

Tackling In-Store Picking Operations by leveraging Proximal Policy Optimization

Tiago Miguel Cardoso Moreira

Dissertação de Mestrado

Orientador na FEUP: Prof. João Claro

Co-orientador: Prof. Fábio Neves-Moreira



Mestrado em Engenharia e Gestão Industrial

2024-06-27

Abstract

This dissertation is about tackling the in-store picking challenges retailers face that come up with the new grocery shopping paradigm of integrating the online channel with the physical channel following an omnichannel approach that can better respond to the different kinds of demand customers have.

It is motivated by the growing efforts that serving customers in a competitive market requires from the customer's point of view - where keeping the service level as high as possible is important - and from the retailer's point of view - where being efficient in its operations is important to keep their costs low while providing better services.

As such, the goal of the dissertation was to train an agent to learn a picking policy using Proximal Policy Optimization, a deep Reinforcement Learning algorithm, capable of travelling the minimum possible distance and avoiding encounters with physical customers in an environment modelled as a Markov Decision Process.

To do so, the first step was to devise a set of features that would drive the learning process. Followed by that, came the creation of synthetic stores with different sizes. A hyperparameter tuning technique was employed to tailor the hyperparameters that serve as input to the model to each configuration and then do a more extensive training process with the sets of hyperparameters that achieved better performance. The results were tested and then based on the stability of the training process and the performance a transfer learning technique was reproduced to verify if it was possible to retrain an agent that was trained to do a simpler task in a more complex one.

Finally, this strategy was employed in a real world store configuration, in which the store was divided into subsections that were sequentially added to the training process. Some important real world considerations were not implemented which leaves room for future work regarding employing this methodology in driving efficient in-store picking in real world scenarios within other parameters. The obtained results validate this approach for the real world scenario as the learning phase reached a value of 0.95 encounters per product picked and 77.28 reward per product picked.

Resumo

Esta dissertação aborda os desafios do processo de levantamento de encomendas em loja que os retalhistas enfrentam com o novo paradigma de compras de supermercado que integra o canal online com o canal físico, seguindo uma abordagem omnicanal que pode responder melhor aos diferentes tipos de procura dos clientes.

É motivada pelos crescentes esforços que servir os clientes num mercado competitivo exige do ponto de vista do cliente, onde manter a qualidade do serviço o mais alto possível é importante, e do ponto de vista do retalhista, onde operações eficientes são cruciais para manter os custos baixos enquanto se desenvolve a qualidade dos serviços.

Assim, o objetivo da dissertação foi treinar um agente para aprender uma política de levantamento de encomendas usando o método *Proximal Policy Optimization*, um algoritmo de *deep Reinforcement Learning*, capaz de percorrer a menor distância possível e evitar encontros com clientes físicos num ambiente descrito como um *Markov Decision Process*.

Para isso, o primeiro passo foi conceber um conjunto de características que guiassem o processo de aprendizagem. De seguida, procedeu-se à criação de lojas sintéticas com tamanhos variáveis. Também se procedeu ao uso de uma técnica de *hyperparameter tuning* para adequar os conjuntos de parâmetros a cada configuração e, em seguida, realizar um processo de treino mais extensivo com os melhores conjuntos de selecionados. Os modelos resultantes foram testados e, em seguida, com base na estabilidade do processo de treino e no desempenho, foi aplicada uma técnica de *transfer learning* para verificar se era possível voltar a treinar um agente - inicialmente treinado para realizar uma tarefa mais simples - para realizar uma tarefa mais complexa.

Por fim, esta estratégia foi implementada numa loja real, na qual a loja foi dividida em sub-seções que foram adicionadas sequencialmente ao processo de treino. Algumas considerações importantes do mundo real não foram implementadas, o que deixa espaço para trabalho futuro relacionado com a utilização desta metodologia para guiar um processo de aprendizagem de políticas de levantamento de produtos eficiente em loja em cenários do mundo real. Os resultados obtidos validam esta abordagem para o cenário do mundo real, sendo que a fase de aprendizagem alcançou um valor de 0.95 encontros por produto levantado e uma recompensa de 77.28 por produto levantado.

Agradecimentos

Esta parte da dissertação é uma das mais fáceis, mas, também, das mais difíceis. Afinal de contas, reconhecer a todas as pessoas que de algum modo contribuíram para a realização deste trabalho o seu mérito deveria ser algo fácil, mas arranjar forma de resumir tudo aquilo que poderia descrever ao espaço que mais ou menos uma página me confere dificulta-me a vida.

Começando pela vertente académica, gostaria de agradecer a todos os professores que lecionaram as unidades curriculares da licenciatura e do mestrado, permitindo que eu e os restantes colegas chegássemos a este ponto. No entanto, gostaria de particularizar alguns professores. Em primeiro lugar, às professoras Henriqueta Nóvoa e Teresa Galvão agradeço pela paciência e disponibilidade que demonstraram ao ajudar-me a ultrapassar um pequeno (grande) obstáculo. Ao professor José Machado agradeço pelo impacto positivo que deixa em todos aqueles que tiveram a sorte de o ter como professor, bem como pela generosidade que demonstrou comigo ao disponibilizar o seu tempo para me ouvir e aconselhar. Ao professor Pedro Amorim pelo seu papel na minha aprendizagem ao longo deste mestrado e por me ter exposto à excelente oportunidade que foi este projeto.

Passando à orientação da dissertação, gostaria de agradecer ao orientador na FEUP, o professor João Claro, pela ajuda na realização da dissertação e flexibilidade que demonstrou em tudo o que a abrangia. Tenