness. Namely, the average time to complete an episode was 9164 seconds, and the workload standard deviation was 66 kilograms, compared to 8586 seconds and 308 kilograms of the pure performance policy. Thus, by sacrificing just 6.7% of efficiency, this policy decreased the workload standard deviation

Table 7.7: Performance of the non-dominated set of multi-objective policies learned on different warehouse types. The picking time is the average number of seconds to complete an episode, and the workload fairness is the average standard deviation of the picker workloads in kilograms over 100 evaluation episodes. The ± indicates the 95%-confidence interval.

(a) Warehouse type S.

Policy	Picking Time	Workload Fairness
S1	15555 ± 125	41 ± 4
S2	12431 ± 86	43 ± 4
S3	9164 ± 60	66 ± 4
S4	9188 ± 55	114 ± 8
S5	9074 ± 60	118 ± 7
S6	9149 ± 68	167 ± 9
Pure Performance	8586 ± 62	308 ± 17
Pure Fairness	19962 ± 86	61 ± 9
Greedy	10619 ± 59	278 ± 15
VI Benchmark	10087 ± 58	442 ± 23

(b) Warehouse type M.

Policy	Picking Time	Workload Fairness
M1	22180 ± 65	86 ± 10
M2	18695 ± 174	100 ± 10
M3	14854 ± 74	103 ± 6
M4	14897 ± 153	140 ± 9
M5	9809 ± 169	154 ± 8
M6	9323 ± 136	223 ± 11
M7	8919 ± 51	266 ± 19
M8	8733 ± 52	460 ± 32
Pure Performance	8425 ± 46	302 ± 13
Pure Fairness	21793 ± 73	73 ± 4
Greedy	11023 ± 58	288 ± 9
VI Benchmark	10669 ± 41	548 ± 17

(c) Warehouse type L.

Policy	Picking Time	Workload Fairness
L1	25562 ± 92	70 ± 7
L2	15474 ± 62	65 ± 3
L3	8463 ± 32	72 ± 5
L4	8296 ± 78	76 ± 4
L5	8116 ± 62	139 ± 6
L6	7400 ± 220	226 ± 9
Pure Performance	6540 ± 37	228 ± 7
Pure Fairness	21525 ± 73	51 ± 3
Greedy	9823 ± 33	253 ± 7
VI Benchmark	9569 ± 61	472 ± 14

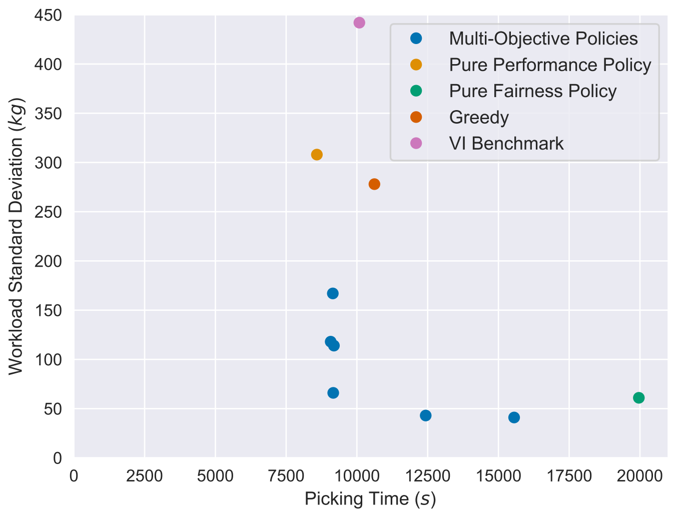


Figure 7.1: Performance evaluation of the non-dominated set of multi-objective policies learned on warehouse type S.

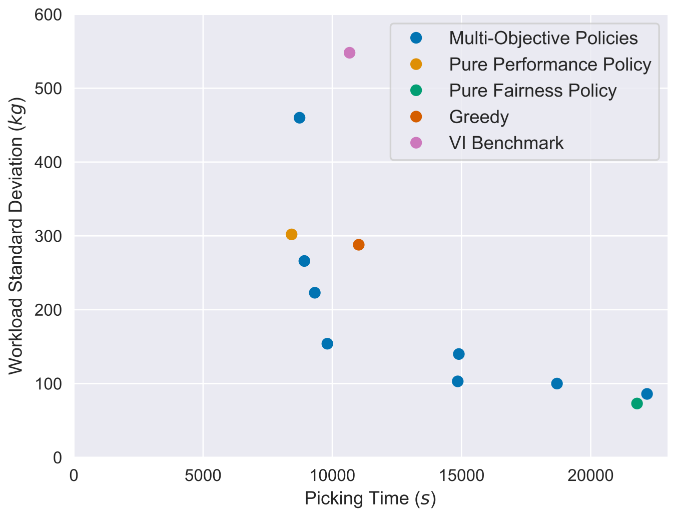


Figure 7.2: Performance evaluation of the non-dominated set of multi-objective policies learned on warehouse type M.

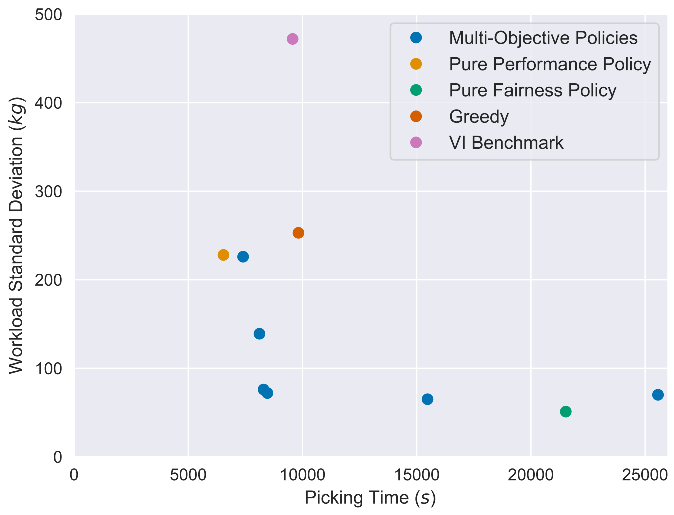


Figure 7.3: Performance evaluation of the non-dominated set of multi-objective policies learned on warehouse type L.

by 78.6%. Compared to the VI Benchmark, which achieved an average time of 10619 and workload standard deviation of 442, several policies achieved both better picking times and fairer workload distributions. For example, policy S3 managed a 9.2% pick time improvement while also having an 85.1% better workload spread. Overall, the front pushes the boundaries of the pure performance and fairness policies, indicating that better trade-offs are hard to achieve. For this warehouse type, the fairest multi-objective policies even achieved better fairness than the single-objective policy on fairness. This may be due to several reasons. The fairness policy might be unstable, leading to problems to capture the final improvements. Alternatively, the multi-objective structure might help to capture some extra policy aspects within these small warehouses. As this was not the case for the larger warehouses, it is not a finding that can be generalized. Besides, the main focus of our study is not to create fairness policies while ignoring efficiency, so we did not try to optimize this pure fairness policy further.

For warehouse type M, the non-dominated policies form a smoother front, as illustrated in Figure 7.2. This indicates that no policy was found that pushes both the picking time and workload distribution values to the lowest possible values. Instead, more of a trade-off is shown. Again, multiple policies (M5, M6, and M7) showed excellent efficiency and better workload fairness compared to the greedy and VI benchmark solutions.

The policies for type L form a front that is more similar to the front for type S, as shown in Figure 7.3. Multiple policies provide a good trade-off between fairness and efficiency depending on the decision-maker's preferences. Again, the shape of the results plot indicates that we can improve one objective value by a large margin without deteriorating the performance of the other objective. For example, policies L3 and L4 achieved 11.6% and 13.3% efficiency improvements with 84.7% and 83.9% lower workload standard deviations than the VI benchmark.

All in all, our multi-objective DRL approach found non-dominated sets of policies that outline the trade-offs between efficiency and fairness using a well-shaped front. Efficiency and fairness can be balanced to a good extent and several policies for each warehouse type performed better than the greedy approach and VI benchmark on both efficiency and workload fairness. These results can provide decision-makers with several potential policies to use based on their preferences.

7.2.2 Picker/AMR Transferability

Having established that multi-objective DRL can provide a good set of non-dominated policies to outline the possible trade-offs between fairness and efficiency, we will discuss the transferability of these multi-objective policies to scenarios with different amounts of pickers and AMRs in the system. Table 7.8 and Figures 7.4-7.9 outline the outcomes of the transferability analysis.

For warehouse type S, we find a broadly similar shape of the fronts formed by the multi-objective policies for each of the picker/AMR ratios, shown in Figures 7.4-7.6. One thing that stands out is that for larger numbers of pickers and AMRs, the multi-objective policies achieved better transferability than the pure fairness policy. Whereas the workload standard deviation of the pure fairness policy increased from roughly 50 kg for 7 pickers and 15 AMRs to over 100 kg for 15 pickers and 35 AMRs, the multi-objective policies all maintained approximately the same levels of fairness. Compared to the pure performance policy, the multi-objective policies transferred in a similar way to the different picker/AMR combinations. For example, policy S3 achieved picking times with workload standard deviation values of 13402 with 65, 8654 with 66, and 6569 with 70 for 7 pickers and 15 AMRs, 10 pickers and 30 AMRs, and 15 and 35 AMRs, respectively. These are improvements of, in order, 13.4%, 7.2%, and 9.2% in efficiency while also reducing the workload standard deviations by 89.0%, 83.7%, and 81.5%. In addition, the relative difference between the multi-objective and pure efficiency policies remained roughly equivalent. However, for the higher picker and AMR numbers, the performance loss is slightly larger. This is a sensible

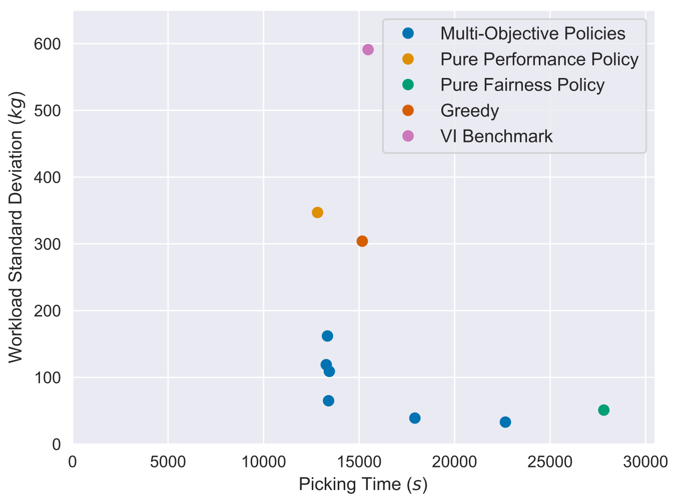


Figure 7.4: Performance evaluation of the non-dominated set of multi-objective policies learned on warehouse type S when evaluated with 7 pickers and 15 AMRs.

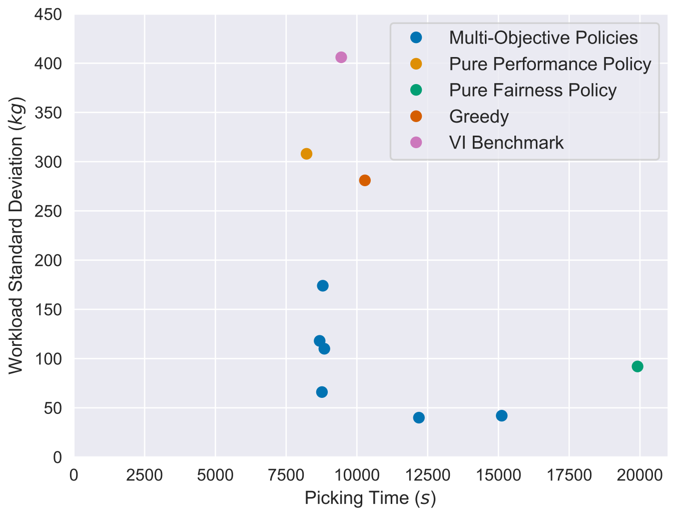


Figure 7.5: Performance evaluation of the non-dominated set of multi-objective policies learned on warehouse type S when evaluated with 10 pickers and 30 AMRs.

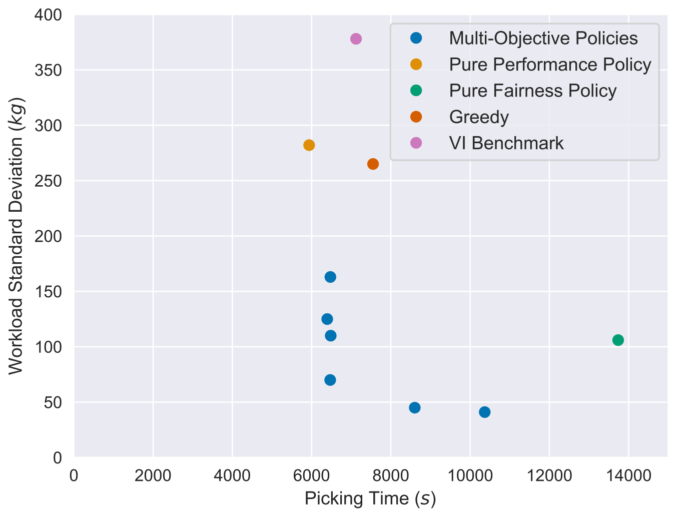


Figure 7.6: Performance evaluation of the non-dominated set of multi-objective policies learned on warehouse type S when evaluated with 15 pickers and 35 AMRs.

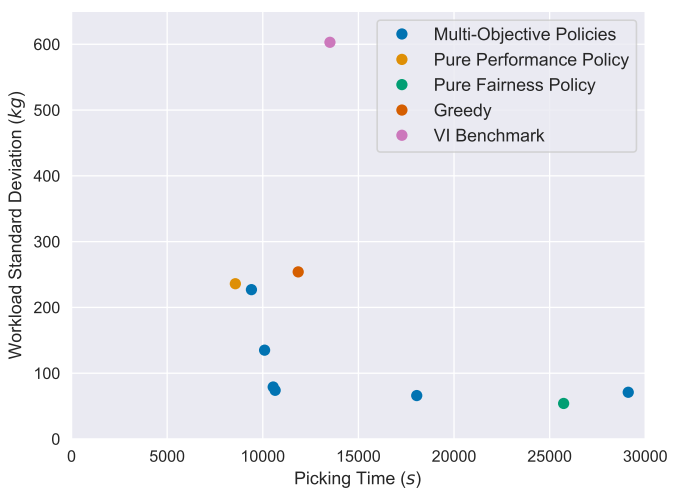


Figure 7.7: Performance evaluation of the non-dominated set of multi-objective policies learned on warehouse type L when evaluated with 25 pickers and 60 AMRs.

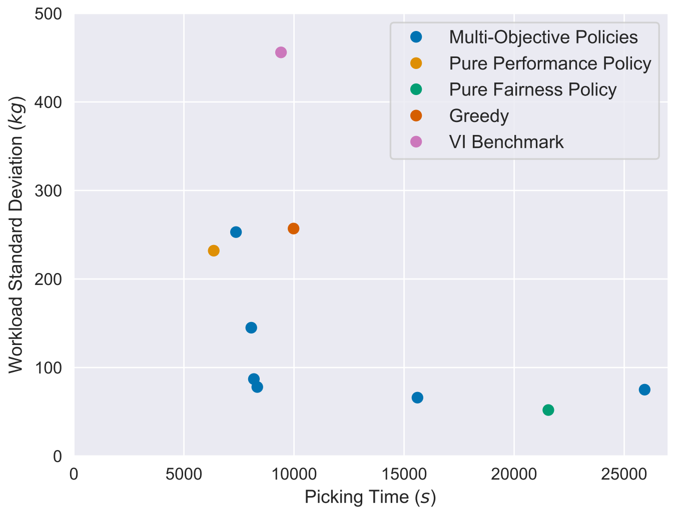


Figure 7.8: Performance evaluation of the non-dominated set of multi-objective policies learned on warehouse type L when evaluated with 30 pickers and 100 AMRs.

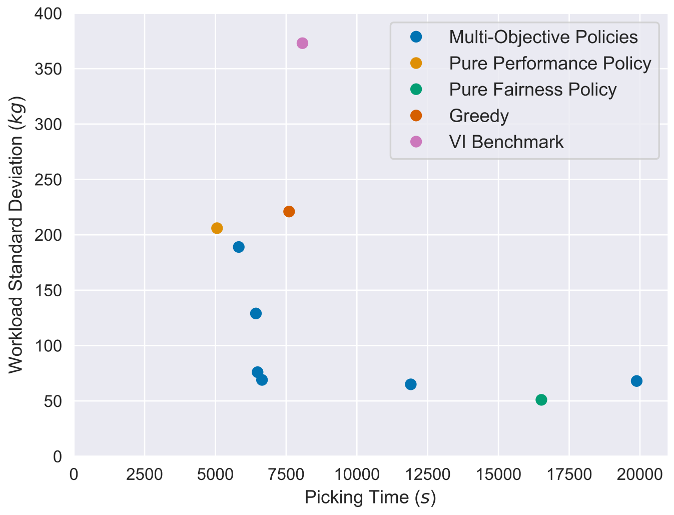


Figure 7.9: Performance evaluation of the non-dominated set of multi-objective policies learned on warehouse type L when evaluated with 40 pickers and 110 AMRs.

Table 7.8: Performance of multi-objective DRL policies trained on warehouse type S and L, given varying combinations of the number of pickers and AMRs within their respective warehouse sizes. PT is the picking time in seconds and WF is the standard deviation of the workloads in kilograms. The ± indicates the 95%-confidence interval.

(a) Warehouse type S.

	7 Pickers/15 AMRs		10 Pickers/30 AMRs		15 Pickers/35 AMRs	
Policy	PT	WF	PT	WF	PT	WF
S1	22659 ± 213	33 ± 4	15117 ± 133	42 ± 4	10369 ± 96	41 ± 3
S2	17921 ± 134	39 ± 4	12191 ± 85	40 ± 3	8603 ± 56	45 ± 3
S3	13402 ± 87	65 ± 4	8765 ± 64	66 ± 5	6469 ± 52	70 ± 4
S4	13443 ± 87	109 ± 8	8850 ± 66	110 ± 7	6482 ± 48	110 ± 6
S5	13281 ± 107	119 ± 10	8684 ± 70	118 ± 8	6394 ± 65	125 ± 6
S6	13345 ± 113	162 ± 12	8795 ± 95	174 ± 13	6474 ± 72	163 ± 11
Pure Performance	12825 ± 83	347 ± 23	8221 ± 54	308 ± 20	5938 ± 35	282 ± 15
Pure Fairness	27812 ± 115	51 ± 5	19916 ± 104	92 ± 12	13736 ± 73	106 ± 12
Greedy	15166 ± 74	304 ± 21	10283 ± 60	281 ± 15	7550 ± 44	265 ± 11
VI Benchmark	15472 ± 87	591 ± 40	9447 ± 52	406 ± 22	7121 ± 38	378 ± 15

(b) Warehouse type L.

	25 Pickers/60 AMRs		30 Pickers/100 AMRs		40 Pickers/110 AMRs	
Policy	PT	WF	PT	WF	PT	WF
L1	29109 ± 82	71 ± 6	25928 ± 93	75 ± 7	19885 ± 75	68 ± 4
L2	18050 ± 62	66 ± 3	15608 ± 53	66 ± 3	11904 ± 48	65 ± 2
L3	10647 ± 78	74 ± 6	8332 ± 64	78 ± 6	6647 ± 63	69 ± 5
L4	10545 ± 81	79 ± 4	8177 ± 76	87 ± 5	6491 ± 68	76 ± 5
L5	10095 ± 91	135 ± 6	8059 ± 73	145 ± 6	6432 ± 62	129 ± 5
L6	9407 ± 42	227 ± 10	7365 ± 61	253 ± 11	5826 ± 56	189 ± 7
Pure Performance	8566 ± 34	236 ± 7	6354 ± 65	232 ± 7	5059 ± 18	206 ± 5
Pure Fairness	25731 ± 71	54 ± 5	21554 ± 72	52 ± 3	16518 ± 64	51 ± 3
Greedy	11852 ± 31	254 ± 8	9980 ± 44	257 ± 7	7605 ± 27	221 ± 6
VI Benchmark	13512 ± 65	603 ± 15	9410 ± 62	456 ± 13	8076 ± 45	373 ± 9

pattern since denser systems enable more close picks with short travel distances. Therefore, if a picker must be relocated to a location further away to balance the workload, the potential loss of pick efficiency is more considerable as more nearby picks are missed. Overall, on warehouse size S, the multi-objective policies transferred well to the different picker and AMR numbers.

For warehouse type L, the multi-objective policies also performed well in the different evaluation settings. Namely, in all scenarios, the multi-objective policy front reached similar levels compared to the pure efficiency and fairness policies, as shown in Figures 7.7-7.9. The relative comparison between the policies looks like the front in Figure 7.3. In this case, equivalent to type S, the fairness levels stayed consistent for the different picker/AMR combinations. Oppositely to warehouse S, the pure fairness policy also maintained its fairness level with larger numbers of entities. In accordance with the previous results, for each combination of pickers and AMRs, several policies achieved better efficiency and fairness than the VI benchmark and greedy baseline. One of these policies is policy L4. For the picker/AMR ratios of 25/60, 30/100, and 40/110, this policy achieved order completion times and workload standard deviations of 10545 and 79, 8177 and 87, and 6491 and 76, respectively. These results are 22.0%, 13.1%, and

19.6% better in terms of picking time and 86.9%, 80.9%, and 79.6% better in terms of workload distribution than the VI benchmark. Similar to the type S policies, the relative cost of fairness in terms of performance decay becomes slightly larger with more entities in the system. Overall, though, the picker/AMR transferability of the multi-objective policies is also adequate for large warehouses.

7.2.3 Warehouse Size Transferability

Having established the performance of the multi-objective policies and their transferability to scenarios with different numbers of pickers and AMRs, we outline the transferability to different warehouse sizes. Table 7.9 and Figures 7.10-7.13 show the warehouse transferability results.

From Figure 7.10 and Table 7.9a, we can see that the multi-objective policies trained on warehouse types M and L were outperformed by the non-dominated policy set that was trained on warehouse size S when evaluated on this smaller warehouse type. There were a few type M policies that still improved both picking time and workload fairness over the benchmarks. In contrast, the type L policies did not manage to do so. Hence, the multi-objective policies of type L do not downscale as well as the pure performance policy did.

For warehouse type M, the type S policies transferred remarkably well, as shown in Figure 7.11 and Table 7.9b. Namely, we find that all type M policies were dominated by the type S policies while evaluating for type M. Using the type S policies, better combinations of fairness and efficiency were achieved than using the type M policies. For example, policy S3 achieved an average completion time of 8578 seconds and a workload standard deviation of 69 kg. These results are 19.6% and 87.4% better than the VI benchmark. In comparison, the best picking time of the type M policies was 8733 seconds, and the best workload standard deviation was 86 kg. The type L policies transferred reasonably to warehouse size M. We see in Figure 7.11 that the fronts pass through each other, with more type M policies having low picking times. All three fronts have several policies that improved upon the greedy method and VI benchmark for both efficiency and workload fairness.

In the larger warehouses of type L, the policy sets that were trained on the three different warehouse types formed similar result fronts, as Figure 7.12 and Table 7.9c show. All three policy sets showed a relatively sharp trade-off angle in the plot, indicating that one objective can be improved a lot without sacrificing much on the other. All sets contained policies that outperformed the benchmarks. The policies of warehouse S achieved slightly better efficiency scores, while the type M and L policies reached marginally better fairness values, but in general, the results were similar. Thus, these findings show that the multi-objective policies trained on warehouse types S and M can scale well to the larger warehouses instances of type L.

The evaluation on the XL warehouses showed a slightly different pattern, which can be seen in Figure 7.13 and Table 7.9d. Namely, what stands out is that the policies that focused more on fairness deteriorated in terms of fairness compared to the more efficient policies, especially for policy set L. Namely, policies L1 and L2 achieved a higher workload standard deviation while having much worse picking times than policies L3 and L4. In contrast, the fairness scores were similar or slightly better for the smaller sizes. Thus, policies with a significant focus on fairness may scale less well to larger warehouses in some cases. However, in practice, these policies will not often be selected as they achieved just a marginal fairness improvement while having much worse performance. On the other hand, the policies with better efficiency scaled relatively well to the largest warehouse sizes, with policy L achieving the best trade-offs. Namely, all policy sets could achieve similar efficiency, but the type L policy set managed to reach better workload distributions for these efficiency levels. For example, policy L3 scored an average picking time of 10357 with a workload standard deviation of 45 kg, constituting improvements of 23.6% and 91.9% over the VI benchmark scores of 13570 seconds and 558 kg, respectively. In general, the

Table 7.9: Performance of multi-objective DRL policies when evaluated on various warehouse sizes. PT is the picking time in seconds and the workload fairness WF is the standard deviation of the workloads in kg. The ± indicates the 95%-confidence intervals. S, M, and L in the columns indicate the training warehouse types of the policies.

(a) Evaluation results on warehouse type S.

	S		M		L	
Policy nr.	PT	WF	PT	WF	PT	WF
1	15555 ± 125	41 ± 4	20876 ± 129	95 ± 19	21182 ± 107	96 ± 14
2	12431 ± 86	43 ± 4	18938 ± 146	98 ± 10	19888 ± 95	82 ± 86
3	9164 ± 60	66 ± 4	14666 ± 96	74 ± 5	13516 ± 140	112 ± 12
4	9188 ± 55	114 ± 8	14879 ± 134	106 ± 11	12163 ± 144	81 ± 89
5	9074 ± 60	118 ± 7	11193 ± 147	155 ± 11	11621 ± 147	200 ± 15
6	9149 ± 68	167 ± 9	10302 ± 168	226 ± 15	10303 ± 113	355 ± 28
7	-	-	9577 ± 53	206 ± 18	-	-
8	-	-	9464 ± 57	441 ± 51	-	-

(b) Evaluation results on warehouse type M.

	S		M		L	
Policy nr.	PT	WF	PT	WF	PT	WF
1	15404 ± 100	51 ± 4	22180 ± 65	86 ± 10	22770 ± 102	114 ± 10
2	13267 ± 63	54 ± 5	18695 ± 174	100 ± 10	19117 ± 77	75 ± 3
3	8578 ± 69	69 ± 4	14854 ± 74	103 ± 6	12596 ± 177	154 ± 9
4	8646 ± 49	114 ± 5	14897 ± 153	140 ± 9	10424 ± 96	102 ± 9
5	8405 ± 50	122 ± 6	9809 ± 169	154 ± 8	10956 ± 185	201 ± 10
6	8485 ± 63	182 ± 9	9323 ± 136	223 ± 11	8960 ± 71	335 ± 22
7	-	-	8919 ± 51	266 ± 19	-	-
8	-	-	8733 ± 52	460 ± 32	-	-

(c) Evaluation results on warehouse type L.

	S		M		L	
Policy nr.	PT	WF	PT	WF	PT	WF
1	15913 ± 90	68 ± 7	26728 ± 159	60 ± 5	25562 ± 92	70 ± 7
2	15146 ± 67	68 ± 7	21520 ± 97	72 ± 6	15474 ± 62	65 ± 3
3	7302 ± 62	85 ± 6	14645 ± 70	128 ± 7	8463 ± 32	72 ± 5
4	7340 ± 53	131 ± 6	12939 ± 81	101 ± 5	8296 ± 78	76 ± 4
5	7249 ± 137	137 ± 6	7678 ± 71	92 ± 6	8116 ± 62	139 ± 6
6	7087 ± 64	199 ± 8	7307 ± 80	186 ± 8	7400 ± 220	226 ± 9
7	-	-	7442 ± 42	220 ± 12	-	-
8	-	-	7343 ± 43	351 ± 18	-	-

(d) Evaluation results on warehouse type XL.

	S		M		L	
Policy nr.	PT	WF	PT	WF	PT	WF
1	25380 ± 81	115 ± 9	46473 ± 337	94 ± 5	40897 ± 125	92 ± 5
2	25039 ± 72	100 ± 10	37004 ± 231	116 ± 5	21019 ± 67	72 ± 2
3	10013 ± 108	130 ± 6	23413 ± 65	160 ± 5	10357 ± 94	45 ± 4
4	9964 ± 87	183 ± 8	22389 ± 267	111 ± 4	10229 ± 112	66 ± 4
5	10208 ± 83	204 ± 8	10730 ± 151	101 ± 5	10411 ± 91	123 ± 4
6	9528 ± 42	229 ± 6	9524 ± 96	230 ± 7	9653 ± 30	211 ± 7
7	-	-	9932 ± 92	312 ± 3	-	-
8	-	-	10173 ± 113	444 ± 17	-	-

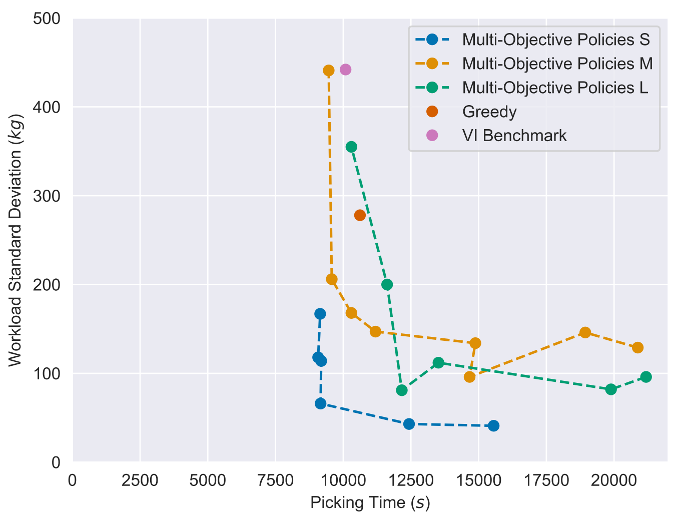


Figure 7.10: Performance of multi-objective DRL policies trained on different warehouse sizes when evaluated on warehouse size S.

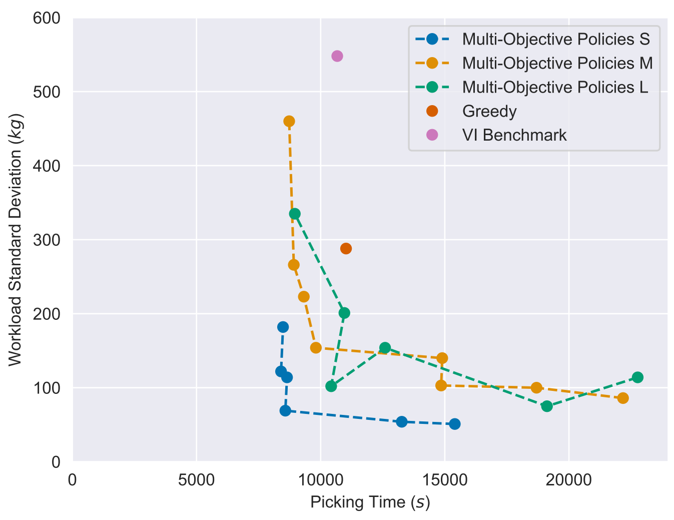


Figure 7.11: Performance of multi-objective DRL policies trained on different warehouse sizes when evaluated on warehouse size M.

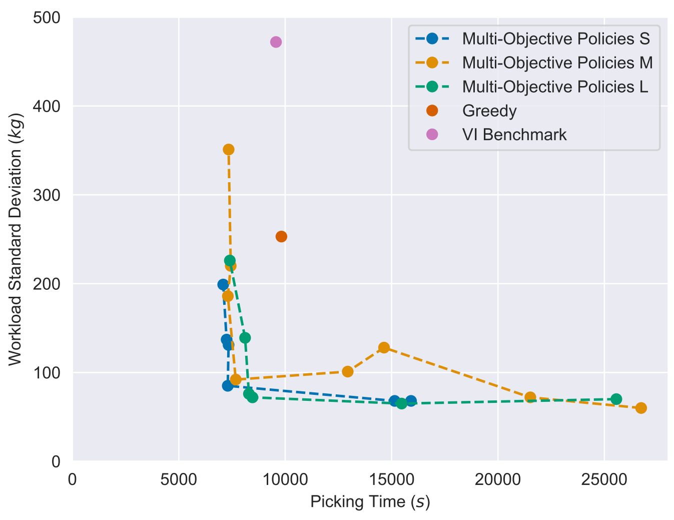


Figure 7.12: Performance of multi-objective DRL policies trained on different warehouse sizes when evaluated on warehouse size L.

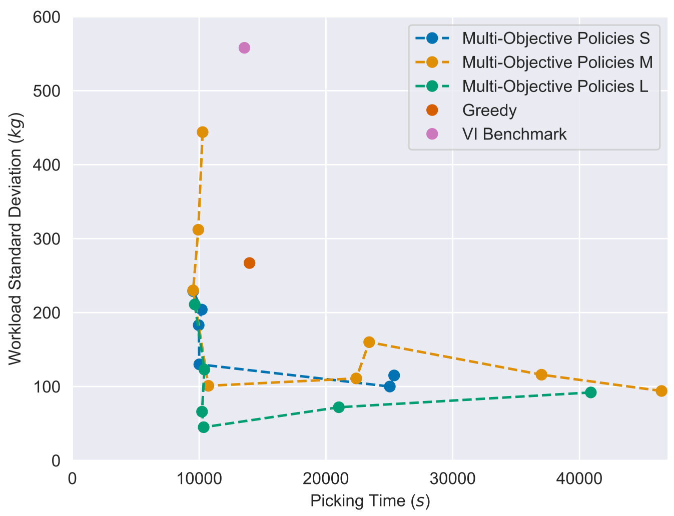


Figure 7.13: Performance of multi-objective DRL policies trained on different warehouse sizes when evaluated on warehouse size XL.

multi-objective policies scaled well to the XL warehouse size, with all policy sets having multiple policies that improved upon both the VI benchmark and greedy baseline.

Summarizing, the multi-objective policies transferred to a satisfactory degree to different warehouse sizes. For the smaller warehouses, the policy set of warehouse S outperformed the policies trained for larger warehouses. Oppositely, toward the large warehouses, the different policy sets performed similarly, and all sets contained policies that performed better than both the greedy baseline and VI benchmark on both objectives.

7.2.4 Architecture Comparison

From the previous results, we established that our proposed multi-objective DRL method could achieve policies that are both efficient and fair with regard to the workload distribution over pickers. In this part, we will compare our used network architecture with other network architectures to show its value.

Performance Comparison

Table 7.10 shows the performance comparison of our aisle-embedding architecture with feature separation (AISLE-EMB-SEP) compared to the aisle-embedding structure without feature separation (AISLE-EMB), invariant feed-forward with feature separation (INV-FF-SEP), and invariant feed-forward network without feature separation (INV-FF). The first thing that stands out is the performance difference between the aisle-embedding architectures and the invariant feed-forward architectures. On 5 of the 6 settings, the aisle-embedding instances achieved better rewards than the invariant feed-forward policies by a clear margin. Thus, whereas with single-objective optimization, the differences between aisle-embedding and invariant feed-forward actors were small, the differences are more prominent when both fairness and performance must be optimized. A possible explanation is that the node features can capture less regional information regarding fairness. Hence, the aisle-embedding architecture has more possibilities to aid in extracting relevant regional information from the graph.

Between the aisle-embedding networks with and without feature separation, we can also observe a difference. The results show that the aisle-embedding without feature separation reached slightly better rewards than the actor with feature separation in three instances. However, the improvements were only marginal. Namely, for the weight vector $(0.1, 0.9)$, the final rewards of the two structures were very close and within each other's 95%-confidence interval, indicating that we cannot conclude a statistically significant difference. In addition, the reward difference

Table 7.10: Comparison of the performance of policies with different network architectures, trained using a weighted-sum reward between performance and fairness for various warehouse sizes and weight combinations for performance (w_{perf}) and fairness (w_{fair}). The table shows the obtained reward, the picking time to complete a full episode in seconds (PT), and the standard deviation of the picker workloads in kg (WF). The ± indicates the 95%-confidence interval. The bold markings indicate the policies with the best rewards per scenario

			AISLE-EMB-SEP			AISLE-EMB			INV-FF-SEP			INV-FF		
Warehouse	w_{perf}	w_{fair}	Reward	PT	WF	Reward	PT	WF	Reward	PT	WF	Reward	PT	WF
S	0.5	0.5	-264 ± 7	17017 ± 180	100 ± 11	**-231 ± 6**	**14693 ± 209**	**91 ± 11**	-372 ± 5	24400 ± 83	129 ± 9	-543 ± 12	22410 ± 112	506 ± 26
S	0.9	0.1	**-236 ± 4**	**10210 ± 182**	**72 ± 10**	-379 ± 3	16366 ± 148	110 ± 9	-288 ± 2	12055 ± 74	166 ± 10	-536 ± 3	21507 ± 118	502 ± 25
S	0.1	0.9	-149 ± 11	21265 ± 113	102 ± 11	**-138 ± 9**	**22106 ± 102**	**89 ± 9**	-187 ± 13	21808 ± 126	142 ± 14	-179 ± 11	21937 ± 88	133 ± 12
M	0.5	0.5	-339 ± 5	20340 ± 141	166 ± 10	-384 ± 5	23555 ± 100	175 ± 10	**-255 ± 7**	**17023 ± 90**	**230 ± 8**	-381 ± 6	20989 ± 110	232 ± 11
M	0.9	0.1	**-361 ± 3**	**15305 ± 139**	**163 ± 8**	-463 ± 2	20154 ± 80	100 ± 6	-463 ± 3	19324 ± 100	282 ± 12	-358 ± 3	15102 ± 74	186 ± 9
M	0.1	0.9	-202 ± 9	25250 ± 84	151 ± 9	**-197 ± 7**	**23302 ± 86**	**151 ± 8**	-253 ± 7	27338 ± 95	200 ± 8	-256 ± 7	16946 ± 93	232 ± 6

was relatively small for weight vector $(0.5, 0.5)$ on warehouse S. Oppositely, in the cases in which the actor with feature separation performed better, the difference in rewards was much larger, being -236 versus -379 and -361 versus -463. Moreover, we observe that in these instances, with these weight vectors, the best overall policies for the warehouse types were found. Namely, both policies dominated all other policies for all weight vectors on both picking time and workload fairness. Thus, this weight vector region in which the aisle-embedding with feature separation outperforms the other architectures is also the region where the best policies are achievable. Hence, the aisle-embedding structure with feature separation is the best network architecture for the multi-objective learning task.

What is also noteworthy in these results is that it is hard to judge which weight vectors lead to which trade-offs between efficiency and fairness. For example, we saw that the policies for weights $w_{perf} = 0.9$ and $w_{fair} = 0.1$ scored very well on both efficiency and fairness and even achieved better fairness than the policies with $w_{perf} = 0.1$ and $w_{fair} = 0.9$. In addition, the outcome of different weight settings varies between warehouse sizes. For example, for the aisle-embedding without feature separation, the best efficiency and fairness scores in warehouse type S were achieved for weight vector $(0.5, 0.5)$, whereas for type M the policy for weight vector $(0.9, 0.1)$ outperformed the other policies. These findings highlight the value of using PGMORL to find the weights that form a high-quality non-dominated set of policies. Otherwise, trying to hand-tune the weights for each problem instance would cost a vast amount of computational resources and effort to find clear trade-off fronts.

Inference Time Comparison

Table 7.11 shows the inference times of the different actor policies. We find that the inference times were similar to those of the single-objective policies. Our aisle-embedding architecture with feature separation was the slowest and the invariant feed-forward the quickest. However, all inference times stayed within a few milliseconds for small and large warehouses. Thus, inference time is no issue for the practical deployment of the multi-objective policies.

Table 7.11: Inference time of multi-objective DRL policies with different network architectures. The numbers represent the average inference time per action in milliseconds over 1000 collected actions, with ± indicating the 95%-confidence intervals. The bold markings indicate the best inference times per warehouse size.

Warehouse	AISLE-EMB-SEP	AISLE-EMB	INV-FF-SEP	INV-FF
S	2.92 ± 0.03	2.13 ± 0.03	1.68 ± 0.03	$\mathbf{1.51 \pm 0.03}$
M	3.10 ± 0.04	2.22 ± 0.03	1.73 ± 0.03	$\mathbf{1.57 \pm 0.03}$
L	3.32 ± 0.06	2.39 ± 0.03	1.89 ± 0.04	$\mathbf{1.63 \pm 0.03}$
XL	3.50 ± 0.04	2.60 ± 0.04	1.97 ± 0.03	$\mathbf{1.79 \pm 0.03}$

7.3 Synthesis

In this chapter, we discussed the performance of our single-objective and multi-objective DRL approaches to optimizing the picker allocation task in collaborative picking. In Section 7.1, we showed that DRL could learn policies that outperform the benchmark methods by a clear margin. For large warehouses, we found consistent efficiency improvements of roughly 30% to 40% over the VI benchmark for a variety of warehouse instances with different numbers of human pickers and AMRs in the system. We also showed that DRL could reach near-optimal solutions in small deterministic instances compared to their MILP solutions. The DRL policies we trained for different warehouse sizes maintained good performance when transferred to warehouses of different scales. For instance, a DRL policy trained on warehouses with just 200 pick locations, 10 pickers, and 25 AMRs only performed roughly 5% worse on warehouses with 1250 pick locations, 30 pickers, and 90 AMRs than a policy trained explicitly for that warehouse type.

In Section 7.2, we outlined the results of the multi-objective experiments. These experiments highlighted that we could find a good set of non-dominated policies for various warehouses that show the trade-off between efficiency and fairness. Decision-makers can use these policy sets to make informed decisions on which efficiency and fairness levels they want to achieve. In general, the standard deviation of these workloads can be greatly decreased while requiring a limited sacrifice in efficiency. For each warehouse size, multiple policies performed better than the greedy heuristic and VI benchmark on both efficiency and workload fairness. On the largest warehouse size, our approach achieved policies that could improve efficiency by over 20% while simultaneously decreasing the standard deviation of the picker workload by roughly 90% over the VI benchmark. In general, the policies transferred well to varying numbers of pickers and AMRs in the system. In addition, the transferability to larger and smaller warehouses was satisfactory. Overall, the multi-objective DRL approach achieved valuable results by offering multiple trade-off options between efficiency and fairness and generating policies that perform well in both aspects.

Chapter 8

Policy Analysis

The results in the previous chapter showed that DRL policies can achieve excellent efficiency and workload fairness. They significantly improved over the VI benchmark and greedy baseline. On the downside, these DRL policies are so-called “black boxes” that do not have transparent rules that can directly be extracted. To gain insights into the logic and behavior of these policies, we performed two analyses. First, we performed a decision tree analysis to extract rules from the behavior of the policies. Second, we created videos of the collaborative order picking simulation for different policies to visually inspect what happens in the warehouses.

8.1 Decision Tree Analysis

To extract meaning from DRL policies, several methods have been developed, as highlighted in a review by Vouros (2023). These methods detect relevant agent trajectories or distinguish when essential actions are taken. However, these methods are generally not applicable to our policy. Namely, a single “trajectory” of the DRL policy does not highlight much as it allocates many pickers in turns. Besides, extracting valuable moments for each action is not possible. Namely, we do not have clear actions such as “left” or “right”. Instead, our action space is highly variable, with nodes that are often masked and can be permuted.

Therefore, we applied a more traditional method, decision tree analysis. However, our actor outputs action probabilities for many nodes, each with many node features. It is not feasible to input a full graph into a decision tree model. Instead, we focussed on the specific nodes. But we cannot apply a decision tree to a specific node to find the action probability. Namely, the action probability is not only dependent on the quality of this specific node but also on the quality of all other nodes. To alleviate this, in our analysis, we trained decision trees that take the node features as input and estimate the node value outputted by the actor network before passing all node values into the softmax functions. These node values indicate the quality of the node, regardless of the quality of other nodes.

We utilized these decision trees in various ways. First, we extracted the learned decision rules of shallow trees. We inspected the rules to understand the internal considerations that the policies use to value nodes. This analysis helps us understand the logic within the “black box” policies. Second, we used these trees to create policies and tested the performance of these policies. In these policies, we used the decision trees to create a node value for each node within the available action space, as outlined in Figure 8.1. Then, the policies selected the node with the highest estimated value. If such policies work well, this offers several advantages over the true DRL actors. Firstly, people can inspect and know the exact rules used within the policy and these rules always stay the same, which stimulates trust in the model. With the rules always being the same, one can be more confident in the robustness of the method when untested scenarios are encountered. Secondly, the trees can be used to explain why specific actions are selected.

Thirdly, decision trees are generally computationally cheap and can be easily implemented.

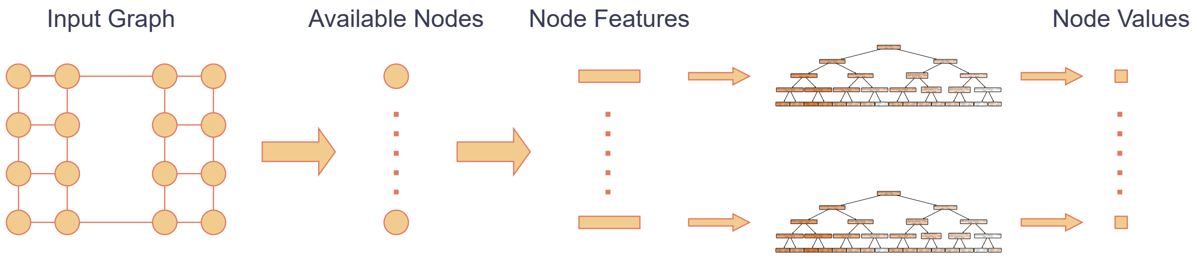


Figure 8.1: Illustration of decision tree policy.

8.1.1 Decision Tree Training

To train the decision trees, we ran the DRL policies for 100 episodes. In these episodes, we collected pairs of node features with the outputted node value calculated right before the softmax function. We only collected these pairs for nodes that were available as action. Then, we sampled 500,000 collected pairs for our training. We performed this procedure for pure efficiency, pure fairness, and multi-objective policies trained on warehouse type S. We also collected the data from runs on warehouse type S. In Appendix C, we show several data rows to illustrate the collected samples.

For each policy, we repeated the following training procedure using the Scikit-learn package (Buitinck et al., 2013). We created an 80%/20% train-test split. Then, on the training set, we performed a hyperparameter search for the maximum tree depth, the maximum number of features, and the minimum number of samples in the leaves. We did so using 5-fold cross-validation. Then, we retrained the decision trees with the optimal hyperparameters on the complete training sets and evaluated them on the independent test sets using the R^2 metric. This R^2 metric measures the percentage of variance in the data explained by the model, with 1 being the highest possible value. In addition, for each policy, we trained shallow decision trees with a maximum depth of 4 using the best other hyperparameters, which were to allow all features and have a minimum number of samples per leaf of 5. Table 8.1 shows the optimal hyperparameters and evaluation scores of the trees for each policy. The full grid search results can be found in Appendix D. We find that the best decision trees achieved high R^2 scores between 0.85 and 0.96, while the shallow decision trees, despite their small sizes, still achieved reasonably good R^2 scores. Thus, the rules from the shallow decision trees can provide reasonable explanations of the most important considerations in policy behavior. From the R^2 values, we see that the estimations of the node values for the fairness policy are worse than for the other policies, and, oppositely, the R^2 is best for the pure efficiency policy. This may be explained by the finding that the aisle-embedding structure improved considerably over the feed-forward architecture when fairness was involved compared to a small improvement for pure efficiency. Namely, the decision trees can only consider the node values and cannot take into account any other nodes, similar to the invariant feed-forward structure. Thus, they cannot capture the behavior based on the aisle-embedding well. The fairness policies and multi-objective policies may utilize the aisle-embedding structure more within their architecture, leading to worse decision tree estimations.

8.1.2 Decision Tree Interpretation

We will highlight a few shallow decision trees to indicate the insights that can be obtained. First, Figure 8.2 shows the shallow decision tree of the pure performance policy. In this tree, we find that this policy considers the distance of the controlled picker to the node very important, as it is used in the first split, and, in general, rules higher in the tree are more important. We find that nodes with a distance of less than 9.6 m, which is roughly two-thirds of an aisle in this warehouse size, generally receive a higher value, as can be seen by comparing the values after

Table 8.1: Overview of the hyperparameters of the best decision tree for each policy and the R^2 scores for the best and the shallow decision trees.

Policy	Max. Depth	Max. Nr. of Features	Min. Samples Leaf Nodes	R^2 Tree-Best	R^2 Tree-Shallow
1	25	All	25	0.85	0.56
2	25	All	25	0.87	0.69
3	50	All	50	0.84	0.70
4	50	All	25	0.87	0.71
5	50	All	25	0.88	0.68
6	25	All	10	0.92	0.69
Pure Performance	40	All	10	0.96	0.76
Pure Fairness	15	All	100	0.65	0.58

this split. However, distance is not the only consideration. Namely, we observe that other splits in the tree use features such as the AMR destination distance, the distance of other pickers, and the number of AMRs in the aisle. For example, the decision path toward the highest node value considers multiple features. First, it considers if the distance from the picker is less than 9.6 m, then it considers if the AMR destination distance is less than -5. This may seem strange, but we must remember that this feature was set to -10 if no AMR is already going to the location. The higher node values are given when the value is more than -5. So, if an AMR is currently going to or is already at the location, the value is higher. Then, in the next step, the node value gets better if the AMR that is going is closer than 6.8 m. In the last step, a check is done whether the closest other picker is within 6.8 m, with the node value being better if the closest other picker is further away. Thus, the best node values are given to nodes close to the picker with an AMR that is going there and is already nearby the location, and with other pickers being further away. These are sensible, understandable considerations that provide insights and trust into the policy logic. Similarly, the lowest node values can be found in the path toward the white box at the bottom left of the tree. This path considers nodes where no AMR is already going, the closest other picker is nearby, and the number of pickers going to the aisle is high. Again, these are sensible considerations that take into account multiple factors.

Figure 8.3 depicts the decision tree of the pure fairness policy. For this tree, similar analyses can be made. We find that the main considered features in this tree are the total workload for the picks of the AMRs that are waiting at a location, the maximum workload compared to the mean, and the workload of the picker being allocated. On the right side of the tree, where the workloads of the items to be picked for the waiting AMRs are higher, toward the bottom of the tree, value splits are done based on the current picker workloads. We observe that in these cases, when the picker workloads are lower, the node values are higher. This is sensible as one wants pickers with a low experienced workload so far to perform heavy tasks to spare the pickers with a high current workload. In general, the rules in this tree are less diverse and less subjectively sensible than those for the pure performance policy. This could be related to the lower estimation performance of the decision tree for fair policies we discussed.

We show the decision tree of multi-objective policy 2, one of the multi-objective policies steered more toward fairness, in Figure 8.4. In this tree, the picker and AMR distances are still prevalent over workload features such as the current picker workload. This is sensible since further pick locations are less optimal when any degree of efficiency is desired. The right side of the decision tree considers nodes that are further away from the picker. Here, we observe that if the picker workloads are above a certain threshold, these further nodes get a higher node value. This can indicate that the policy steers pickers to further away nodes when the current picker workload is high to get out of the current region of heavy products. However, the decision tree is not sufficiently deep to verify this logic via additional features that consider the product workloads at the nodes. This indicates one of the trade-offs of decision trees. On the one hand, shallow trees are easier to understand and can help detect the most important patterns, while on the other hand, they lack detailed information. The other shallow decision trees are available in Appendix E. These showed a similar pattern in which the distance features are essential, as they are present in the first few value splits.

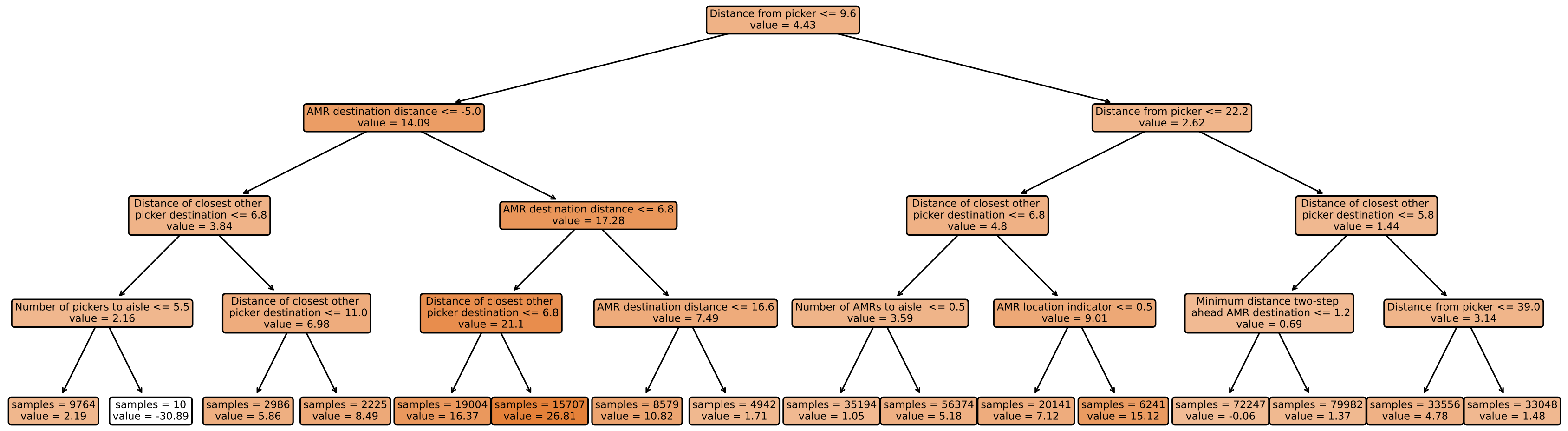


Figure 8.2: Shallow decision tree for the pure performance policy. Decision paths must be read from top to bottom, with the left side indicating that the condition is satisfied and the right side that it is not. Darker colors indicate high node values and lighter colors represent low node values.

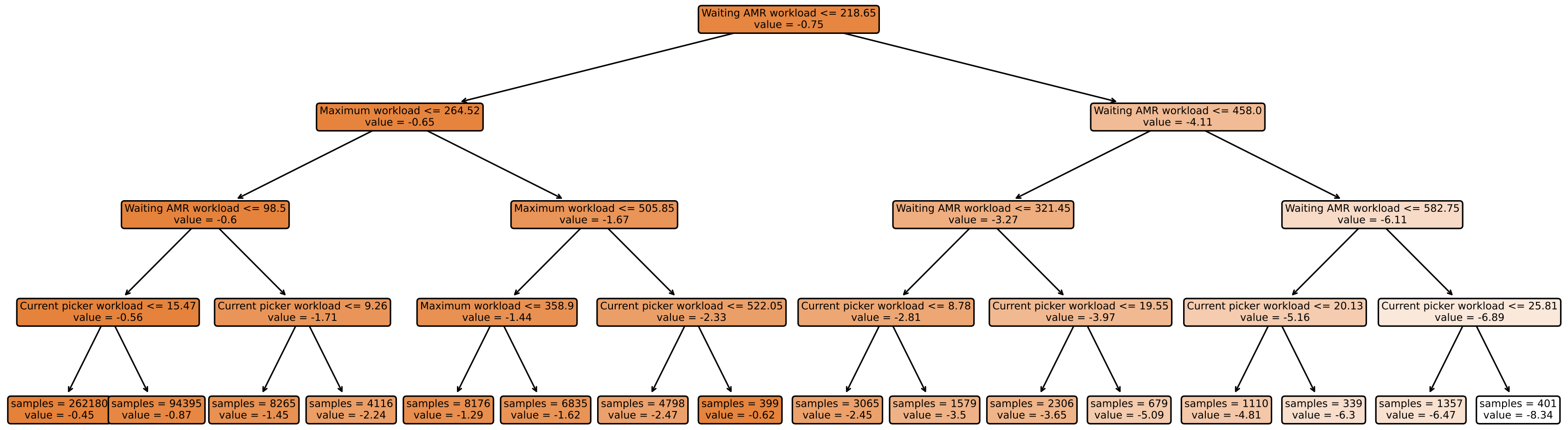


Figure 8.3: Shallow decision tree for the pure fairness policy. Decision paths must be read from top to bottom, with the left side indicating that the condition is satisfied and the right side that it is not. Darker colors indicate high node values and lighter colors represent low node values.

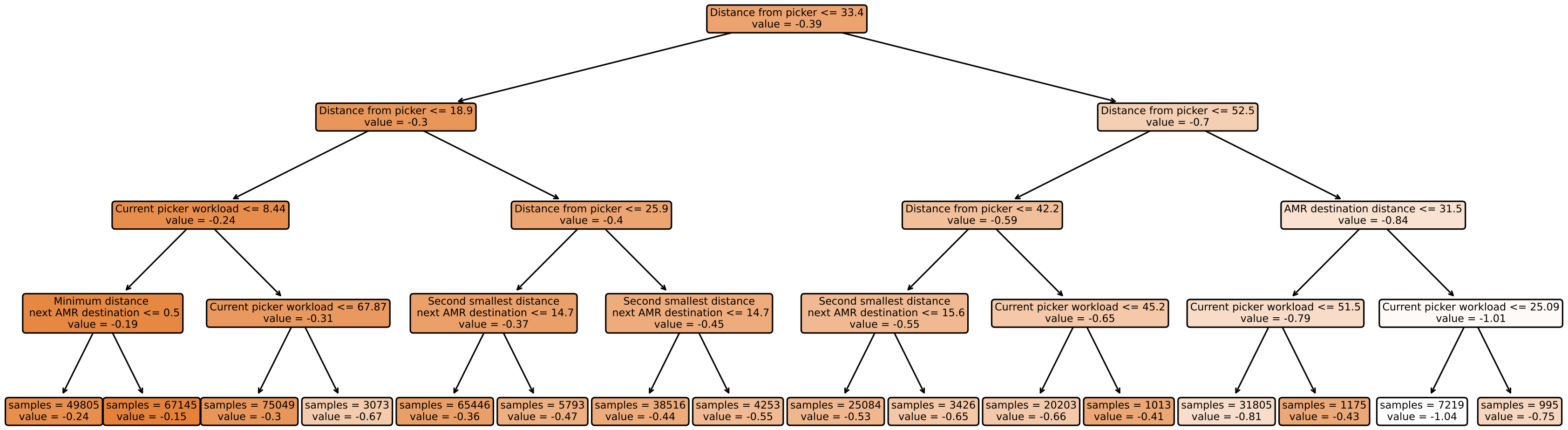


Figure 8.4: Shallow decision tree for multi-objective policy 2. Decision paths must be read from top to bottom, with the left side indicating that the condition is satisfied and the right side that it is not. Darker colors indicate high node values and lighter colors represent low node values.

8.1.3 Decision Tree Policy Analysis

Having trained the decision trees and shown how these can be used to extract the logic underlying the DRL policies, we evaluated the performance of the decision tree policies. To do so, we ran each policy for 100 episodes on warehouse type S and collected the order completion times and workload distributions. We did so for the shallow and best decision tree of each policy. We compared these performances with those of the DRL policies.

Performance Evaluation

Table 8.2 and Figure 8.5 show the results of this experiment. In the figure, two patterns stand out. First, the policies of the best trees perform very similarly to the DRL policies. Second, the shallow trees have a significant drop in performance compared to the best trees and DRL solutions. For the performance policy, we see that using the best decision tree policy only increased the average total picking times from 8586 to 8785 seconds, a mere 2.3% performance loss. Oppositely, the shallow decision tree achieved an average time of 10539 seconds, which is a more significant loss. This performance is comparable with the VI benchmark and greedy policy times of 10087 and 10619 seconds, respectively. Thus, this rough performance level seems to be the approximate limit of relatively simple rule-based methods.

Table 8.2: Comparison of performance of DRL policies and their associated best and shallow decision tree policies on warehouse type S. The picking time is the number of seconds to complete an episode and the workload fairness is the standard deviation of the picker workloads. The ± indicates the 95%-confidence interval.

	Tree-Best		Tree-Shallow		DRL	
Policy	PT	WF	PT	WF	PT	WF
1	16373 ± 139	41 ± 5	14409 ± 88	80 ± 6	15555 ± 125	41 ± 4
2	12347 ± 83	41 ± 4	12803 ± 76	60 ± 4	12431 ± 86	43 ± 4
3	9350 ± 65	81 ± 5	11378 ± 60	283 ± 17	9164 ± 60	66 ± 4
4	9505 ± 62	161 ± 10	10783 ± 51	276 ± 15	9188 ± 55	114 ± 8
5	9561 ± 77	197 ± 14	10781 ± 69	268 ± 13	9074 ± 60	118 ± 7
6	9480 ± 79	225 ± 12	10957 ± 69	320 ± 14	9149 ± 68	167 ± 9
Pure Performance	8785 ± 57	304 ± 21	10539 ± 68	284 ± 16	8586 ± 62	308 ± 17
Pure Fairness	19141 ± 110	173 ± 20	18873 ± 114	300 ± 14	19962 ± 86	61 ± 9

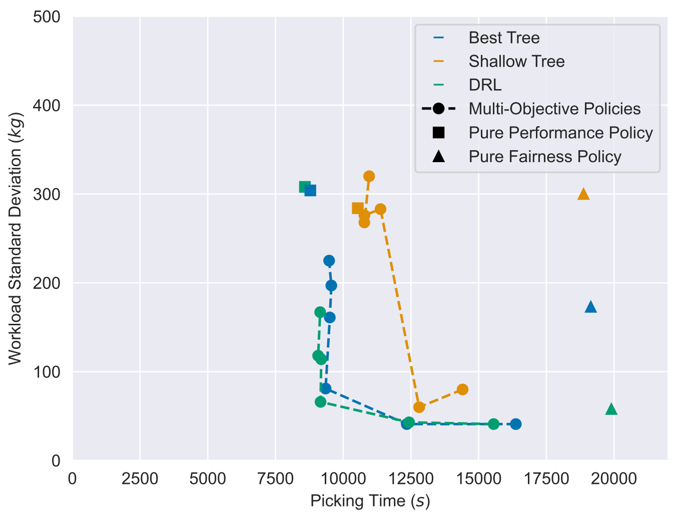


Figure 8.5: Performance evaluation of decision tree policies.

For the multi-objective policies, we also find that the efficiency decreased by just a small margin. Oppositely, for the multi-objective policies that lean toward good performance, the loss in workload fairness is more considerable. We observe that, in the figure, the efficiency values are very close, but the workload standard deviations are larger compared to the DRL policies that the trees belong to. For example, the standard deviation of policy 5 increases from 118 to 197 kg and of policy 6 from 167 to 225 kg. This more significant decrease in fairness may be caused by the trees not sufficiently capturing regional information to mimic the fairness considerations of the DRL policies. For the multi-objective shallow tree policies, we see that the policy set does not form a well-shaped trade-off front anymore. None of these policies achieved both good efficiency and good fairness.

For pure fairness policies, the decrease in performance for the decision trees is greater than for the multi-objective and pure efficiency-focused policies. We observed that the best decision tree policy nearly tripled the workload standard deviation from 61 to 173 kg compared to DRL. The shallow tree policy lost its fairness and did not achieve a better workload standard deviation than the efficiency-focused policies.

In general, the multi-objective and pure performance policies of the best decision trees performed well compared to the DRL policies. The efficiency was just slightly lower, and the fairness decrease was reasonably low. Thus, they can provide an alternative for the DRL policies. Decision-makers can decide whether the increased transparency and having a fixed rule set instead of the black box network outweigh this performance decrease.

Picker/AMR Transferability

The decision trees were trained using experiences sampled from small warehouse instances with 10 pickers and 25 AMRs. Therefore, their policies could mainly work well in those settings and not when the instance parameters change. For the decision tree policies to form a suitable alternative to the black box DRL policies, they must also transfer well to other settings. Hence, we evaluated the picker/AMR transferability. To do so, we tested the performance of the policies of the best decision trees on warehouse instances with picker/AMR numbers of 7/15, 10/30, and 15/35.

Table 8.3 and Figures 8.6-8.8 present the outcomes of these runs. The figures show that the shapes of the formed fronts by the policies are similar for all picker/AMR combinations. In addition, the differences between the tree policies and DRL policies are similar. The tree policies all showed just a small decrease in efficiency compared to the DRL policies. For each of the scenarios, we find that the decision trees lack more in terms of fairness. For the larger numbers of pickers and AMRs, these decays seem a bit more intense, as shown by the steeper deviation between the fronts. However, the differences are not drastic. Therefore, we conclude that the decision tree policies also transferred well to different crowdedness levels within the warehouses, despite being learned from behavior in warehouses with fixed numbers of pickers and AMRs.

Warehouse Size Transferability

As the decision trees were only sampled from experiences in small warehouses, one can expect the performance to decay when the decision tree policies are applied in larger warehouses. Namely, the used rules were based on certain specific values for, for example, the distances, while in larger warehouses, distances may have different scales and become larger. To validate whether this is the case or whether the decision tree policies also scale well despite this training procedure, we evaluated the policies on warehouse types M and L.

We present these results in Table 8.4 and Figures 8.9 and 8.10. Remarkably, the decision tree policies transferred well to the larger warehouse types. Namely, the general shapes of the fronts for these larger warehouses are on par with the fronts for the smaller warehouse size. What stands out is the performance of the policies on the large warehouses of size L. For this warehouse size,

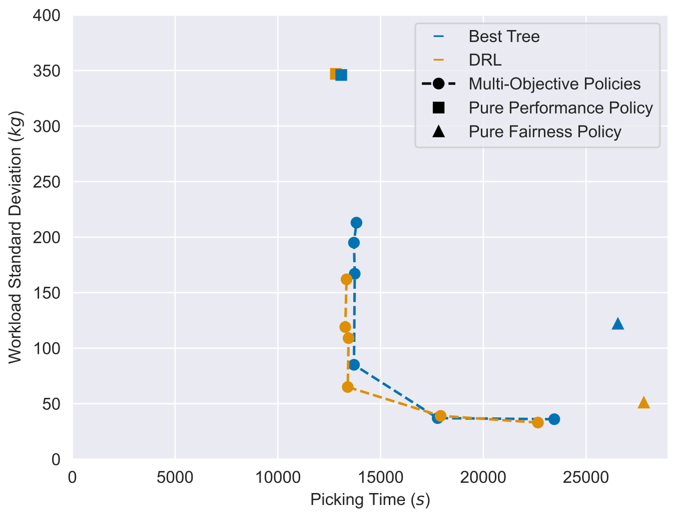


Figure 8.6: Performance evaluation of decision tree policies on warehouses with 7 pickers and 15 AMRs.

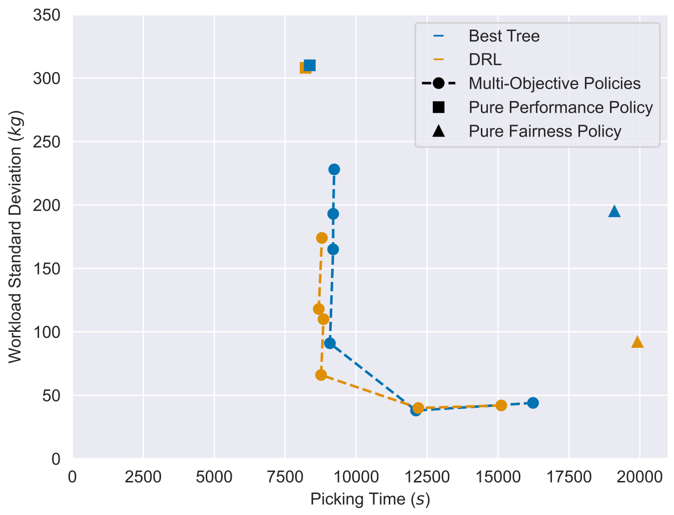


Figure 8.7: Performance evaluation of decision tree policies on warehouses with 10 pickers and 30 AMRs.

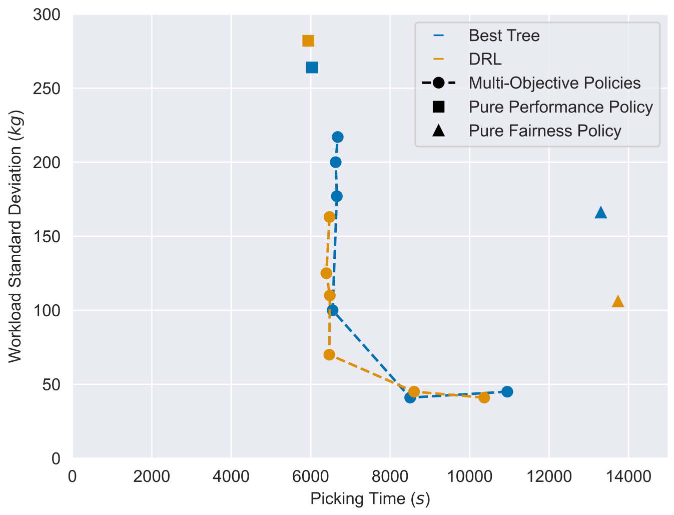


Figure 8.8: Performance evaluation of decision tree policies on warehouses with 15 pickers and 35 AMRs.

Table 8.3: Comparison of performance of DRL policies and the associated best decision tree policies on warehouse type S with varying numbers of pickers and AMRs. The picking time is the number of seconds to complete an episode and the workload fairness is the standard deviation of the picker workloads. The ± indicates the 95%-confidence interval.

(a) 7 pickers and 15 AMRs.

	Tree-Best		DRL	
Policy	PT	WF	PT	WF
1	23454 ± 219	36 ± 5	22659 ± 213	33 ± 4
2	17778 ± 122	37 ± 4	17921 ± 134	39 ± 4
3	13706 ± 87	85 ± 7	13402 ± 87	65 ± 4
4	13742 ± 85	167 ± 13	13443 ± 87	109 ± 8
5	13704 ± 93	195 ± 15	13281 ± 107	119 ± 10
6	13826 ± 83	213 ± 15	13345 ± 113	162 ± 12
Pure Performance	13087 ± 62	346 ± 24	12825 ± 83	347 ± 23
Pure Fairness	26551 ± 158	122 ± 18	27812 ± 115	51 ± 5

(b) 10 pickers and 30 AMRs.

	Tree-Best		DRL	
Policy	PT	WF	PT	WF
1	16236 ± 135	44 ± 6	15117 ± 133	42 ± 4
2	12110 ± 82	38 ± 3	12191 ± 85	40 ± 3
3	9077 ± 69	91 ± 7	8765 ± 64	66 ± 5
4	9190 ± 65	165 ± 12	8850 ± 66	110 ± 7
5	9192 ± 81	193 ± 14	8684 ± 70	118 ± 8
6	9228 ± 78	228 ± 15	8795 ± 95	174 ± 13
Pure Performance	8367 ± 66	310 ± 20	8221 ± 54	308 ± 20
Pure Fairness	19103 ± 122	195 ± 17	19916 ± 104	92 ± 12

(c) 15 pickers and 35 AMRs.

	Tree-Best		DRL	
Policy	PT	WF	PT	WF
1	10949 ± 95	45 ± 44	10369 ± 96	41 ± 3
2	8504 ± 62	41 ± 3	8603 ± 56	45 ± 3
3	6551 ± 46	100 ± 6	6469 ± 52	70 ± 4
4	6658 ± 45	177 ± 10	6482 ± 48	110 ± 6
5	6629 ± 53	200 ± 10	6394 ± 65	125 ± 6
6	6680 ± 46	217 ± 11	6474 ± 72	163 ± 11
Pure Performance	6032 ± 42	264 ± 17	5938 ± 35	282 ± 15
Pure Fairness	13305 ± 81	166 ± 15	13736 ± 73	106 ± 12

the decision tree approaches of multi-objective policies 1 and 2, which lean toward fairness, and the pure fairness policy outperformed the equivalent DRL policies on both efficiency and fairness. Thus, the fairness-focussed tree policies actually transferred better than the fairness-focussed DRL policies. It may be that the aisle-embedding strength of the network weakens slightly when the scale size of the aisle changes drastically. Then, the tree policies which used node-wise information solely might start to do relatively well as they do not depend on the aisle aggregation. Oppositely, for the more efficiency-focused multi-objective policies, we find that fairness is still clearly better for the DRL policies while achieving similar efficiency levels.

In general, the decision tree policies transferred well to changing warehouse sizes. Thus, the decision tree policies can achieve efficiency levels close to the DRL policies while reaching reasonable fairness levels. They can be applied to varying warehouse sizes with different numbers of pickers and AMRs. Hence, they offer a usable alternative to black box DRL policies. Decision makers should decide whether the advantages of using a decision tree policy, such as interpretability, simplicity, and robustness of the rules, outweigh the performance decrease. In any case, the decision trees can gather insights into the internal considerations made by the DRL policies.

Table 8.4: Comparison of performance of DRL policies and the associated best decision tree policies that were trained on warehouse type S on other warehouse types. The picking time is the number of seconds to complete an episode and the workload fairness is the standard deviation of the picker workloads. The ± indicates the 95%-confidence interval.

(a) Warehouse type M.

	Tree-Best		DRL	
Policy	PT	WF	PT	WF
1	15432 ± 50	50 ± 4	15404 ± 100	51 ± 4
2	12997 ± 65	46 ± 3	13267 ± 63	54 ± 5
3	8629 ± 54	114 ± 8	8578 ± 69	69 ± 4
4	8868 ± 54	189 ± 10	8646 ± 49	114 ± 5
5	8718 ± 67	224 ± 10	8405 ± 50	122 ± 6
6	8820 ± 61	238 ± 12	8485 ± 63	182 ± 9
Pure Performance	8015 ± 41	309 ± 13	7931 ± 42	301 ± 12
Pure Fairness	20752 ± 116	138 ± 57	21565 ± 112	112 ± 11

(b) Warehouse type L.

	Tree-Best		DRL	
Policy	PT	WF	PT	WF
1	13106 ± 82	50 ± 4	15913 ± 90	68 ± 7
2	12509 ± 59	49 ± 3	15146 ± 67	68 ± 7
3	6761 ± 44	155 ± 6	7302 ± 62	85 ± 6
4	7159 ± 195	195 ± 7	7340 ± 53	131 ± 6
5	7192 ± 51	216 ± 7	7249 ± 137	137 ± 6
6	7270 ± 44	230 ± 6	7087 ± 64	199 ± 8
Pure Performance	6986 ± 39	268 ± 18	6877 ± 31	248 ± 9
Pure Fairness	21382 ± 122	74 ± 5	22329 ± 56	80 ± 6

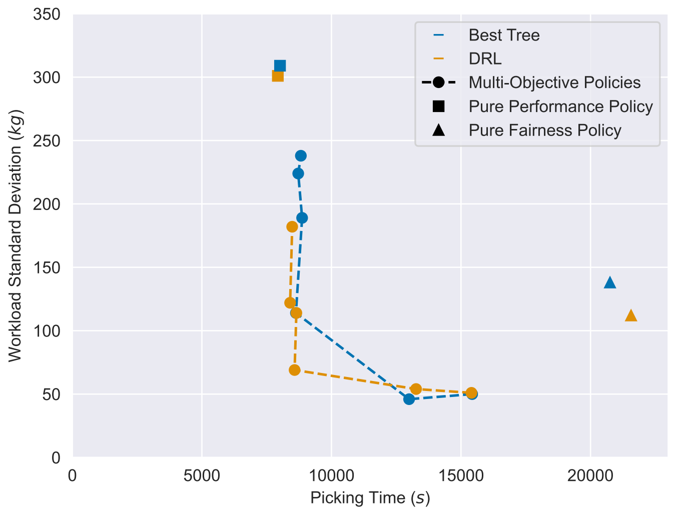


Figure 8.9: Performance of decision tree policies trained on warehouse type S when evaluated on warehouse type M.

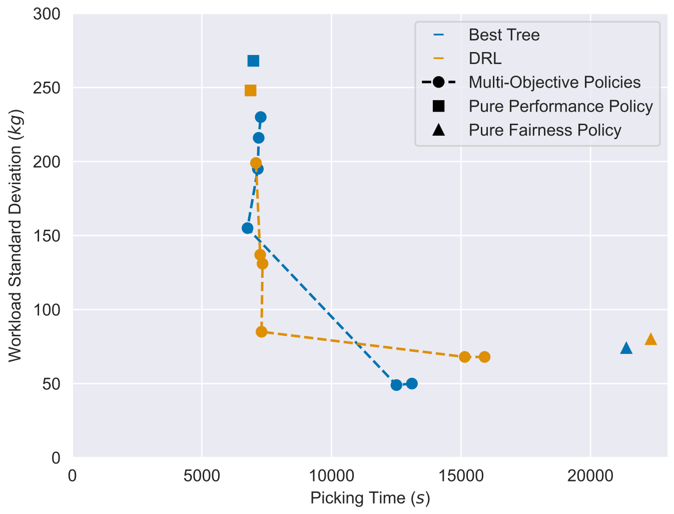


Figure 8.10: Performance of decision tree policies trained on warehouse type S when evaluated on warehouse type L.

8.2 Inspecting Policy Behavior

In addition to the decision tree analysis, we explored the behavior of the DRL policies compared to the greedy method and the VI benchmark. To do so, we ran the pure performance policy of size S, the greedy method, and the VI benchmark on the same problem instance of a warehouse of type S with 15 pickers and 30 AMRs. Then, we created videos[1] of the simulation for each of these runs to visualize the behavior of the system. We used these visualizations to inspect how the behavior in the warehouse differs and how this can be related to the performance difference.

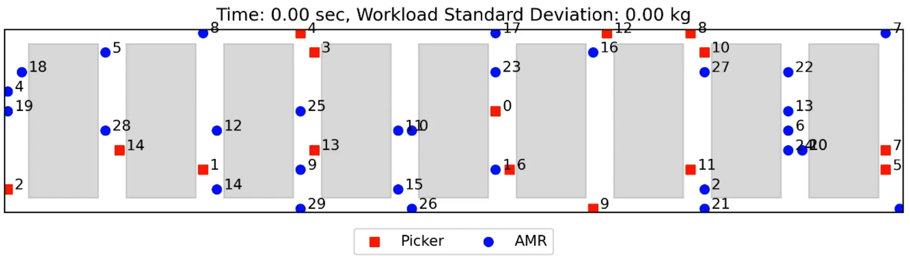
The main observed difference between the policies in these videos is how the pickers and AMRs are being spread through the warehouse. We find that the DRL policy managed to keep all pickers and AMRs distributed over many aisles. Oppositely, the greedy controller and VI benchmark method struggled to do so. In their runs, eventually, the entities got accumulated together into a few aisles. These processes can be clearly illustrated by considering several timestamps within the simulations. Figures 8.11, 8.12, and 8.13 illustrate these timestamps.

[1]cf. the following video links: https://youtu.be/LqF-8DEkXsg, https://youtu.be/wyIyajqqfjU, https://youtu.be/HctW6a69XN8.

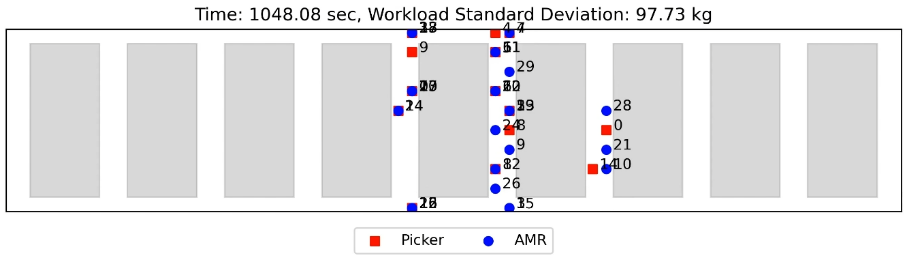
The first timestamp that we consider is after roughly 1000 seconds. At this moment, we observe that the greedy solution has already accumulated all pickers and AMRs in just three aisles. The VI benchmark keeps a better spread through the system. Most aisles still contain pickers and AMRs. Thus, the VI benchmark managed to keep the entities in the warehouse distributed longer. Similarly, the DRL optimizer maintained an even spread through the warehouse. After 2500 seconds, the distribution of agents over the aisles was still very balanced for the DRL agent. The greedy policy, on the other hand, did not manage to escape the accumulation of pickers and AMRs, and all were still close to each other. In addition, the VI benchmark also started accumulating the process within a few aisles, although the distribution was spread more than for the greedy policy. Lastly, after 5500 seconds, both the greedy policy and VI benchmark accumulated all pickers and AMRs within a few aisles. Oppositely, the DRL policy, which is near completion, still distributed all activity over many aisles.

Thus, the DRL policy managed to maintain a consistent spread of pickers and AMRs through the warehouse, whereas the benchmark methods accumulated all activity within a few aisles, after which they could not escape this accumulation. This explains the performance difference for two main reasons. First, having a more even spread over the warehouse prevents congestion. With many AMRs in the same aisle, many overtaking maneuvers must be performed, which can significantly slow down the movement process. Second, with many pickers and AMRs close to each other, pickers pick items for AMRs that other pickers can easily and efficiently pick. Thus, each picker is used less efficiently than when each picker takes care of an independent set of AMRs in a more spread-out system.

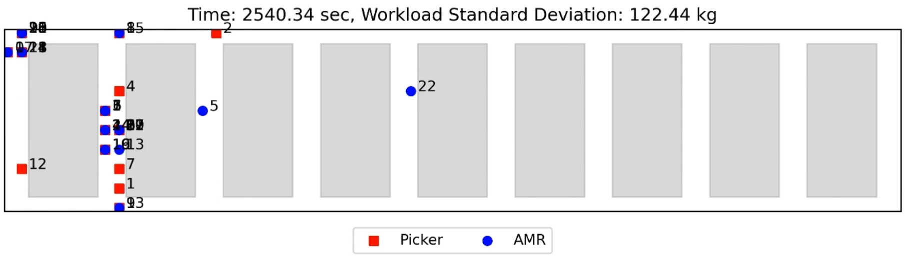
Lastly, when inspecting the simulations for a more extended time, we can also observe that in the VI benchmark run, the pickers seem to be staying within the same region for longer than in the other runs. This may cause the relatively high workload standard deviation of the VI benchmark. Namely, pickers continue to pick in regions with the same products for longer, and thus, when one region contains multiple light products and the other many heavy products, the workloads are not balanced well, and the fairness decreases.



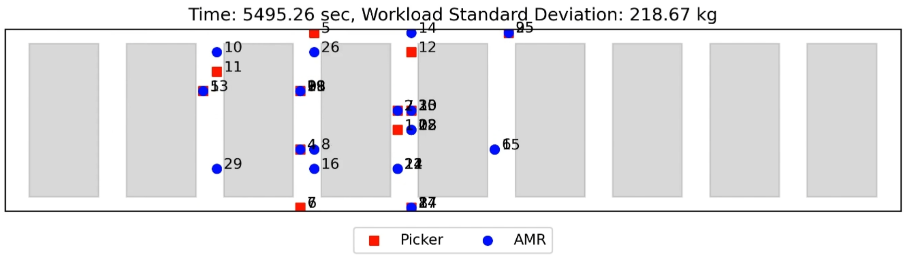
(a) 0 seconds.



(b) 1000 seconds.

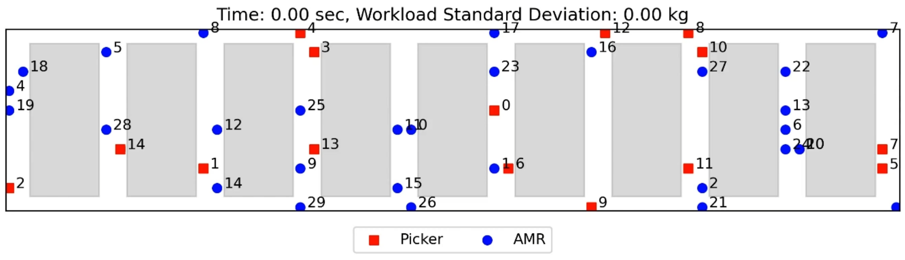


(c) 2500 seconds.

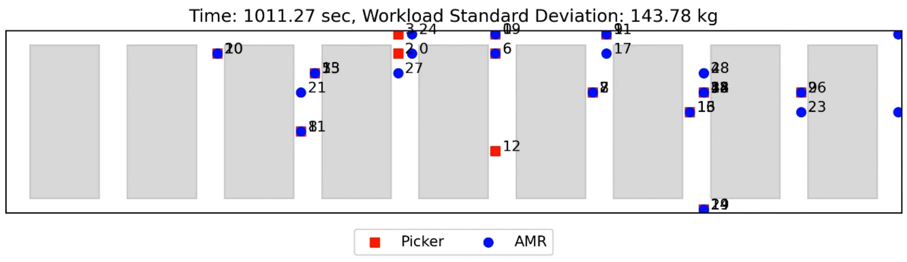


(d) 5500 seconds.

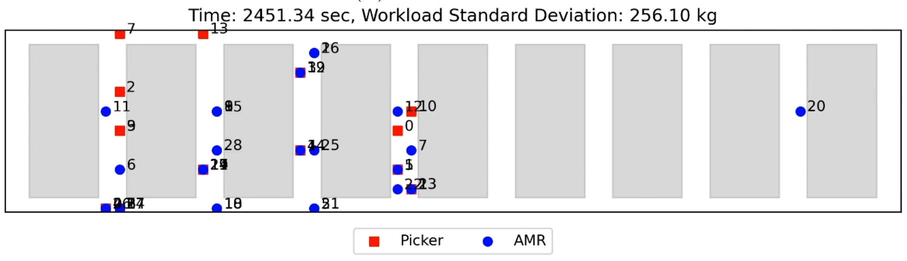
Figure 8.11: Snapshots of the policy visualization of the greedy policy.



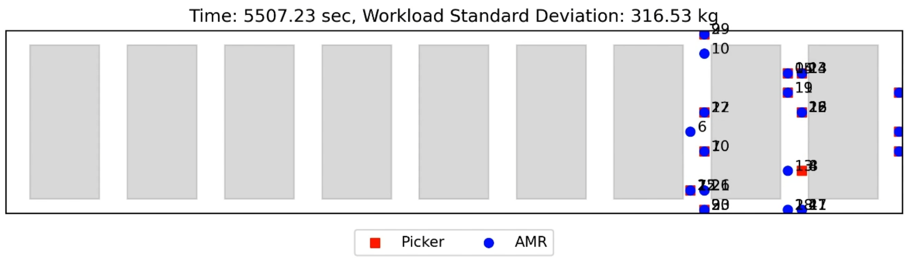
(a) 0 seconds.



(b) 1000 seconds.

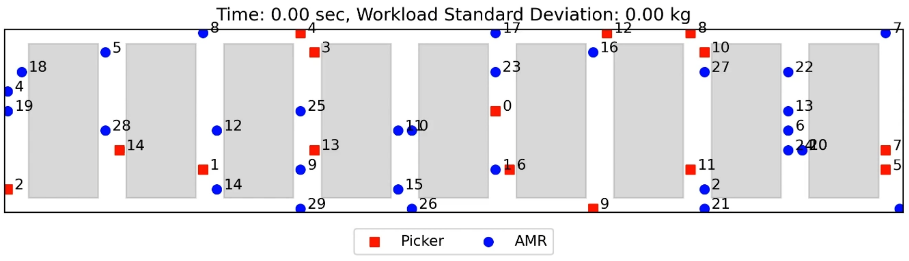


(c) 2500 seconds.

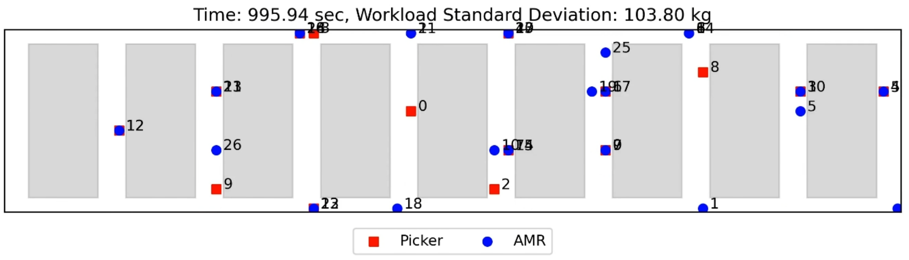


(d) 5500 seconds.

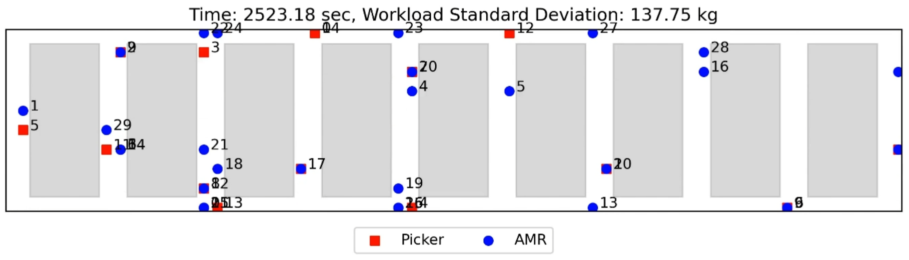
Figure 8.12: Snapshots of the policy visualization of the VI benchmark policy.



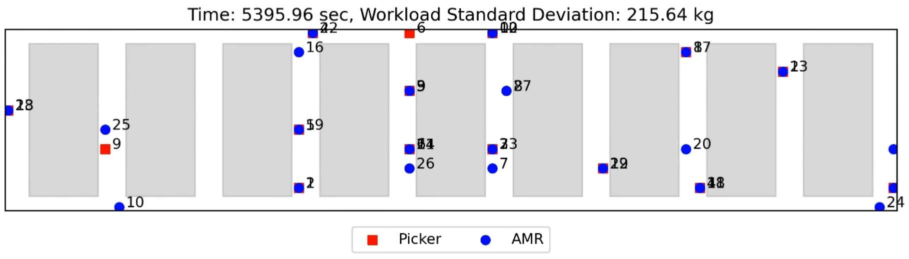
(a) 0 seconds.



(b) 1000 seconds.



(c) 2500 seconds.



(d) 5500 seconds.

Figure 8.13: Snapshots of the policy visualization of the DRL policy.

Chapter 9

Conclusions and Recommendations

In this study, we answered how to solve the problem of sequentially allocating pickers to orders in a collaborative order picking system with performance considerations in terms of pick efficiency and workload fairness.

We proposed a novel DRL approach to create a picker allocation optimizer. We used multi-objective learning to include efficiency and workload fairness objectives. To this end, we outlined the state-of-the-art multi-objective DRL methods. In addition, we explored how fairness is incorporated into optimization problems and which workload metrics are feasible. Based on these results and in consultation with business stakeholders, we used the total completion time of a series of pickruns as the efficiency objective and the standard deviation of lifted masses by the human pickers as the workload fairness metric.

To create our DRL approach, we use a graph structure that represents warehouses with node features to describe the current state of the system. The DRL policy receives requests from pickers to be allocated and assigns pickers to new destinations to fulfill a pick. We defined our actor architectures by combining neural networks with warehouse domain knowledge. Namely, we introduced the principles of aisle-embeddings and feature separation in our architecture. To train our agents, we adopted the PPO and PGMORL learning algorithms.

For training and evaluation of our approach, we developed a discrete-event simulation model of the collaborative picking process. We included uncertainty, congestion, and real product data to mimic the complexity of real-world processes. Consequently, we performed elaborate experiments. First, we tested which efficiency levels our approach could reach without considering fairness, and then we evaluated how efficiency and workload fairness can be balanced. Our method provided multiple policies that outline the potential trade-offs between efficiency and fairness. Lastly, we analyzed the designed policies using a decision tree analysis and visualizing the system behavior.

9.1 Main Findings

Our study showed that our DRL approach could learn good policies to allocate pickers to new orders in a collaborative picking system. We found significant efficiency and workload fairness improvements over the benchmark methods Vanderlande currently uses.

First, DRL policies focussing purely on efficiency consistently achieved improved order completion times over the benchmark policies. For large warehouse instances with 35 aisles and 2800 pick locations, our method achieved efficiency improvements of up to 40% over the method that Vanderlande currently considers, while for smaller warehouses, the improvements were generally over 20%. The learned policies adapted well to changing numbers of pickers and AMRs in the warehouses, as well as warehouses with different sizes and product locations. Thus, the policies can be generally applied and do not need to be retrained or adjusted for each change in warehouse layouts or product distribution, greatly improving the practical value. Our approach can be applied not only to increase the pick capacities of warehouses but also to reduce costs. For example, for the large warehouse instances, our DRL approach could achieve similar efficiency using 50 pickers and 120 AMRs as the current Vanderlande method did with 80 pickers and 220 AMRs. This is an almost 40% decrease in human pickers and 45% in AMRs. Considering just the saved human pickers in a single warehouse, which runs for 16 hours a day at an hourly tariff of €15, would mean cost savings of €7200 per day. If this warehouse operates 350 days a year, this would already amount to €2,520,000 of savings. These calculations are crude, but they highlight the potential benefits of our method. From visual inspection, we found that our methods keep a better spread of pickers through the warehouse than the benchmark methods, preventing congestion and making more efficient use of the resources.

Aside from efficiency, we showed that our multi-objective DRL method could achieve policies that balance fairness and efficiency well. We showed that we can find sets of policies that outline the trade-offs between the two considerations. Through this method, we provide insights to decision-makers regarding the possibilities such that they can make informed decisions about which policy they desire to use. These policy sets showed that we could significantly improve workload fairness while only sacrificing limited efficiency performance. Thus, although the objectives seem highly conflicting at first glance, the DRL solutions successfully mitigate the conflicts between the objectives. The policies can achieve efficient allocations for pickers while correcting the behavior when picker workloads become more variable. As a result, the policy sets all contained multiple policies that increased both efficiency and workload fairness compared to the benchmark methods. For instance, on the large warehouses, we found a policy that improved the total picking times by 23.6% compared to the current method by Vanderlande while simultaneously reducing the workload standard deviation by 92%. The multi-objective policies also transferred well to warehouses with different pickers and AMR numbers. Similarly, in general, the policies adapted sufficiently well to varying warehouse sizes, although for large differences in scale, the policies scaled less easily than the single-objective policies. Overall, the multi-objective policies offer suitable methods that can be applied in practice.

Lastly, we evaluated how to create more understandable policies using decision trees extracted from the DRL policies. We found that deeper decision tree policies concede a small efficiency loss and a slightly bigger fairness loss compared to the DRL policies. On the other hand, they are more interpretable and provide insights into a guaranteed, fixed set of rules that are applied. Thus, they offer a suitable alternative if black box policies are undesired.

In short, our DRL method can generate picker allocation policies to allocate pickers to pick locations in a collaborative order picking system. These policies can be applied in warehouses with uncertainty and disturbances. In addition, they can handle changing warehouse sizes and numbers of pickers and AMRs well. We can generate policies that significantly outperform the benchmark methods in efficiency and fairness. Moreover, using our multi-objective approach, we can provide policy sets that outline the trade-offs between the two such that decision-makers can make informed decisions on which policy to use based on their preferences.

9.2 Business Recommendations

We recommend that Vanderlande integrates DRL policies in their collaborative order picking concept. To do so, the DRL picker allocation policies should be considered as one of the core considerations of the collaborative picking concept. As Vanderlande is in the process of further developing the collaborative order picking solutions, they should consider incorporating our method sooner rather than later.

Namely, during the concept development phase that they are in, they still have many opportunities to integrate our policies into their pilot tests. This allows them to consider the setup of the hardware and software requirements of the DRL policies as an integral part of the concept development. The main barrier to implementing DRL policies is often transferring the simulation-based policies to real-world systems and evaluating the performance within those systems. By integrating the DRL solution from scratch, Vanderlande can manage this process well. We advise them to consider a pilot test of the collaborative picking system, which includes our method.

In addition, we suggest collaborating with customers to further clarify their needs and expand the alignment of the DRL policies with those needs. For instance, we suggest collaborating with customers to define relevant workload metrics. We showed the capabilities of our method using a relatively generic workload measure. By incorporating metrics that consider multiple factors, picker welfare can be improved more and more. In addition, using extensive product and order data, as well as warehouse information, can help to fine-tune the DRL policies to get the best possible results

Our overarching message to Vanderlande is to start working toward the practical implementation of our concept in a timely manner. In this study, we showed that the approach has much potential to provide significant efficiency and fairness improvements over existing approaches. The relevant real-world considerations, barriers, and chances can be understood and managed only by starting the development toward real-world implementation. In addition, it allows them to set up a workflow to bring DRL solutions into practice. By doing so, Vanderlande can start its journey toward successful integration of DRL approaches in real-time decision-making, paving the way for many more applications to follow.

9.3 Limitations and Future Research

One of the main limitations of our method is the dependence on the used simulation model. As we did not have extensive data for many days in various warehouses, we could not incorporate real-world problem instances. In addition, it was challenging to incorporate real-world disturbance scenarios as data for this was unavailable. We did use real-world product data and assumptions from domain experts to increase trust in the results as much as possible. However, it remains an estimation. As policies were trained and evaluated on the simulation instances, the exact performance improvement numbers may vary in real warehouses. However, with the large performance improvements we found, we can still be confident that the policies can achieve good results in real-life warehouses.

In addition, some of the features that we used incorporate estimated times. If these estimated times are too far from the real-world warehouse parameters, this may skew the results. However, these features can be adapted with relative ease to either newly sampled estimates or more robust values, such as distances instead of expected travel times. Hence, we do not expect this to affect the applicability in practice a lot.

We identified several lines of future research to expand on our study. Firstly, related to the above limitations, we suggest applied research into transferring DRL policies to real-world applications. There is a lack of established frameworks to steer such processes. Researchers and companies like Vanderlande can collaborate on implementing DRL methods to understand the chances and obstacles in these transitions. By establishing general principles or frameworks for such studies, the adoption of DRL in practice can be considerably improved.

Secondly, future studies can extend the scope of DRL methods in collaborative picking. Besides picker allocation, many other factors influence the performance, such as order batching and the release process of AMR pickruns. Future works could focus on integrating these decisions into the solution methodology. For example, cooperative multi-agent DRL could be used, with one agent being the picker optimizer and the other handling the releasing strategy. In addition, more objectives, such as minimizing order tardiness, can be incorporated into the multi-objective framework.

Thirdly, additional research can be done into modeling warehouses using different graph structures. As we saw, message passing networks did not work well on the graph structure that we used. We overcame this by proposing a new neural network architecture. Instead, other research streams could focus on developing warehouse graph structures that fit better with message passing networks. In addition, more neural network architectures that may be capable of capturing long-range dependencies can be explored. This is an active research field in which, for example, the hierarchical graph neural network (Rampasek & Wolf, 2021) has been shown to better capture long-term dependencies in graphs with few node features. Further developments in this field may be valuable for large warehouse graphs.

Fourthly, different learning algorithms may be explored. With further advancement of the multi-objective DRL research fields, better and more efficient algorithms may become available. Instead of model-free methods, model-based DRL algorithms can be studied and applied to collaborative picking problems. If model-based algorithms perform well, integrating model-based DRL with multi-objective DRL offers possibilities for future works.

As a final remark, we hope our study stimulates research into the integration between fairness and performance in the optimization domain. In operations management, there are many systems in which humans are crucial for performance. This should not be disregarded when optimizing these systems. Applying methods such as multi-objective DRL or other multi-objective optimization techniques to actively balance performance and fairness, and to outline their trade-offs has received little attention. This research stream provides a vast range of opportunities, and we hope that future research will be done to not only make operations more efficient but also improve the human experiences within these systems.

Bibliography

Abdelfattah, S., Merrick, K., & Hu, J. (2021). Intrinsically motivated hierarchical policy learning in multiobjective markov decision processes. *IEEE Transactions on Cognitive and Developmental Systems*, *13*, 262–273. https://doi.org/10.1109/TCDS.2019.2948025

Abdolmaleki, A., Huang, S., Hasenclever, L., Neunert, M., Song, F., Zambelli, M., Martins, M., Heess, N., Hadsell, R., & Riedmiller, M. (2020). A distributional view on multi-objective policy optimization. In H. D. III & A. Singh (Eds.). PMLR. https://proceedings.mlr.press/v119/abdolmaleki20a.html

Abels, A., Roijers, D., Lenaerts, T., Nowé, A., & Steckelmacher, D. (2019). Dynamic weights in multi-objective deep reinforcement learning. In K. Chaudhuri & R. Salakhutdinov (Eds.). PMLR. https://proceedings.mlr.press/v97/abels19a.html

Alabi, D., Immorlica, N., & Kalai, A. (2018). Unleashing linear optimizers for group-fair learning and optimization. In S. Bubeck, V. Perchet, & P. Rigollet (Eds.). PMLR. https://proceedings.mlr.press/v75/alabi18a.html

Alomrani, M. A., Moravej, R., & Khalil, E. B. (2021). Deep policies for online bipartite matching: A reinforcement learning approach. https://doi.org/10.48550/arxiv.2109.10380

Amaldi, E., Capone, A., Coniglio, S., & Gianoli, L. G. (2013). Network optimization problems subject to max-min fair flow allocation. *IEEE Communications Letters*, *17*, 1463–1466. https://doi.org/10.1109/LCOMM.2013.060513.130351

Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., & Zaremba, W. (2017). Hindsight experience replay. *NIPS*, 5055–5065. http://papers.nips.cc/paper/7090-hindsight-experience-replay

Arribas, E., Mancuso, V., & Cholvi, V. (2019). Fair cellular throughput optimization with the aid of coordinated drones. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 295–300. https://doi.org/10.1109/INFCOMW.2019.8845248

Arribas, E., Mancuso, V., & Cholvi, V. (2022). Fair throughput optimization with a dynamic network of drone relays. http://arxiv.org/abs/2207.04955

Azadeh, K., Koster, R. D., & Roy, D. (2019). Robotized and automated warehouse systems: Review and recent developments. *Transportation Science*, *53*, 917–945. https://doi.org/10.1287/trsc.2018.0873

Azadeh, K., Roy, D., & Koster, M. B. M. D. (2020). Dynamic human-robot collaborative picking strategies. *Available at SSRN 3585396.*

Bacciu, D., Errica, F., Micheli, A., & Podda, M. (2020). A gentle introduction to deep learning for graphs. *Neural Networks*, *129*, 203–221. https://doi.org/10.1016/j.neunet.2020.06.006

Barocas, S., Hardt, M., & Narayanan, A. (2019). *Fairness and machine learning*. fairmlbook.org. http://www.fairmlbook.org

Beeks, M., Afshar, R. R., Zhang, Y., Dijkman, R., Dorst, C. V., & Looijer, S. D. (2022). Deep reinforcement learning for a multi-objective online order batching problem. *Proceedings* *of the International Conference on Automated Planning and Scheduling*, *32*, 435–443. https://doi.org/10.1609/icaps.v32i1.19829

Bertin, R., Hunold, S., Legrand, A., & Touati, C. (2014). Fair scheduling of bag-of-tasks applications using distributed lagrangian optimization. *Journal of Parallel and Distributed Computing*, *74*, 1914–1929. https://doi.org/10.1016/j.jpdc.2013.08.011

Borg, G. (1982). A category scale with ratio properties for intermodal and interindividual comparisons. *Psychophysical judgment and the process of perception*, 25–34.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. http://arxiv.org/abs/1606.01540

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). API design for machine learning software: Experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

Busa-Fekete, R., Szörényi, B., Weng, P., & Mannor, S. (2017). Multi-objective bandits: Optimizing the generalized gini index. In D. Precup & Y. W. Teh (Eds.). PMLR. https://proceedings.mlr.press/v70/busa-fekete17a.html

Canese, L., Cardarilli, G. C., Nunzio, L. D., Fazzolari, R., Giardino, D., Re, M., & Spanò, S. (2021). Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, *11*, 4948. https://doi.org/10.3390/app11114948

Caputo, A. C., & Pelagagge, P. M. (2006). Management criteria of automated order picking systems in high-rotation high-volume distribution centers. *Industrial Management & Data Systems*, *106*, 1359–1383. https://doi.org/10.1108/02635570610712627

Chen, D., Qi, Q., Zhuang, Z., Wang, J., Liao, J., & Han, Z. (2021). Mean field deep reinforcement learning for fair and efficient uav control. *IEEE Internet of Things Journal*, *8*, 813–828. https://doi.org/10.1109/JIOT.2020.3008299

Chen, X., Ghadirzadeh, A., Bjorkman, M., & Jensfelt, P. (2019). Meta-learning for multi-objective reinforcement learning. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 977–983. https://doi.org/10.1109/IROS40897.2019.8968092

Clausen, A., Umair, A., Demazeau, Y., & Jørgensen, B. N. (2020). Impact of social welfare metrics on energy allocation in multi-objective optimization. *Energies*, *13*, 2961. https://doi.org/10.3390/en13112961

Corbett-Davies, S., & Goel, S. (2018). The measure and mismeasure of fairness: A critical review of fair machine learning. https://doi.org/10.48550/arxiv.1808.00023

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, *6*, 80–91. https://doi.org/10.1287/mnsc.6.1.80

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. https://dask.org

Dely, P., D'Andreagiovanni, F., & Kassler, A. (2015). Fair optimization of mesh-connected wlan hotspots. *Wireless Communications and Mobile Computing*, *15*, 924–946. https://doi.org/10.1002/wcm.2393

Devarajan, R., Jha, S. C., Phuyal, U., & Bhargava, V. K. (2012). Energy-aware resource allocation for cooperative cellular network using multi-objective optimization approach. *IEEE Transactions on Wireless Communications*, *11*, 1797–1807. https://doi.org/10.1109/TWC.2012.030512.110895

Diao, X., Zheng, J., Cai, Y., Wu, Y., & Anpalagan, A. (2019). Fair data allocation and trajectory optimization for uav-assisted mobile edge computing. *IEEE Communications Letters*, *23*, 2357–2361. https://doi.org/10.1109/LCOMM.2019.2943461

Ding, R., Gao, F., & Shen, X. S. (2020). 3d uav trajectory design and frequency band allocation for energy-efficient and fair communication: A deep reinforcement learning approach.

IEEE Transactions on Wireless Communications, *19*, 7796–7809. https://doi.org/10.1109/TWC.2020.3016024

Doi, T., Nishi, T., & Voß, S. (2018). Two-level decomposition-based matheuristic for airline crew rostering problems with fair working time. *European Journal of Operational Research*, *267*, 428–438. https://doi.org/10.1016/j.ejor.2017.11.046

Dornheim, J. (2022). Gtlo: A generalized and non-linear multi-objective deep reinforcement learning approach. https://doi.org/10.48550/arxiv.2204.04988

Dukic, G., & Oluic, C. (2007). Order-picking methods: Improving order-picking efficiency. *International Journal of Logistics Systems and Management*, *3*, 451. https://doi.org/10.1504/IJLSM.2007.013214

Escoffier, B., Gourvès, L., & Monnot, J. (2013). Fair solutions for some multiagent optimization problems. *Autonomous Agents and Multi-Agent Systems*, *26*, 184–201. https://doi.org/10.1007/s10458-011-9188-z

Fan, Z., Peng, N., Tian, M., & Fain, B. (2022). Welfare and fairness in multi-objective reinforcement learning. http://arxiv.org/abs/2212.01382

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. http://arxiv.org/abs/1903.02428

Gajane, P., & Pechenizkiy, M. (2017). On formalizing fairness in prediction with machine learning. https://doi.org/10.48550/arxiv.1710.03184

Gajane, P., Saxena, A., Tavakol, M., Fletcher, G., & Pechenizkiy, M. (2022). Survey on fair reinforcement learning: Theory and practice. https://doi.org/10.48550/arxiv.2205.10032

Gajšek, B., Šinko, S., Kramberger, T., Butlewski, M., Özceylan, E., & Đukić, G. (2021). Towards productive and ergonomic order picking: Multi-objective modeling approach. *Applied Sciences*, *11*, 4179. https://doi.org/10.3390/app11094179

Garg, A., Chaffin, D. B., & Herrin, G. D. (1978). Prediction of metabolic rates for manual materials handling jobs. *American Industrial Hygiene Association Journal*, *39*, 661–674. https://doi.org/10.1080/0002889778507831

Ghelichi, Z., & Kilaru, S. (2021). Analytical models for collaborative autonomous mobile robot solutions in fulfillment centers. *Applied Mathematical Modelling*, *91*, 438–457. https://doi.org/10.1016/j.apm.2020.09.059

Gong, H., & Guo, C. (2023). Influence maximization considering fairness: A multi-objective optimization approach with prior knowledge. *Expert Systems with Applications*, *214*, 119138. https://doi.org/10.1016/j.eswa.2022.119138

Gu, S., Holly, E., Lillicrap, T., & Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 3389–3396. https://doi.org/10.1109/ICRA.2017.7989385

Guo, X., Xu, Z., Zhang, J., Lu, J., & Zhang, H. (2020). An od flow clustering method based on vector constraints: A case study for beijing taxi origin-destination data. *ISPRS International Journal of Geo-Information*, *9*, 128. https://doi.org/10.3390/ijgi9020128

Gurobi Optimization, LLC. (2023). Gurobi Optimizer Reference Manual. https://www.gurobi.com

Harks, T. (2005). Utility proportional fair bandwidth allocation: An optimization oriented approach. In M. A. Marsan, G. Bianchi, M. Listanti, & M. Meo (Eds.). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-30573-6_5

Hasselt, H. V., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *30*. https://doi.org/10.1609/aaai.v30i1.10295

Hayes, C. F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L. M., Dazeley, R., Heintz, F., Howley, E., Irissappane, A. A., Mannion, P., Nowé, A., Ramos, G., Restelli, M., Vamplew, P., & Roijers, D. M. (2022).

A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, *36*, 26. https://doi.org/10.1007/s10458-022-09552-y

Heess, N., Hunt, J. J., Lillicrap, T. P., & Silver, D. (2015). Memory-based control with recurrent neural networks. https://doi.org/10.48550/arxiv.1512.04455

Hignett, S., & McAtamney, L. (2000). Rapid entire body assessment (reba). *Applied Ergonomics*, *31*, 201–205. https://doi.org/10.1016/S0003-6870(99)00039-3

Holler, J., Vuorio, R., Qin, Z., Tang, X., Jiao, Y., Jin, T., Singh, S., Wang, C., & Ye, J. (2019). Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem. http://arxiv.org/abs/1911.11260

Hoßfeld, T., Skorin-Kapov, L., Heegaard, P. E., & Varela, M. (2017). Definition of qoe fairness in shared systems. *IEEE Communications Letters*, *21*, 184–187. https://doi.org/10.1109/LCOMM.2016.2616342

Hu, C., Zhu, Z., Wang, L., Zhu, C., & Yang, Y. (2022). An improved multi-objective deep reinforcement learning algorithm based on envelope update. *Electronics*, *11*, 2479. https://doi.org/10.3390/electronics11162479

Ishida, T., Ueda, K., & Yakoh, T. (2006). Fairness and utilization in multipath network flow optimization. *2006 IEEE International Conference on Industrial Informatics*, 1096–1101. https://doi.org/10.1109/INDIN.2006.275770

Issabekov, R., & Vamplew, P. (2012). An empirical comparison of two common multiobjective reinforcement learning algorithms. In M. Thielscher & D. Zhang (Eds.). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35101-3_53

Jagtenberg, C., & Mason, A. (2020). Improving fairness in ambulance planning by time sharing. *European Journal of Operational Research*, *280*, 1095–1107. https://doi.org/10.1016/j.ejor.2019.08.003

Jain, R. K., Chiu, D.-M. W., Hawe, W. R., et al. (1984). A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, *21*.

Jiang, J., & Lu, Z. (2019). Learning fairness in multi-agent systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.). Curran Associates, Inc. https://dl.acm.org/doi/10.5555/3454287.3455528

Jiang, Y., Hu, Z., Liu, Z., & Zhang, H. (2023). Optimization of multi-objective airport gate assignment problem: Considering fairness between airlines. *Transportmetrica B: Transport Dynamics*, *11*, 196–210. https://doi.org/10.1080/21680566.2022.2056542

Jiao, Y., Tang, X., Qin, Z. T., Li, S., Zhang, F., Zhu, H., & Ye, J. (2020). A deep value-based policy search approach for real-world vehicle repositioning on mobility-on-demand platforms. *NeurIPS 2020 Deep Reinforcement Learning Workshop*.

Jin, J., Zhou, M., Zhang, W., Li, M., Guo, Z., Qin, Z., Jiao, Y., Tang, X., Wang, C., Wang, J., Wu, G., & Ye, J. (2019). Coride: Joint order dispatching and fleet management for multi-scale ride-hailing platforms. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 1983–1992. https://doi.org/10.1145/3357384.3357978

Karhu, O., Kansi, P., & Kuorinka, I. (1977). Correcting working postures in industry: A practical method for analysis. *Applied Ergonomics*, *8*, 199–201. https://doi.org/10.1016/0003-6870(77)90164-8

Karp, R. M., Vazirani, U. V., & Vazirani, V. V. (1990). An optimal algorithm for on-line bipartite matching. *Proceedings of the twenty-second annual ACM symposium on Theory of computing - STOC '90*, 352–358. https://doi.org/10.1145/100216.100262

Kermany, N. R., Zhao, W., Yang, J., Wu, J., & Pizzato, L. (2020). An ethical multi-stakeholder recommender system based on evolutionary multi-objective optimization. *2020 IEEE International Conference on Services Computing (SCC)*, 478–480. https://doi.org/10.1109/SCC49832.2020.00074

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. http://arxiv.org/abs/1412.6980

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. http://arxiv.org/abs/1609.02907

Kleinberg, J., Rabani, Y., & Tardos, É. (2001). Fairness in routing and load balancing. *Journal of Computer and System Sciences*, *63*, 2–20. https://doi.org/10.1006/jcss.2001.1752

Koppen, M., Verschae, R., Yoshida, K., & Tsuru, M. (2010). Comparison of evolutionary multi-objective optimization algorithms for the utilization of fairness in network control. *2010 IEEE International Conference on Systems, Man and Cybernetics*, 2647–2655. https://doi.org/10.1109/ICSMC.2010.5641898

Kuai, Z., Wang, T., & Wang, S. (2022). Fair virtual network function mapping and scheduling using proximal policy optimization. *IEEE Transactions on Communications*, *70*, 7434–7445. https://doi.org/10.1109/TCOMM.2022.3211071

Larco, J. A., de Koster, R., Roodbergen, K. J., & Dul, J. (2017). Managing warehouse efficiency and worker discomfort through enhanced storage assignment decisions. *International Journal of Production Research*, *55*, 6407–6422. https://doi.org/10.1080/00207543.2016.1165880

Law, A. M. (2015). *Simulation modeling and analysis* (5th International ed). Mcgraw-Hill.

Lee, H.-Y., & Murray, C. C. (2019). Robotics in order picking: Evaluating warehouse layouts for pick, place, and transport vehicle routing systems. *International Journal of Production Research*, *57*, 5821–5841. https://doi.org/10.1080/00207543.2018.1552031

Li, C., Ma, X., Xia, L., Zhao, Q., & Yang, J. (2020). Fairness control of traffic light via deep reinforcement learning. *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 652–658. https://doi.org/10.1109/CASE48305.2020.9216899

Li, F., Jun, W., Tan, Wentong, & Cai. (2022). Multi-objective optimization of clustering-based scheduling for multi-workflow on clouds considering fairness. http://arxiv.org/abs/2205.11173

Li, K., Zhang, T., & Wang, R. (2021). Deep reinforcement learning for multiobjective optimization. *IEEE Transactions on Cybernetics*, *51*, 3103–3114. https://doi.org/10.1109/TCYB.2020.2977661

Li, Z., Dong, C., Li, A., & Wang, H. (2016). Traffic offloading from lte-u to wifi: A multi-objective optimization approach. *2016 IEEE International Conference on Communication Systems (ICCS)*, 1–5. https://doi.org/10.1109/ICCS.2016.7833622

Li, Z., Xie, C., Peng, P., Gao, X., & Wan, Q. (2021). Multi-objective location-scale optimization model and solution methods for large-scale emergency rescue resources. *Environmental Science and Pollution Research*. https://doi.org/10.1007/s11356-021-12753-9

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. https://doi.org/10.48550/arxiv.1509.02971

Limmer, S., & Dietrich, M. (2018). Optimization of dynamic prices for electric vehicle charging considering fairness. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2304–2311. https://doi.org/10.1109/SSCI.2018.8628756

Liu, C. H., Chen, Z., Tang, J., Xu, J., & Piao, C. (2018). Energy-efficient uav control for effective and fair communication coverage: A deep reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, *36*, 2059–2070. https://doi.org/10.1109/JSAC.2018.2864373

Liu, F. Y., & Qian, C. (2021). Prediction guided meta-learning for multi-objective reinforcement learning. *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2171–2178. https://doi.org/10.1109/CEC45853.2021.9504972

Liu, X., Zhang, X., Li, W., & Zhang, X. (2017). Swarm optimization algorithms applied to multi-resource fair allocation in heterogeneous cloud computing systems. *Computing*, *99*, 1231–1255. https://doi.org/10.1007/s00607-017-0561-x

Liu, Y., Wu, F., Lyu, C., Li, S., Ye, J., & Qu, X. (2022). Deep dispatching: A deep reinforcement learning approach for vehicle dispatching on online ride-hailing platform. *Transportation Research Part E: Logistics and Transportation Review*, *161*, 102694. https://doi.org/10.1016/j.tre.2022.102694

Liu, Y., Huangfu, W., Zhou, H., Zhang, H., Liu, J., & Long, K. (2022). Fair and energy-efficient coverage optimization for uav placement problem in the cellular network. *IEEE Transactions on Communications*, *70*, 4222–4235. https://doi.org/10.1109/TCOMM.2022.3170615

Löffler, M., Boysen, N., & Schneider, M. (2022). Picker routing in agv-assisted order picking systems. *INFORMS Journal on Computing*, *34*, 440–462. https://doi.org/10.1287/ijoc.2021.1060

Lorson, F., Fügener, A., & Hübner, A. (2022). New team mates in the warehouse: Human interactions with automated and robotized systems. *IISE Transactions*, 1–18. https://doi.org/10.1080/24725854.2022.2072545

Lu, R., & Wang, K. (2021). Max-min energy-efficiency fair optimization in star-ris assisted system. http://arxiv.org/abs/2112.06495

Malencia, M., Kumar, V., Pappas, G., & Prorok, A. (2021). Fair robust assignment using redundancy. *IEEE Robotics and Automation Letters*, *6*, 4217–4224. https://doi.org/10.1109/LRA.2021.3067283

Mandal, D., & Gan, J. (2022). Socially fair reinforcement learning. https://doi.org/10.48550/arxiv.2208.12584

Mao, C., Liu, Y., & Shen, Z.-J. ((2020). Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, *115*, 102626. https://doi.org/10.1016/j.trc.2020.102626

Matloff, N. (2008). Introduction to discrete-event simulation and the simpy language. *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August*, *2*(2009), 1–33.

McAtamney, L., & Corlett, E. N. (1993). Rula: A survey method for the investigation of work-related upper limb disorders. *Applied Ergonomics*, *24*, 91–99. https://doi.org/10.1016/0003-6870(93)90080-S

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2019). A survey on bias and fairness in machine learning. https://doi.org/10.48550/arxiv.1908.09635

Meng, Q., & Khoo, H. L. (2010). A pareto-optimization approach for a fair ramp metering. *Transportation Research Part C: Emerging Technologies*, *18*, 489–506. https://doi.org/10.1016/j.trc.2009.10.001

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533. https://doi.org/10.1038/nature14236

Mossalam, H., Assael, Y. M., Roijers, D. M., & Whiteson, S. (2016). Multi-objective deep reinforcement learning. https://doi.org/10.48550/arXiv.1610.02707

Munguía-López, A., & Ponce-Ortega, J. M. (2021). Fair allocation of potential covid-19 vaccines using an optimization-based strategy. *Process Integration and Optimization for Sustainability*, *5*, 3–12. https://doi.org/10.1007/s41660-020-00141-8

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*, 91–95. https://doi.org/10.1016/0305-0483(83)90088-9

Nemer, I. A., Sheltami, T. R., Belhaiza, S., & Mahmoud, A. S. (2022). Energy-efficient uav movement control for fair communication coverage: A deep reinforcement learning approach. *Sensors*, *22*, 1919. https://doi.org/10.3390/s22051919

Nguyen, T. T., Nguyen, N. D., Vamplew, P., Nahavandi, S., Dazeley, R., & Lim, C. P. (2020). A multi-objective deep reinforcement learning framework. *Engineering Applications of Artificial Intelligence*, *96*, 103915. https://doi.org/10.1016/j.engappai.2020.103915

Nguyen, V. H., & Weng, P. (2017). An efficient primal-dual algorithm for fair combinatorial optimization problems. In X. Gao, H. Du, & M. Han (Eds.). Springer International Publishing. https://doi.org/10.1007/978-3-319-71150-8_28

Nian, X., Irissappane, A. A., & Roijers, D. (2020). Dcrac: Deep conditioned recurrent actor-critic for multi-objective partially observable environments. *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, 931–938.

Nicholson, A., & Wong, Y.-D. (1993). Are accidents poisson distributed? a statistical test. *Accident Analysis & Prevention*, *25*, 91–97. https://doi.org/10.1016/0001-4575(93)90100-B

Niu, Y., & Schulte, F. (2021). Human aspects in collaborative order picking – what if robots learned how to give humans a break? In A. Dolgui, A. Bernard, D. Lemoine, G. von Cieminski, & D. Romero (Eds.). Springer International Publishing.

Niu, Y., Schulte, F., & Negenborn, R. R. (2021). Human aspects in collaborative order picking – letting robotic agents learn about human discomfort. *Procedia Computer Science*, *180*, 877–886. https://doi.org/10.1016/j.procs.2021.01.338

Partov, B., Leith, D. J., & Razavi, R. (2015). Utility fair optimization of antenna tilt angles in lte networks. *IEEE/ACM Transactions on Networking*, *23*, 175–185. https://doi.org/10.1109/TNET.2013.2294965

Pasparakis, A., de Vries, J., & de Koster, M. R. (2021). In control or under control? human-robot collaboration in warehouse order picking. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3816533

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. http://arxiv.org/abs/1912.01703

Pedreshi, D., Ruggieri, S., & Turini, F. (2008). Discrimination-aware data mining. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 560–568. https://doi.org/10.1145/1401890.1401959

Petersen, C. G. (1997). An evaluation of order picking routeing policies. *International Journal of Operations & Production Management*, *17*, 1098–1111. https://doi.org/10.1108/01443579710177860

Pioro, M. (2007). Fair routing and related optimization problems. *15th International Conference on Advanced Computing and Communications (ADCOM 2007)*, 229–235. https://doi.org/10.1109/ADCOM.2007.121

Pishdad, L., Rabiee, H. R., & Mirarmandehi, N. (2010). A fair optimization scheduling scheme for ieee 802.16 networks in multimedia applications. *Journal of Visual Communication and Image Representation*, *21*, 167–174. https://doi.org/10.1016/j.jvcir.2009.05.004

Purushothaman, K. E., & Nagarajan, V. (2021). Evolutionary multi-objective optimization algorithm for resource allocation using deep neural network in 5g multi-user massive mimo. *International Journal of Electronics*, *108*, 1214–1233. https://doi.org/10.1080/00207217.2020.1843715

Qi, H., Hu, Z., Huang, H., Wen, X., & Lu, Z. (2020). Energy efficient 3-d uav control for persistent communication service and fairness: A deep reinforcement learning approach. *IEEE Access*, *8*, 53172–53184. https://doi.org/10.1109/ACCESS.2020.2981403

Qin, Z., Liu, Z., Han, G., Lin, C., Guo, L., & Xie, L. (2021). Distributed uav-bss trajectory optimization for user-level fair communication service with multi-agent deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, *70*, 12290–12301. https://doi.org/10.1109/TVT.2021.3117792

Raeis, M., & Leon-Garcia, A. (2021). A deep reinforcement learning approach for fair traffic signal control. *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2512–2518. https://doi.org/10.1109/ITSC48978.2021.9564847

Rahmattalabi, A., Jabbari, S., Lakkaraju, H., Vayanos, P., Izenberg, M., Brown, R., Rice, E., & Tambe, M. (2021). Fair influence maximization: A welfare optimization approach. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*, 11630–11638. https://doi.org/10.1609/aaai.v35i13.17383

Rampasek, L., & Wolf, G. (2021). Hierarchical graph neural nets can capture long-range interactions. *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, 1–6. https://doi.org/10.1109/MLSP52302.2021.9596069

Ratliff, H. D., & Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, *31*, 507–521. https://doi.org/10.1287/opre.31.3.507

Raza, S. M., Sajid, M., & Singh, J. (2022). Vehicle routing problem using reinforcement learning: Recent advancements (D. Gupta, K. Sambyo, M. Prasad, & S. Agarwal, Eds.). *Advanced Machine Intelligence and Signal Processing*, 269–280. https://doi.org/10.1007/978-981-19-0840-8_20

Reymond, M., & Nowe, A. (2019). Pareto-dqn: Approximating the pareto front in complex multi-objective decision problems. *Proceedings of the Adaptive and Learning Agents Workshop 2019 (ALA-19) at AAMAS.*

Roijers, D. M., Vamplew, P., Whiteson, S., & Dazeley, R. (2013). A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, *48*, 67–113. https://doi.org/10.1613/jair.3987

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*, 455–472. https://doi.org/10.1287/trsc.1050.0135

Sadiq, A. S., Dehkordi, A. A., Mirjalili, S., & Pham, Q.-V. (2022). Nonlinear marine predator algorithm: A cost-effective optimizer for fair power allocation in noma-vlc-b5g networks. *Expert Systems with Applications*, *203*, 117395. https://doi.org/10.1016/j.eswa.2022.117395

Sawik, T. (2015). On the fair optimization of cost and customer service level in a supply chain under disruption risks. *Omega*, *53*, 58–66. https://doi.org/10.1016/j.omega.2014.12.004

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. https://doi.org/10.48550/arxiv.1707.06347

Siddique, U., Weng, P., & Zimmer, M. (2020). Learning fair policies in multi-objective (deep) reinforcement learning with average and discounted rewards. In H. D. III & A. Singh (Eds.). PMLR. https://proceedings.mlr.press/v119/siddique20a.html

Silver, D. (2015). Lectures on reinforcement learning. https://www.davidsilver.uk/teaching/

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*, 484–489. https://doi.org/10.1038/nature16961

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, *104*, 333–339. https://doi.org/10.1016/j.jbusres.2019.07.039

Song, W., Chen, X., Li, Q., & Cao, Z. (2023). Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, *19*, 1600–1610. https://doi.org/10.1109/TII.2022.3189725

Srinivas, S., & Yu, S. (2022). Collaborative order picking with multiple pickers and robots: Integrated approach for order batching, sequencing and picker-robot routing. *International Journal of Production Economics*, *254*, 108634. https://doi.org/10.1016/j.ijpe.2022.108634

Stolletz, R., & Brunner, J. O. (2012). Fair optimization of fortnightly physician schedules with flexible shifts. *European Journal of Operational Research*, *219*, 622–629. https://doi.org/10.1016/j.ejor.2011.10.038

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

Tajmajer, T. (2018). Modular multi-objective deep reinforcement learning with decision values. *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 85–93.

Tang, X., He, Y., Qi, P., Chang, Z., Jiang, M., & Dai, Z. (2021). A new multi-objective optimization model of water resources considering fairness and water shortage risk. *Water*, *13*, 2648. https://doi.org/10.3390/w13192648

Tangpattanakul, P., Jozefowiez, N., & Lopez, P. (2012). Multi-objective optimization for selecting and scheduling observations by agile earth observing satellites. In C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, & M. Pavone (Eds.). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-32964-7_12

Tong, L., Chen, Y., Zhou, X., & Sun, Y. (2021). Qoe-fairness tradeoff scheme for dynamic spectrum allocation based on deep reinforcement learning. *The 5th International Conference on Computer Science and Application Engineering*, 1–7. https://doi.org/10.1145/3487075.3487137

Vouros, G. A. (2023). Explainable deep reinforcement learning: State of the art and challenges. *ACM Computing Surveys*, *55*, 1–39. https://doi.org/10.1145/3527448

Wang, Y., Zhang, M., Ao, J., Wang, Z., Dong, H., & Zeng, M. (2022). Profit allocation strategy of virtual power plant based on multi-objective optimization in electricity market. *Sustainability*, *14*, 6229. https://doi.org/10.3390/su14106229

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. https://doi.org/10.48550/arxiv.1511.06581

Waters, T. R., Putz-Anderson, V., Garg, A., & Fine, L. J. (1993). Revised niosh equation for the design and evaluation of manual lifting tasks. *Ergonomics*, *36*, 749–776. https://doi.org/10.1080/00140139308967940

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292. https://doi.org/10.1007/BF00992698

Wen, J., Zhao, J., & Jaillet, P. (2017). Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 220–225. https://doi.org/10.1109/ITSC.2017.8317908

Weng, J., Chen, H., Yan, D., You, K., Duburcq, A., Zhang, M., Su, Y., Su, H., & Zhu, J. (2022). Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, *23*(267), 1–6. http://jmlr.org/papers/v23/21-1127.html

Wu, H., Ma, C., Mitra, B., Diaz, F., & Liu, X. (2023). A multi-objective optimization framework for multi-stakeholder fairness-aware recommendation. *ACM Transactions on Information Systems*, *41*, 1–29. https://doi.org/10.1145/3564285

Xie, L., Li, H., & Luttmann, L. (2022). Formulating and solving integrated order batching and routing in multi-depot agv-assisted mixed-shelves warehouses. *European Journal of Operational Research.* https://doi.org/10.1016/j.ejor.2022.08.047

Xu, J., Tian, Y., Ma, P., Rus, D., Sueda, S., & Matusik, W. (2020). Prediction-guided multi-objective reinforcement learning for continuous robot control. In H. D. III & A. Singh (Eds.). PMLR. https://proceedings.mlr.press/v119/xu20h.html

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? http://arxiv.org/abs/1810.00826

Xu, L., Li, Y.-p., Li, Q.-m., Yang, Y.-w., Tang, Z.-m., & Zhang, X.-f. (2015). Proportional fair resource allocation based on hybrid ant colony optimization for slow adaptive ofdma system. *Information Sciences*, *293*, 1–10. https://doi.org/10.1016/j.ins.2014.09.028

Yaacoub, E., & Dawy, Z. (2014). Fair optimization of video streaming quality of experience in lte networks using distributed antenna systems and radio resource management (D. Nace, Ed.). *Journal of Applied Mathematics*, *2014*, 1–12. https://doi.org/10.1155/2014/562079

Yang, F., Huang, H., Shi, W., Ma, Y., Feng, Y., Cheng, G., & Liu, Z. (2022). Pmdrl: Pareto-front-based multi-objective deep reinforcement learning. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-022-04232-x

Yang, R., Sun, X., & Narasimhan, K. (2019). A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/4a46fbfca3f1465a27b210f4bdfe6ab3-Paper.pdf

Ye, Q. C., Zhang, Y., & Dekker, R. (2017). Fair task allocation in transportation. *Omega*, *68*, 1–16. https://doi.org/10.1016/j.omega.2016.05.005

Yu, J. J. Q., Yu, W., & Gu, J. (2019). Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, *20*, 3806–3817. https://doi.org/10.1109/TITS.2019.2909109

Yue, D., & You, F. (2014). Fair profit allocation in supply chain optimization with transfer price and revenue sharing: Minlp model and algorithm for cellulosic biofuel supply chains. *AIChE Journal*, *60*, 3211–3229. https://doi.org/10.1002/aic.14511

Zhang, M., Winkelhaus, S., & Grosse, E. H. (2021). Evaluation of human workload in a hybrid order picking system. *IFAC-PapersOnLine*, *54*, 458–463. https://doi.org/10.1016/j.ifacol.2021.08.053

Zhang, Z., Qin, H., & Li, Y. (2020). Multi-objective optimization for the vehicle routing problem with outsourcing and profit balancing. *IEEE Transactions on Intelligent Transportation Systems*, *21*, 1987–2001. https://doi.org/10.1109/TITS.2019.2910274

Zhao, J. (2019). Optimizations with intelligent reflecting surfaces (irss) in 6g wireless networks: Power control, quality of service, max-min fair beamforming for unicast, broadcast, and multicast with multi-antenna mobile users and multiple irss. http://arxiv.org/abs/1908.03965

Zhou, S.-Z., Zhan, Z.-H., Chen, Z.-G., Kwong, S., & Zhang, J. (2021). A multi-objective ant colony system algorithm for airline crew rostering problem with fairness and satisfaction. *IEEE Transactions on Intelligent Transportation Systems*, *22*, 6784–6798. https://doi.org/10.1109/TITS.2020.2994779

Zhou, Y., Pahwa, A., & Yang, S.-S. (2006). Modeling weather-related failures of overhead distribution lines. *IEEE Transactions on Power Systems*, *21*, 1683–1690. https://doi.org/10.1109/TPWRS.2006.881131

Zhu, Q., & Oh, J. (2018). Deep reinforcement learning for fairness in distributed robotic multi-type resource allocation. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 460–466. https://doi.org/10.1109/ICMLA.2018.00075

Zimmer, M., Glanois, C., Siddique, U., & Weng, P. (2021). Learning fair policies in decentralized cooperative multi-agent reinforcement learning. In M. Meila & T. Zhang (Eds.). PMLR. https://proceedings.mlr.press/v139/zimmer21a.html

Zou, Y., Zhang, D., & Qi, M. (2019). Order picking system optimization based on picker-robot collaboration. *Proceedings of the 2019 5th International Conference on Industrial and Business Engineering*, 1–6. https://doi.org/10.1145/3364335.3364386

Žulj, I., Salewski, H., Goeke, D., & Schneider, M. (2022). Order batching and batch sequencing in an amr-assisted picker-to-parts system. *European Journal of Operational Research*, *298*, 182–201. https://doi.org/10.1016/j.ejor.2021.05.033

Appendices

Appendix A

Expanded Mathematical Formulation

Indices and Sets

$i \in \mathcal{N}$	Set of all items that must be picked. Note that multiple items can have the same location.
$r \in \mathcal{R}$	Set of AMRs.
$k \in \mathcal{K}$	Set of human pickers.
$t \in \mathcal{T}$	Set of pickruns.

Parameters

M	Sufficiently large positive number.
$\tau^K_{i,i'}$	Travel time from location of item i to location of item i' by human pickers.
$\tau^R_{i,i'}$	Travel time from location of item i to location of item i' by AMRs.
$\tau^{o,R}_{r,i}$	Travel time from starting location of AMR r to location i.
$\tau^{o,K}_{k,i}$	Travel time from starting location of picker k to location i.
η^U_i	Time to retrieve item i from its storage location, which can already be done before AMR is there. Note, this does not apply to our use case.
η^L_i	Time to place item i on an AMR.
$a^T_{i,t}$	1 if item i must be picked in pickrun t, 0 otherwise.
$u^R_{i,i'}$	1 if AMR r collects item i before item i' in the same trip but not necessarily exactly before item i (i.e. relative precedence), 0 otherwise.
$l_{i,t}$	1 if item i is the last item to be picked in pickrun t, 0 otherwise.
$u^T_{t,t'}$	1 if tour t occurs before tour t'.

Variables

Decision Variables

$A^K_{i,k}$	1 if human picker k is assigned to pick item i, 0 otherwise.
$U^K_{i,i'}$	1 if item i must be retrieved before item i' by the same picker, 0 otherwise.

Other Variables

$A^R_{i,r,t}$	1 if AMR r is assigned to transport item i in tour t, 0 otherwise.
$A^R_{r,t}$	1 if AMR r is assigned to perform pickrun t.
$B^K_{i,k}$	Retrieval begin time of item i by picker k.
$F^K_{i,k}$	Retrieval finish time of item i by picker k.

$L^K_{i,k}$	Time at which picker k is ready to leave the location of item i.
$F^R_{i,r,t}$	Collection finish time of item i by AMR r in pickrun t.
$B^R_{i,r,t}$	Collection begin time of item i by AMR r in pickrun t.
$C^R_{r,t}$	Completion time of pickrun t by AMR r.
S_t	Start time of pickrun t.
C_t	Completion time of pickrun t.
C	Delivery completion time of the last fulfilled order.
W_k	The total workload of picker k.

$$\min C \tag{A.1}$$

$$\max F(W_1, W_2, \dots, W_{|\mathcal{K}|}) \tag{A.2}$$

subject to

$$\sum_{k \in \mathcal{K}} A^K_{i,k} = 1 \qquad \forall i \in \mathcal{N} \tag{A.3}$$

$$A^K_{i,k} - A^K_{i',k} \leq 1 - (U^K_{i,i'} + U^K_{i',i}) \qquad \forall i, i' \in \mathcal{N}, i \neq i', k \in \mathcal{K} \tag{A.4}$$

$$A^K_{i,k} + A^K_{i',k} \leq 1 + (U^K_{i,i'} + U^K_{i',i}) \qquad \forall i, i' \in \mathcal{N}, i \neq i', k \in \mathcal{K} \tag{A.5}$$

$$B^K_{i.k} \geq \tau^{o,K}_{k,i} - M \cdot (1 - A^K_{i,k}) \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{A.6}$$

$$\sum_{k \in \mathcal{K}} B^K_{i.k} \geq \sum_{k \in \mathcal{K}} L^K_{i',k} + \tau^K_{i',i} \cdot U^K_{i',i} - M \cdot (1 - U^K_{i',i}) \qquad \forall i, i' \in \mathcal{N}, i \neq i' \tag{A.7}$$

$$F^K_{i,k} = B^K_{i,k} + \eta^U_i \cdot A^K_{i,k} \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{A.8}$$

$$L^K_{i,k} \geq F^K_{i,k} \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{A.9}$$

$$L^K_{i,k} \geq \sum_{t \in \mathcal{T}} F^R_{i,r,t} - M \cdot \left(2 - A^K_{i,k} - \sum_{t \in \mathcal{T}} A^R_{i,r,t}\right) \qquad \forall i \in \mathcal{N}, k \in \mathcal{K}, r \in \mathcal{R} \tag{A.10}$$

$$B^K_{i,k} \leq M \cdot A^K_{i,k} \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{A.11}$$

$$F_{i,k} \leq M \cdot A^K_{i,k} \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{A.12}$$

$$L^K_{i,k} \leq M \cdot A^K_{i,k} \qquad \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{A.13}$$

$$\begin{aligned} B^R_{i,r,t} \geq S_t &+ \sum_{i' \in \mathcal{N}} \left(\tau^R_{i',i} \cdot l_{i',t'} \cdot u^T_{t',t}\right) \\ &- M \cdot \left(2 - A^T_{i,r,t} - A^T_{r,t'}\right) \end{aligned} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, t, t' \in \mathcal{T}, t \neq t' \tag{A.14}$$

$$\sum_{r \in \mathcal{R}} B^R_{i,r,t} \geq \left(\sum_{r \in \mathcal{R}} F^R_{i',r,t} + \tau^R_{i',i}\right) \cdot u^T_{i',i} \cdot a^T_{i,t} \cdot a^T_{i',t} \qquad \forall i, i' \in \mathcal{N}, i \neq i', t \in \mathcal{T} \tag{A.15}$$

$$B^R_{i,r,t} \geq \tau^{o,R}_{r,i} - M \cdot (1 - A^R_{i,r,t}) \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \tag{A.16}$$

$$B^R_{i,r,t} \geq F^K_{i,k} - M \cdot (2 - A^K_{i,k} - A^R_{i,r,t}) \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T}, k \in \mathcal{K} \tag{A.17}$$

$$F^R_{i,r,t} = B^R_{i,r,t} + \eta^L_i \cdot A^R_{i,r,t} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \tag{A.18}$$

$$B^R_{i,r,t} \leq M \cdot A^R_{i,r,t} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \tag{A.19}$$

$$F^R_{i,r,t} \leq M \cdot A^R_{i,r,t} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \tag{A.20}$$

$$C_t \geq F^R_{i,r,t} - M \cdot (1 - A^R_{i,r,t}) \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \tag{A.21}$$

$$S_t \geq C_{t'} \cdot u^T_{t',t} - M \cdot (2 - A^T_{r,t} - A^T_{r,t'}) \qquad \forall r \in \mathcal{R}, t, t' \in \mathcal{T}, t \neq t' \tag{A.22}$$

$$C \geq C_t \qquad \forall t \in \mathcal{T} \tag{A.23}$$

$$W_k = \sum_{i \in \mathcal{N}} w_i \cdot A_{i,k} \qquad \forall k \in \mathcal{K} \tag{A.24}$$

$$\sum_{r \in \mathcal{R}} A^T_{r,t} = 1 \qquad \forall t \in \mathcal{T} \tag{A.25}$$

$$A^R_{i,r,t} = A^T_{r,t} \cdot a^T_{i,t} \qquad \forall i \in \mathcal{N}, r \in \mathcal{R}, t \in \mathcal{T} \tag{A.26}$$

$$A^K_{i,k}, U^K_{i,i'}, A^R_{i,r,t}, A^T_{r,t} \in \{0, 1\} \qquad \forall i, i' \in \mathcal{N}, k \in \mathcal{K}, r \in \mathcal{R}, t \in \mathcal{T} \tag{A.27}$$

$$B^K_{i,k}, F^K_{i,k}, L^K_{i,k}, F^R_{i,r,t}, B^R_{i,r,t}, C_t, S_t, C \geq 0 \qquad \begin{aligned} &\forall i \in \mathcal{N}, k \in \mathcal{K}, r \in \mathcal{R}, \\ &t \in \mathcal{T} \end{aligned} \tag{A.28}$$

Appendix B

Simulation Data Distributions

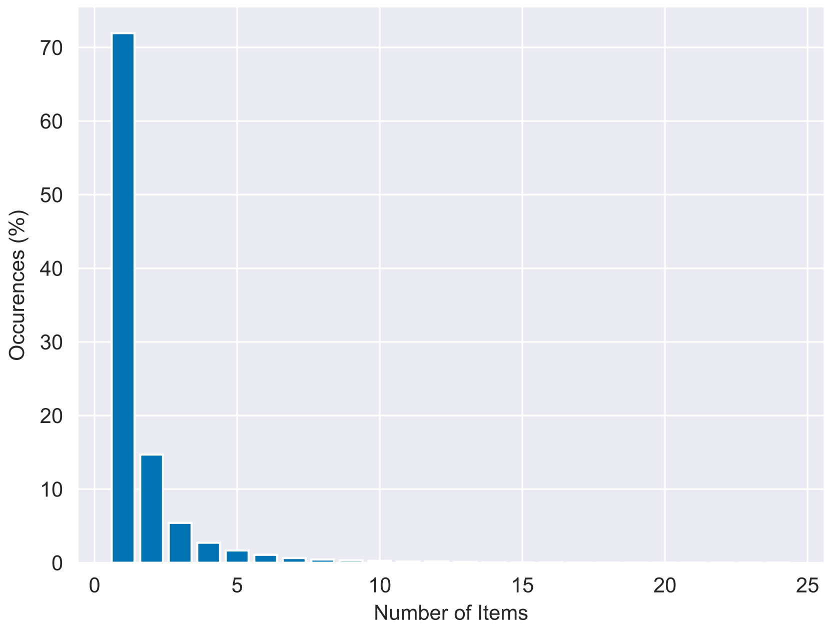


Figure B.1: Distribution of the occurrences of pick frequencies used in the simulation model.

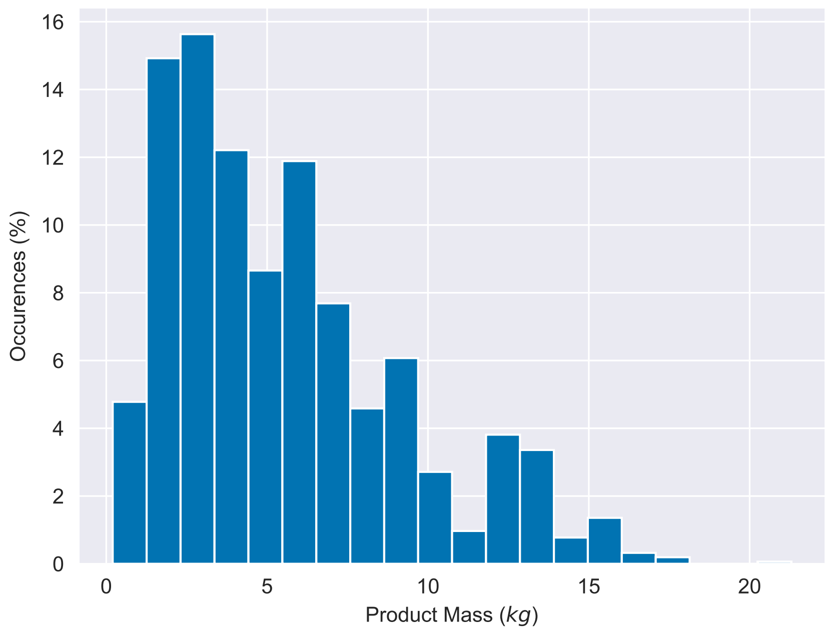


Figure B.2: Distribution of product masses.

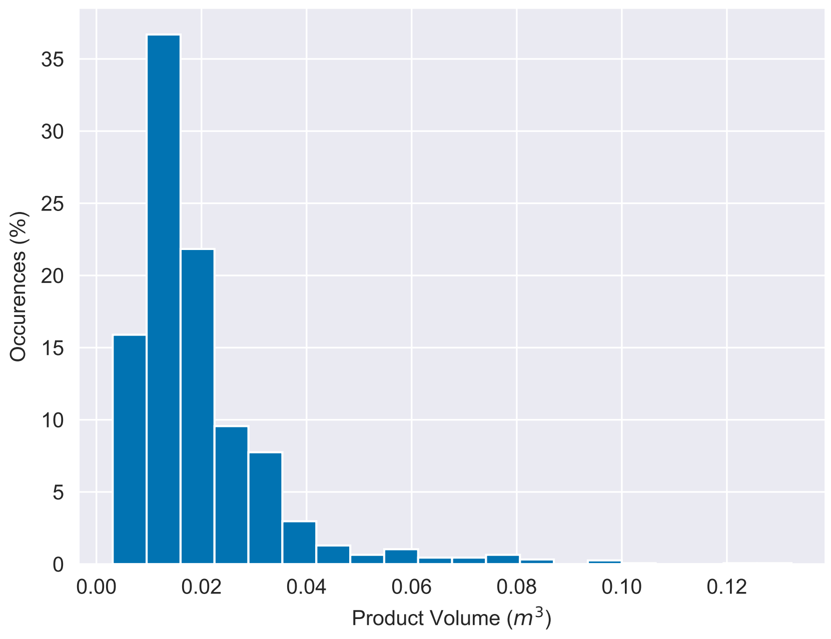


Figure B.3: Distribution of product volumes.

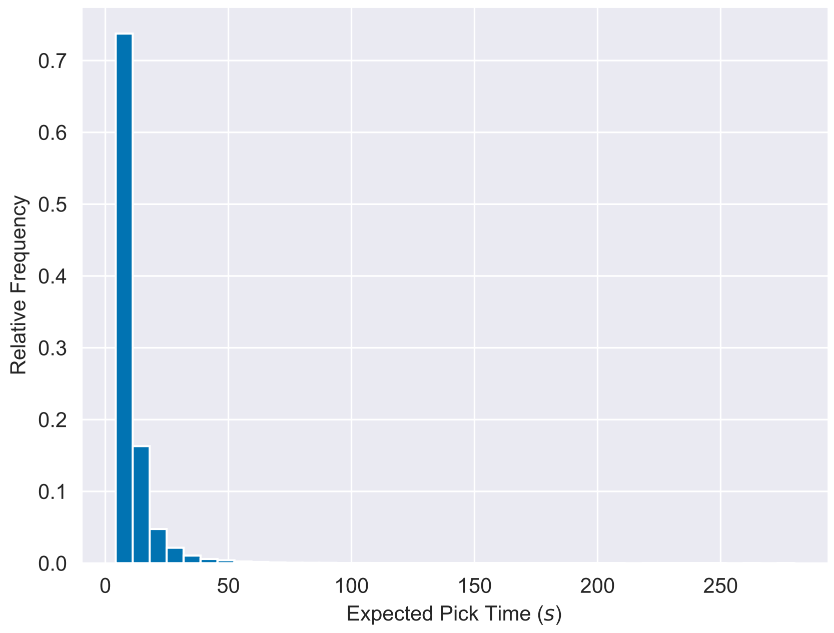


Figure B.4: Histogram of the expected pick times of 100,000 sampled combinations of products and number of picked items.

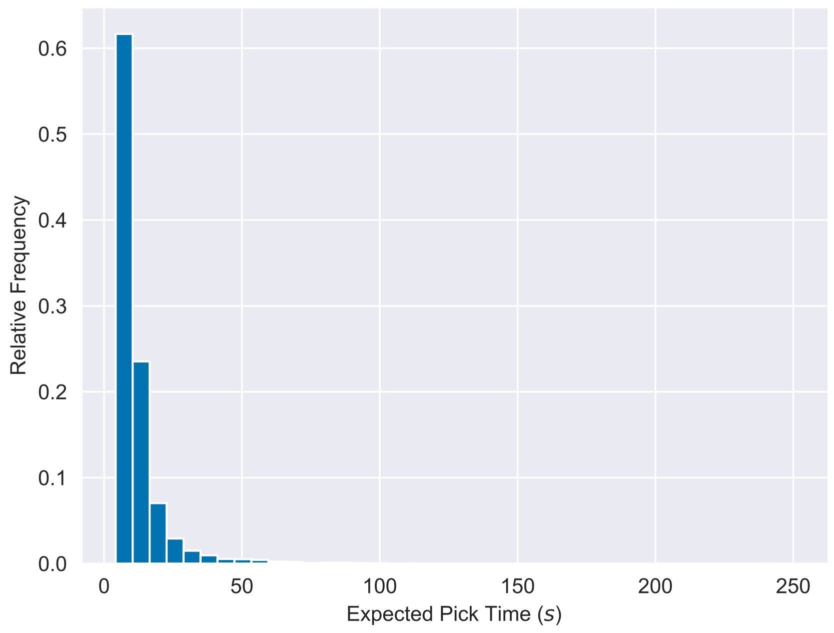


Figure B.5: Histogram of the expected pick times of combinations of products and number of picked items from the real order data.

Appendix C

Snapshot of Decision Tree Data

Table C.1 outlines a snapshot of five rows of the data used to train the decision tree for the pure performance policy. The numbered columns relate to the following features.

1. Current picker location
2. AMR location indicator
3. Distance from picker
4. AMR destination distance
5. Expected time until next AMR destination
6. Expected time until two-step ahead AMR destination
7. Picker location indicator
8. Picker destination distance
9. Number of pickers to aisle
10. Number of AMRs to aisle
11. Minimum distance of other pickers
12. Node depth within aisle
13. Aisle distance from origin
14. Minimum distance next AMR destination
15. Second smallest distance next AMR destination
16. Minimum distance two-step ahead AMR destination
17. Second smallest distance two-step ahead AMR destination
18. Number of waiting AMRs in aisle
19. Number of AMRs to aisle
20. Distance of closest other picker destination
21. Distance of closest unserved AMR destination
22. Distance of second closest unserved AMR destination
23. Minimum expected time of other pickers

Table C.1: Snapshot of 5 rows of the dataset used to train the decision tree for the pure performance policy. Columns 1-23 indicate the node features and y the value assigned to the node by the pure performance policy.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	y
0.0	1.0	16.8	0.0	-10.0	-10.0	0.0	-10.0	2.0	9.0	5.2	0.9	0.7	0.0	0.0	0.0	0.0	6.0	1.0	5.2	0.0	0.0	7.0	8.4
0.0	0.0	19.2	11.2	-10.0	-10.0	0.0	-10.0	1.0	3.0	8.4	0.2	0.9	4.2	11.2	6.6	0.0	1.0	2.0	7.0	0.0	0.0	10.9	6.1
0.0	0.0	71.4	8.4	-10.0	-10.0	0.0	-10.0	3.0	16.0	1.4	0.6	0.0	16.8	19.6	21.0	0.0	5.0	2.0	1.4	0.0	0.0	6.8	-0.3
0.0	1.0	29.0	0.0	-10.0	-10.0	0.0	-10.0	2.0	5.0	2.4	0.1	0.4	2.4	0.0	8.0	0.0	4.0	1.0	2.4	0.0	0.0	7.6	1.3
0.0	1.0	9.4	0.0	-10.0	-10.0	0.0	-10.0	3.0	10.0	5.6	0.2	0.1	8.4	0.0	22.4	0.0	4.0	1.0	1.4	0.0	0.0	11.3	18.0

Appendix D

Grid Search Results for Decision Tree Analysis

Table D.1: Grid search results for the decision tree analysis of policy 1. R^2 is the average R^2 evaluation value over the five cross-validation folds.

Max. Depth	Max. Nr. of Features	Min. Samples Leaf Nodes	R^2
4	sqrt	5	0.31
4	sqrt	10	0.35
4	sqrt	25	0.32
4	sqrt	50	0.36
4	sqrt	100	0.32
4	sqrt	250	0.35
4	log2	5	0.35
4	log2	10	0.35
4	log2	25	0.35
4	log2	50	0.38
4	log2	100	0.31
4	log2	250	0.34
4	1	5	0.56
4	1	10	0.56
4	1	25	0.56
4	1	50	0.56
4	1	100	0.56
4	1	250	0.56
7	sqrt	5	0.49
7	sqrt	10	0.46
7	sqrt	25	0.49
7	sqrt	50	0.51
7	sqrt	100	0.43
7	sqrt	250	0.42
7	log2	5	0.49
7	log2	10	0.47
7	log2	25	0.41
7	log2	50	0.47
7	log2	100	0.48
7	log2	250	0.40

7	1	5	0.71
7	1	10	0.71
7	1	25	0.71
7	1	50	0.71
7	1	100	0.71
7	1	250	0.71
10	sqrt	5	0.55
10	sqrt	10	0.58
10	sqrt	25	0.55
10	sqrt	50	0.58
10	sqrt	100	0.57
10	sqrt	250	0.52
10	log2	5	0.59
10	log2	10	0.59
10	log2	25	0.59
10	log2	50	0.60
10	log2	100	0.59
10	log2	250	0.58
10	1	5	0.80
10	1	10	0.80
10	1	25	0.80
10	1	50	0.79
10	1	100	0.79
10	1	250	0.78
15	sqrt	5	0.67
15	sqrt	10	0.68
15	sqrt	25	0.71
15	sqrt	50	0.66
15	sqrt	100	0.65
15	sqrt	250	0.62
15	log2	5	0.64
15	log2	10	0.68
15	log2	25	0.70
15	log2	50	0.68
15	log2	100	0.63
15	log2	250	0.63
15	1	5	0.84
15	1	10	0.84
15	1	25	0.84
15	1	50	0.83
15	1	100	0.82
15	1	250	0.80
25	sqrt	5	0.68
25	sqrt	10	0.72
25	sqrt	25	0.70
25	sqrt	50	0.67
25	sqrt	100	0.66
25	sqrt	250	0.63
25	log2	5	0.70
25	log2	10	0.70
25	log2	25	0.71

25	log2	50	0.68
25	log2	100	0.65
25	log2	250	0.65
25	1	5	0.82
25	1	10	0.84
25	1	25	0.84
25	1	50	0.83
25	1	100	0.82
25	1	250	0.80
40	sqrt	5	0.69
40	sqrt	10	0.70
40	sqrt	25	0.72
40	sqrt	50	0.68
40	sqrt	100	0.68
40	sqrt	250	0.60
40	log2	5	0.70
40	log2	10	0.69
40	log2	25	0.72
40	log2	50	0.70
40	log2	100	0.68
40	log2	250	0.63
40	1	5	0.82
40	1	10	0.83
40	1	25	0.84
40	1	50	0.83
40	1	100	0.82
40	1	250	0.80
50	sqrt	5	0.68
50	sqrt	10	0.70
50	sqrt	25	0.70
50	sqrt	50	0.70
50	sqrt	100	0.65
50	sqrt	250	0.64
50	log2	5	0.69
50	log2	10	0.71
50	log2	25	0.71
50	log2	50	0.68
50	log2	100	0.65
50	log2	250	0.63
50	1	5	0.82
50	1	10	0.83
50	1	25	0.84
50	1	50	0.83
50	1	100	0.82
50	1	250	0.80

Table D.2: Grid search results for the decision tree analysis of policy 2. R^2 is the average R^2 evaluation value over the five cross-validation folds.

Max. Depth	Max. Nr. of Features	Min. Samples Leaf Nodes	R^2
4	sqrt	5	0.47
4	sqrt	10	0.42
4	sqrt	25	0.40
4	sqrt	50	0.38
4	sqrt	100	0.51
4	sqrt	250	0.48
4	log2	5	0.43
4	log2	10	0.40
4	log2	25	0.40
4	log2	50	0.36
4	log2	100	0.43
4	log2	250	0.50
4	1	5	0.69
4	1	10	0.69
4	1	25	0.69
4	1	50	0.69
4	1	100	0.69
4	1	250	0.69
7	sqrt	5	0.59
7	sqrt	10	0.53
7	sqrt	25	0.53
7	sqrt	50	0.48
7	sqrt	100	0.54
7	sqrt	250	0.58
7	log2	5	0.53
7	log2	10	0.44
7	log2	25	0.55
7	log2	50	0.54
7	log2	100	0.55
7	log2	250	0.49
7	1	5	0.78
7	1	10	0.78
7	1	25	0.78
7	1	50	0.78
7	1	100	0.78
7	1	250	0.78
10	sqrt	5	0.59
10	sqrt	10	0.64
10	sqrt	25	0.62
10	sqrt	50	0.63
10	sqrt	100	0.63
10	sqrt	250	0.60
10	log2	5	0.61
10	log2	10	0.63
10	log2	25	0.64
10	log2	50	0.65
10	log2	100	0.68

10	log2	250	0.62
10	1	5	0.83
10	1	10	0.83
10	1	25	0.83
10	1	50	0.83
10	1	100	0.82
10	1	250	0.81
15	sqrt	5	0.69
15	sqrt	10	0.72
15	sqrt	25	0.70
15	sqrt	50	0.71
15	sqrt	100	0.67
15	sqrt	250	0.67
15	log2	5	0.72
15	log2	10	0.72
15	log2	25	0.75
15	log2	50	0.71
15	log2	100	0.71
15	log2	250	0.70
15	1	5	0.86
15	1	10	0.86
15	1	25	0.86
15	1	50	0.86
15	1	100	0.85
15	1	250	0.83
25	sqrt	5	0.74
25	sqrt	10	0.76
25	sqrt	25	0.75
25	sqrt	50	0.76
25	sqrt	100	0.72
25	sqrt	250	0.70
25	log2	5	0.74
25	log2	10	0.76
25	log2	25	0.75
25	log2	50	0.75
25	log2	100	0.74
25	log2	250	0.70
25	1	5	0.85
25	1	10	0.86
25	1	25	0.87
25	1	50	0.86
25	1	100	0.85
25	1	250	0.83
40	sqrt	5	0.78
40	sqrt	10	0.78
40	sqrt	25	0.75
40	sqrt	50	0.73
40	sqrt	100	0.71
40	sqrt	250	0.70
40	log2	5	0.75
40	log2	10	0.76

40	log2	25	0.77
40	log2	50	0.77
40	log2	100	0.75
40	log2	250	0.68
40	1	5	0.85
40	1	10	0.86
40	1	25	0.87
40	1	50	0.86
40	1	100	0.85
40	1	250	0.83
50	sqrt	5	0.76
50	sqrt	10	0.77
50	sqrt	25	0.77
50	sqrt	50	0.74
50	sqrt	100	0.74
50	sqrt	250	0.67
50	log2	5	0.74
50	log2	10	0.76
50	log2	25	0.73
50	log2	50	0.76
50	log2	100	0.74
50	log2	250	0.68
50	1	5	0.85
50	1	10	0.86
50	1	25	0.87
50	1	50	0.86
50	1	100	0.85
50	1	250	0.83

Table D.3: Grid search results for the decision tree analysis of policy 3. R^2 is the average R^2 evaluation value over the five cross-validation folds.

Max. Depth	Max. Nr. of Features	Min. Samples Leaf Nodes	R^2
4	sqrt	5	0.30
4	sqrt	10	0.37
4	sqrt	25	0.50
4	sqrt	50	0.38
4	sqrt	100	0.32
4	sqrt	250	0.34
4	log2	5	0.55
4	log2	10	0.47
4	log2	25	0.39
4	log2	50	0.37
4	log2	100	0.41
4	log2	250	0.38
4	1	5	0.71
4	1	10	0.71
4	1	25	0.71
4	1	50	0.70
4	1	100	0.70
4	1	250	0.70
7	sqrt	5	0.55
7	sqrt	10	0.50
7	sqrt	25	0.52
7	sqrt	50	0.55
7	sqrt	100	0.57
7	sqrt	250	0.55
7	log2	5	0.50
7	log2	10	0.57
7	log2	25	0.66
7	log2	50	0.46
7	log2	100	0.46
7	log2	250	0.49
7	1	5	0.79
7	1	10	0.79
7	1	25	0.79
7	1	50	0.79
7	1	100	0.79
7	1	250	0.79
10	sqrt	5	0.67
10	sqrt	10	0.66
10	sqrt	25	0.60
10	sqrt	50	0.67
10	sqrt	100	0.68
10	sqrt	250	0.62
10	log2	5	0.59
10	log2	10	0.64
10	log2	25	0.65
10	log2	50	0.62
10	log2	100	0.68

10	log2	250	0.59
10	1	5	0.82
10	1	10	0.82
10	1	25	0.82
10	1	50	0.82
10	1	100	0.82
10	1	250	0.82
15	sqrt	5	0.77
15	sqrt	10	0.71
15	sqrt	25	0.71
15	sqrt	50	0.72
15	sqrt	100	0.67
15	sqrt	250	0.69
15	log2	5	0.74
15	log2	10	0.69
15	log2	25	0.71
15	log2	50	0.67
15	log2	100	0.71
15	log2	250	0.73
15	1	5	0.82
15	1	10	0.83
15	1	25	0.84
15	1	50	0.84
15	1	100	0.84
15	1	250	0.83
25	sqrt	5	0.72
25	sqrt	10	0.77
25	sqrt	25	0.71
25	sqrt	50	0.70
25	sqrt	100	0.73
25	sqrt	250	0.72
25	log2	5	0.73
25	log2	10	0.71
25	log2	25	0.75
25	log2	50	0.69
25	log2	100	0.71
25	log2	250	0.70
25	1	5	0.81
25	1	10	0.83
25	1	25	0.84
25	1	50	0.84
25	1	100	0.84
25	1	250	0.83
40	sqrt	5	0.72
40	sqrt	10	0.74
40	sqrt	25	0.76
40	sqrt	50	0.74
40	sqrt	100	0.70
40	sqrt	250	0.71
40	log2	5	0.74
40	log2	10	0.74

40	log2	25	0.75
40	log2	50	0.75
40	log2	100	0.72
40	log2	250	0.67
40	1	5	0.81
40	1	10	0.83
40	1	25	0.84
40	1	50	0.84
40	1	100	0.84
40	1	250	0.83
50	sqrt	5	0.74
50	sqrt	10	0.76
50	sqrt	25	0.74
50	sqrt	50	0.72
50	sqrt	100	0.73
50	sqrt	250	0.67
50	log2	5	0.72
50	log2	10	0.73
50	log2	25	0.72
50	log2	50	0.73
50	log2	100	0.73
50	log2	250	0.67
50	1	5	0.81
50	1	10	0.83
50	1	25	0.84
50	1	50	0.84
50	1	100	0.84
50	1	250	0.83

Table D.4: Grid search results for the decision tree analysis of policy 4. R^2 is the average R^2 evaluation value over the five cross-validation folds.

Max. Depth	Max. Nr. of Features	Min. Samples Leaf Nodes	R^2
4	sqrt	5	0.44
4	sqrt	10	0.43
4	sqrt	25	0.36
4	sqrt	50	0.34
4	sqrt	100	0.47
4	sqrt	250	0.40
4	log2	5	0.48
4	log2	10	0.37
4	log2	25	0.37
4	log2	50	0.33
4	log2	100	0.37
4	log2	250	0.38
4	1	5	0.72
4	1	10	0.72
4	1	25	0.72
4	1	50	0.72
4	1	100	0.72
4	1	250	0.72
7	sqrt	5	0.58
7	sqrt	10	0.58
7	sqrt	25	0.58
7	sqrt	50	0.57
7	sqrt	100	0.46
7	sqrt	250	0.51
7	log2	5	0.56
7	log2	10	0.44
7	log2	25	0.57
7	log2	50	0.60
7	log2	100	0.56
7	log2	250	0.45
7	1	5	0.81
7	1	10	0.81
7	1	25	0.81
7	1	50	0.81
7	1	100	0.81
7	1	250	0.81
10	sqrt	5	0.64
10	sqrt	10	0.68
10	sqrt	25	0.68
10	sqrt	50	0.63
10	sqrt	100	0.64
10	sqrt	250	0.67
10	log2	5	0.64
10	log2	10	0.62
10	log2	25	0.67
10	log2	50	0.65
10	log2	100	0.64

10	log2	250	0.65
10	1	5	0.85
10	1	10	0.85
10	1	25	0.85
10	1	50	0.85
10	1	100	0.85
10	1	250	0.84
15	sqrt	5	0.74
15	sqrt	10	0.76
15	sqrt	25	0.72
15	sqrt	50	0.70
15	sqrt	100	0.73
15	sqrt	250	0.72
15	log2	5	0.71
15	log2	10	0.74
15	log2	25	0.73
15	log2	50	0.70
15	log2	100	0.72
15	log2	250	0.70
15	1	5	0.86
15	1	10	0.86
15	1	25	0.87
15	1	50	0.87
15	1	100	0.86
15	1	250	0.85
25	sqrt	5	0.77
25	sqrt	10	0.77
25	sqrt	25	0.79
25	sqrt	50	0.71
25	sqrt	100	0.74
25	sqrt	250	0.73
25	log2	5	0.76
25	log2	10	0.74
25	log2	25	0.75
25	log2	50	0.73
25	log2	100	0.75
25	log2	250	0.68
25	1	5	0.85
25	1	10	0.86
25	1	25	0.87
25	1	50	0.87
25	1	100	0.86
25	1	250	0.85
40	sqrt	5	0.75
40	sqrt	10	0.78
40	sqrt	25	0.76
40	sqrt	50	0.78
40	sqrt	100	0.74
40	sqrt	250	0.71
40	log2	5	0.76
40	log2	10	0.74

40	log2	25	0.77
40	log2	50	0.76
40	log2	100	0.69
40	log2	250	0.74
40	1	5	0.85
40	1	10	0.86
40	1	25	0.87
40	1	50	0.87
40	1	100	0.86
40	1	250	0.85
50	sqrt	5	0.75
50	sqrt	10	0.77
50	sqrt	25	0.79
50	sqrt	50	0.75
50	sqrt	100	0.72
50	sqrt	250	0.68
50	log2	5	0.77
50	log2	10	0.75
50	log2	25	0.79
50	log2	50	0.76
50	log2	100	0.75
50	log2	250	0.70
50	1	5	0.85
50	1	10	0.86
50	1	25	0.87
50	1	50	0.87
50	1	100	0.86
50	1	250	0.85

Table D.5: Grid search results for the decision tree analysis of policy 5. R^2 is the average R^2 evaluation value over the five cross-validation folds.

Max. Depth	Max. Nr. of Features	Min. Samples Leaf Nodes	R^2
4	5	sqrt	0.46
4	10	sqrt	0.38
4	25	sqrt	0.38
4	50	sqrt	0.38
4	100	sqrt	0.36
4	250	sqrt	0.41
4	5	log2	0.42
4	10	log2	0.33
4	25	log2	0.38
4	50	log2	0.32
4	100	log2	0.42
4	250	log2	0.45
4	5	1	0.68
4	10	1	0.68
4	25	1	0.68
4	50	1	0.68
4	100	1	0.68
4	250	1	0.68
7	5	sqrt	0.51
7	10	sqrt	0.57
7	25	sqrt	0.58
7	50	sqrt	0.53
7	100	sqrt	0.51
7	250	sqrt	0.59
7	5	log2	0.58
7	10	log2	0.49
7	25	log2	0.58
7	50	log2	0.61
7	100	log2	0.62
7	250	log2	0.53
7	5	1	0.79
7	10	1	0.79
7	25	1	0.79
7	50	1	0.79
7	100	1	0.79
7	250	1	0.79
10	5	sqrt	0.59
10	10	sqrt	0.69
10	25	sqrt	0.64
10	50	sqrt	0.70
10	100	sqrt	0.67
10	250	sqrt	0.62
10	5	log2	0.69
10	10	log2	0.67
10	25	log2	0.65
10	50	log2	0.66
10	100	log2	0.65

10	250	log2	0.60
10	5	1	0.84
10	10	1	0.84
10	25	1	0.84
10	50	1	0.84
10	100	1	0.83
10	250	1	0.83
15	5	sqrt	0.76
15	10	sqrt	0.76
15	25	sqrt	0.77
15	50	sqrt	0.73
15	100	sqrt	0.72
15	250	sqrt	0.70
15	5	log2	0.77
15	10	log2	0.75
15	25	log2	0.73
15	50	log2	0.69
15	100	log2	0.70
15	250	log2	0.69
15	5	1	0.86
15	10	1	0.87
15	25	1	0.87
15	50	1	0.87
15	100	1	0.86
15	250	1	0.84
25	5	sqrt	0.80
25	10	sqrt	0.77
25	25	sqrt	0.80
25	50	sqrt	0.75
25	100	sqrt	0.69
25	250	sqrt	0.68
25	5	log2	0.78
25	10	log2	0.78
25	25	log2	0.75
25	50	log2	0.72
25	100	log2	0.73
25	250	log2	0.72
25	5	1	0.86
25	10	1	0.87
25	25	1	0.87
25	50	1	0.87
25	100	1	0.86
25	250	1	0.84
40	5	sqrt	0.78
40	10	sqrt	0.79
40	25	sqrt	0.77
40	50	sqrt	0.76
40	100	sqrt	0.72
40	250	sqrt	0.67
40	5	log2	0.76
40	10	log2	0.77

40	25	log2	0.77
40	50	log2	0.78
40	100	log2	0.72
40	250	log2	0.70
40	5	1	0.86
40	10	1	0.87
40	25	1	0.87
40	50	1	0.87
40	100	1	0.86
40	250	1	0.84
50	5	sqrt	0.79
50	10	sqrt	0.76
50	25	sqrt	0.79
50	50	sqrt	0.78
50	100	sqrt	0.73
50	250	sqrt	0.68
50	5	log2	0.78
50	10	log2	0.78
50	25	log2	0.74
50	50	log2	0.78
50	100	log2	0.74
50	250	log2	0.72
50	5	1	0.86
50	10	1	0.87
50	25	1	0.87
50	50	1	0.87
50	100	1	0.86
50	250	1	0.84

Table D.6: Grid search results for the decision tree analysis of policy 6. R^2 is the average R^2 evaluation value over the five cross-validation folds.

Max. Depth	Max. Nr. of Features	Min. Samples Leaf Nodes	R^2
4	5	sqrt	0.38
4	10	sqrt	0.32
4	25	sqrt	0.30
4	50	sqrt	0.33
4	100	sqrt	0.38
4	250	sqrt	0.40
4	5	log2	0.37
4	10	log2	0.35
4	25	log2	0.38
4	50	log2	0.43
4	100	log2	0.41
4	250	log2	0.38
4	5	1	0.69
4	10	1	0.69
4	25	1	0.69
4	50	1	0.69
4	100	1	0.69
4	250	1	0.69
7	5	sqrt	0.62
7	10	sqrt	0.63
7	25	sqrt	0.50
7	50	sqrt	0.58
7	100	sqrt	0.50
7	250	sqrt	0.61
7	5	log2	0.58
7	10	log2	0.57
7	25	log2	0.51
7	50	log2	0.56
7	100	log2	0.55
7	250	log2	0.61
7	5	1	0.82
7	10	1	0.82
7	25	1	0.82
7	50	1	0.82
7	100	1	0.82
7	250	1	0.81
10	5	sqrt	0.65
10	10	sqrt	0.66
10	25	sqrt	0.67
10	50	sqrt	0.65
10	100	sqrt	0.69
10	250	sqrt	0.63
10	5	log2	0.70
10	10	log2	0.67
10	25	log2	0.71
10	50	log2	0.69
10	100	log2	0.67

10	250	log2	0.63
10	5	1	0.87
10	10	1	0.87
10	25	1	0.87
10	50	1	0.87
10	100	1	0.87
10	250	1	0.86
15	5	sqrt	0.80
15	10	sqrt	0.79
15	25	sqrt	0.77
15	50	sqrt	0.77
15	100	sqrt	0.76
15	250	sqrt	0.70
15	5	log2	0.76
15	10	log2	0.78
15	25	log2	0.80
15	50	log2	0.78
15	100	log2	0.74
15	250	log2	0.73
15	5	1	0.91
15	10	1	0.91
15	25	1	0.91
15	50	1	0.90
15	100	1	0.89
15	250	1	0.88
25	5	sqrt	0.81
25	10	sqrt	0.78
25	25	sqrt	0.81
25	50	sqrt	0.76
25	100	sqrt	0.74
25	250	sqrt	0.70
25	5	log2	0.83
25	10	log2	0.84
25	25	log2	0.81
25	50	log2	0.74
25	100	log2	0.76
25	250	log2	0.72
25	5	1	0.91
25	10	1	0.92
25	25	1	0.92
25	50	1	0.91
25	100	1	0.90
25	250	1	0.88
40	5	sqrt	0.83
40	10	sqrt	0.81
40	25	sqrt	0.81
40	50	sqrt	0.80
40	100	sqrt	0.79
40	250	sqrt	0.67
40	5	log2	0.82
40	10	log2	0.83

40	25	log2	0.80
40	50	log2	0.77
40	100	log2	0.80
40	250	log2	0.73
40	5	1	0.91
40	10	1	0.92
40	25	1	0.92
40	50	1	0.91
40	100	1	0.90
40	250	1	0.88
50	5	sqrt	0.82
50	10	sqrt	0.81
50	25	sqrt	0.84
50	50	sqrt	0.77
50	100	sqrt	0.75
50	250	sqrt	0.72
50	5	log2	0.82
50	10	log2	0.82
50	25	log2	0.81
50	50	log2	0.82
50	100	log2	0.81
50	250	log2	0.73
50	5	1	0.91
50	10	1	0.92
50	25	1	0.92
50	50	1	0.91
50	100	1	0.90
50	250	1	0.88

Table D.7: Grid search results for the decision tree analysis of the pure performance policy. R^2 is the average R^2 evaluation value over the five cross-validation folds.

Max. Depth	Max. Nr. of Features	Min. Samples Leaf Nodes	R^2
4	5	sqrt	0.49
4	10	sqrt	0.35
4	25	sqrt	0.40
4	50	sqrt	0.44
4	100	sqrt	0.49
4	250	sqrt	0.49
4	5	log2	0.49
4	10	log2	0.42
4	25	log2	0.44
4	50	log2	0.50
4	100	log2	0.34
4	250	log2	0.48
4	5	1	0.76
4	10	1	0.76
4	25	1	0.76
4	50	1	0.76
4	100	1	0.76
4	250	1	0.76
7	5	sqrt	0.63
7	10	sqrt	0.65
7	25	sqrt	0.67
7	50	sqrt	0.68
7	100	sqrt	0.65
7	250	sqrt	0.67
7	5	log2	0.65
7	10	log2	0.61
7	25	log2	0.63
7	50	log2	0.68
7	100	log2	0.65
7	250	log2	0.67
7	5	1	0.88
7	10	1	0.88
7	25	1	0.88
7	50	1	0.88
7	100	1	0.88
7	250	1	0.88
10	5	sqrt	0.76
10	10	sqrt	0.79
10	25	sqrt	0.75
10	50	sqrt	0.76
10	100	sqrt	0.79
10	250	sqrt	0.74
10	5	log2	0.72
10	10	log2	0.77
10	25	log2	0.79
10	50	log2	0.79
10	100	log2	0.75

10	250	log2	0.77
10	5	1	0.93
10	10	1	0.93
10	25	1	0.93
10	50	1	0.93
10	100	1	0.93
10	250	1	0.92
15	5	sqrt	0.88
15	10	sqrt	0.85
15	25	sqrt	0.84
15	50	sqrt	0.83
15	100	sqrt	0.83
15	250	sqrt	0.75
15	5	log2	0.85
15	10	log2	0.85
15	25	log2	0.82
15	50	log2	0.83
15	100	log2	0.78
15	250	log2	0.77
15	5	1	0.96
15	10	1	0.96
15	25	1	0.95
15	50	1	0.95
15	100	1	0.94
15	250	1	0.93
25	5	sqrt	0.88
25	10	sqrt	0.89
25	25	sqrt	0.86
25	50	sqrt	0.66
25	100	sqrt	0.83
25	250	sqrt	0.77
25	5	log2	0.88
25	10	log2	0.88
25	25	log2	0.87
25	50	log2	0.85
25	100	log2	0.78
25	250	log2	0.71
25	5	1	0.96
25	10	1	0.96
25	25	1	0.96
25	50	1	0.95
25	100	1	0.94
25	250	1	0.93
40	5	sqrt	0.89
40	10	sqrt	0.85
40	25	sqrt	0.88
40	50	sqrt	0.81
40	100	sqrt	0.81
40	250	sqrt	0.73
40	5	log2	0.87
40	10	log2	0.86

40	25	log2	0.87
40	50	log2	0.86
40	100	log2	0.80
40	250	log2	0.78
40	5	1	0.96
40	10	1	0.96
40	25	1	0.96
40	50	1	0.95
40	100	1	0.94
40	250	1	0.93
50	5	sqrt	0.87
50	10	sqrt	0.87
50	25	sqrt	0.85
50	50	sqrt	0.87
50	100	sqrt	0.85
50	250	sqrt	0.78
50	5	log2	0.89
50	10	log2	0.88
50	25	log2	0.84
50	50	log2	0.84
50	100	log2	0.83
50	250	log2	0.77
50	5	1	0.96
50	10	1	0.96
50	25	1	0.96
50	50	1	0.95
50	100	1	0.94
50	250	1	0.93

Table D.8: Grid search results for the decision tree analysis of the pure fairness policy. R^2 is the average R^2 evaluation value over the five cross-validation folds.

Max. Depth	Max. Nr. of Features	Min. Samples Leaf Nodes	R^2
4	5	sqrt	0.39
4	10	sqrt	0.39
4	25	sqrt	0.43
4	50	sqrt	0.43
4	100	sqrt	0.37
4	250	sqrt	0.44
4	5	log2	0.44
4	10	log2	0.47
4	25	log2	0.37
4	50	log2	0.31
4	100	log2	0.37
4	250	log2	0.44
4	5	1	0.57
4	10	1	0.57
4	25	1	0.57
4	50	1	0.57
4	100	1	0.57
4	250	1	0.57
7	5	sqrt	0.54
7	10	sqrt	0.55
7	25	sqrt	0.46
7	50	sqrt	0.56
7	100	sqrt	0.51
7	250	sqrt	0.53
7	5	log2	0.53
7	10	log2	0.54
7	25	log2	0.56
7	50	log2	0.54
7	100	log2	0.52
7	250	log2	0.51
7	5	1	0.62
7	10	1	0.62
7	25	1	0.62
7	50	1	0.62
7	100	1	0.62
7	250	1	0.62
10	5	sqrt	0.57
10	10	sqrt	0.60
10	25	sqrt	0.58
10	50	sqrt	0.58
10	100	sqrt	0.58
10	250	sqrt	0.56
10	5	log2	0.59
10	10	log2	0.60
10	25	log2	0.57
10	50	log2	0.58
10	100	log2	0.58

10	250	log2	0.55
10	5	1	0.63
10	10	1	0.63
10	25	1	0.63
10	50	1	0.63
10	100	1	0.63
10	250	1	0.63
15	5	sqrt	0.59
15	10	sqrt	0.59
15	25	sqrt	0.60
15	50	sqrt	0.61
15	100	sqrt	0.57
15	250	sqrt	0.58
15	5	log2	0.59
15	10	log2	0.60
15	25	log2	0.58
15	50	log2	0.60
15	100	log2	0.59
15	250	log2	0.57
15	5	1	0.61
15	10	1	0.62
15	25	1	0.63
15	50	1	0.64
15	100	1	0.64
15	250	1	0.63
25	5	sqrt	0.54
25	10	sqrt	0.57
25	25	sqrt	0.59
25	50	sqrt	0.59
25	100	sqrt	0.60
25	250	sqrt	0.57
25	5	log2	0.53
25	10	log2	0.57
25	25	log2	0.59
25	50	log2	0.59
25	100	log2	0.59
25	250	log2	0.59
25	5	1	0.54
25	10	1	0.58
25	25	1	0.61
25	50	1	0.63
25	100	1	0.63
25	250	1	0.63
40	5	sqrt	0.52
40	10	sqrt	0.55
40	25	sqrt	0.59
40	50	sqrt	0.59
40	100	sqrt	0.58
40	250	sqrt	0.53
40	5	log2	0.51
40	10	log2	0.56

40	25	log2	0.59
40	50	log2	0.60
40	100	log2	0.57
40	250	log2	0.58
40	5	1	0.51
40	10	1	0.57
40	25	1	0.61
40	50	1	0.63
40	100	1	0.63
40	250	1	0.63
50	5	sqrt	0.52
50	10	sqrt	0.56
50	25	sqrt	0.60
50	50	sqrt	0.60
50	100	sqrt	0.57
50	250	sqrt	0.58
50	5	log2	0.52
50	10	log2	0.55
50	25	log2	0.58
50	50	log2	0.60
50	100	log2	0.59
50	250	log2	0.58
50	5	1	0.51
50	10	1	0.57
50	25	1	0.61
50	50	1	0.63
50	100	1	0.63
50	250	1	0.63

Appendix E

Decision Tree Figures

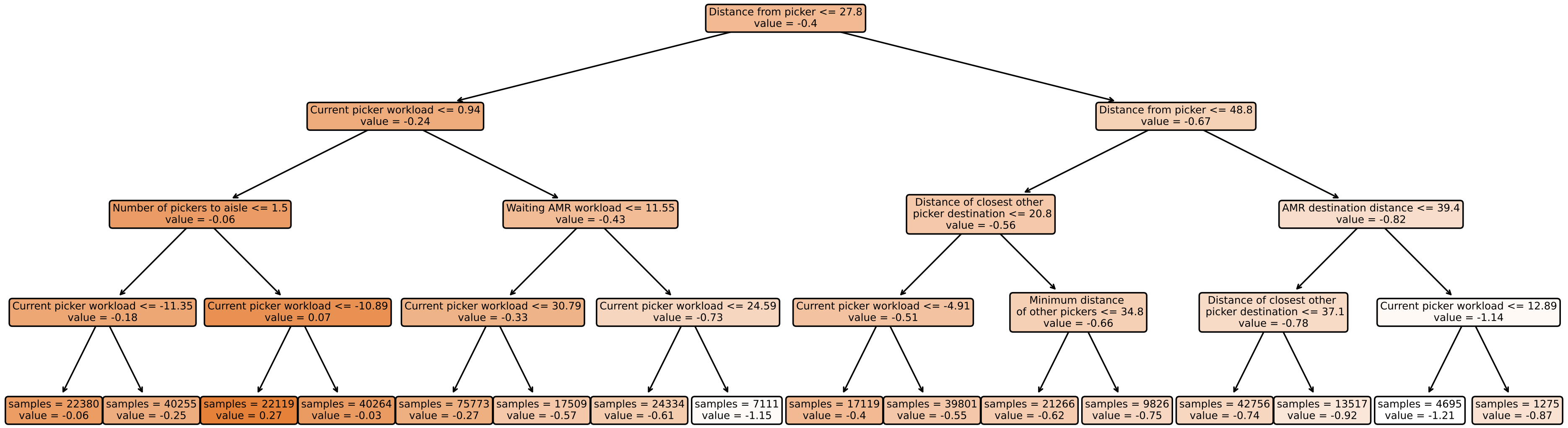


Figure E.1: Shallow decision tree for multi-objective policy 1. Decision paths must be read from top to bottom, with the left side indicating that the condition is satisfied and the right side that it is not. Darker colors indicate high node values and lighter colors represent low node values.

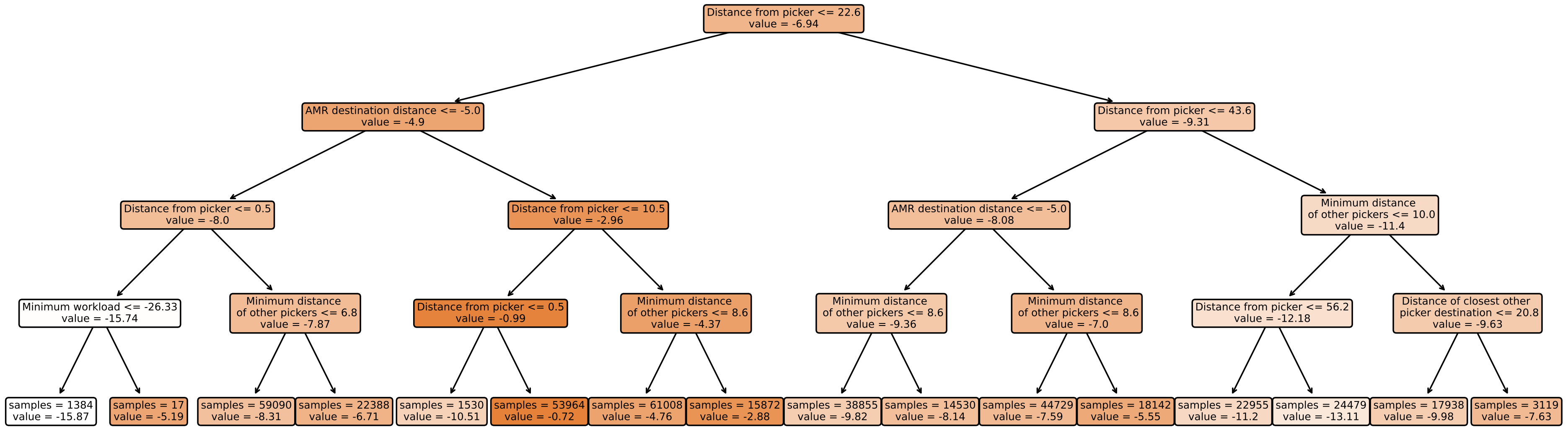


Figure E.2: Shallow decision tree for multi-objective policy 3. Decision paths must be read from top to bottom, with the left side indicating that the condition is satisfied and the right side that it is not. Darker colors indicate high node values and lighter colors represent low node values.

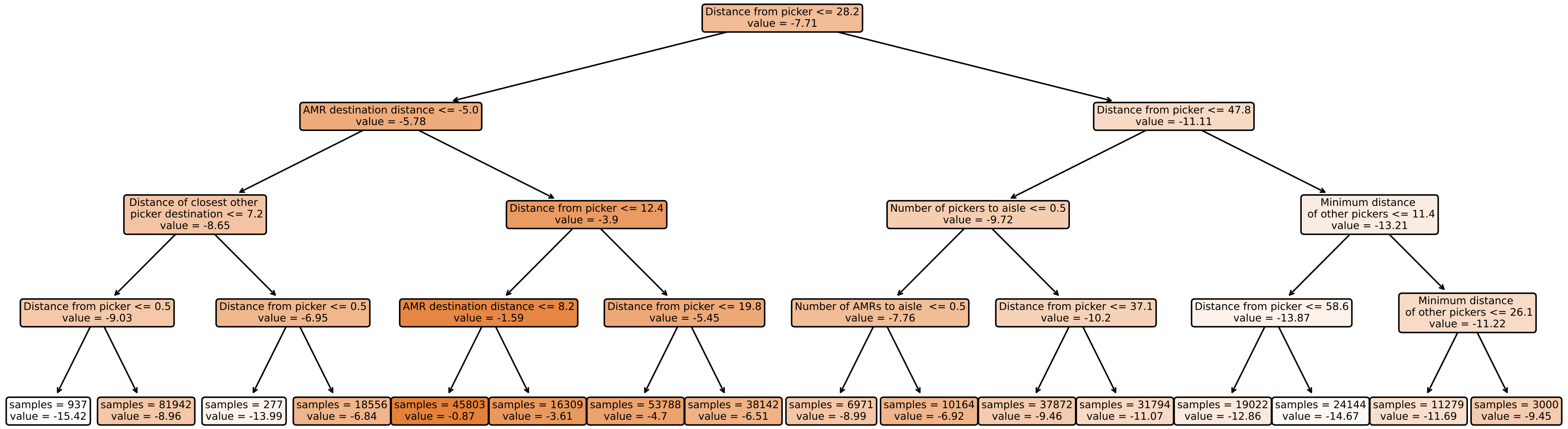


Figure E.3: Shallow decision tree for multi-objective policy 4. Decision paths must be read from top to bottom, with the left side indicating that the condition is satisfied and the right side that it is not. Darker colors indicate high node values and lighter colors represent low node values.

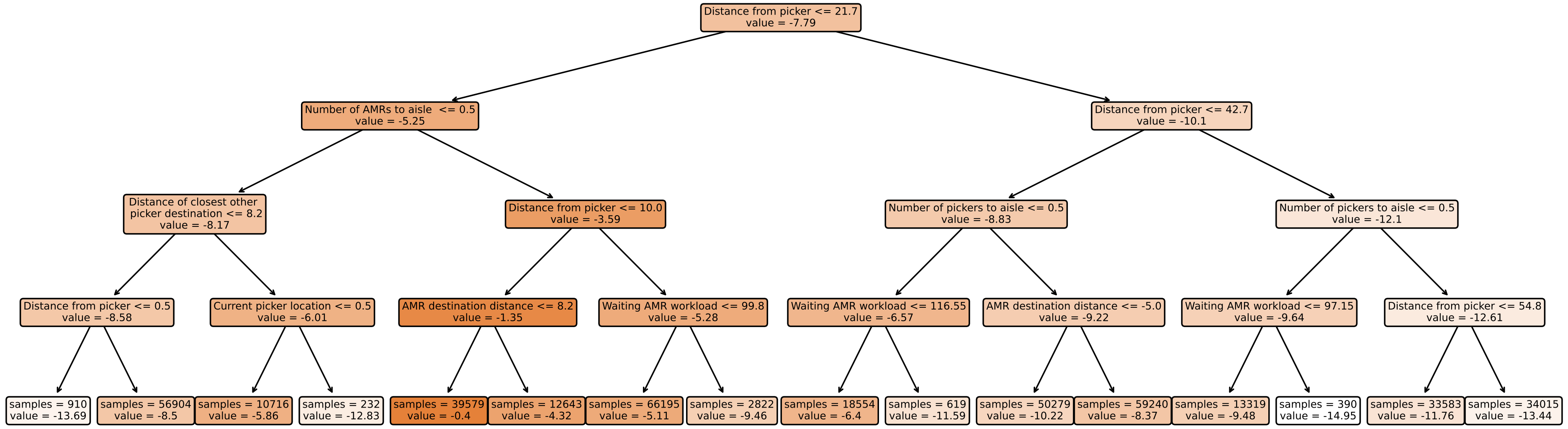


Figure E.4: Shallow decision tree for multi-objective policy 5. Decision paths must be read from top to bottom, with the left side indicating that the condition is satisfied and the right side that it is not. Darker colors indicate high node values and lighter colors represent low node values.

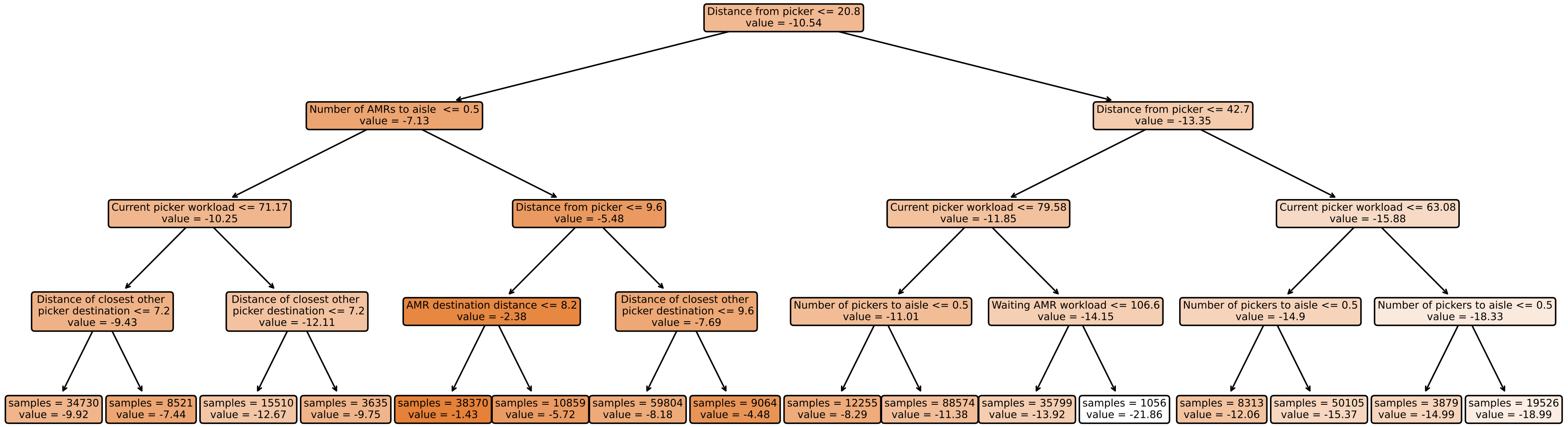


Figure E.5: Shallow decision tree for multi-objective policy 6. Decision paths must be read from top to bottom, with the left side indicating that the condition is satisfied and the right side that it is not. Darker colors indicate high node values and lighter colors represent low node values.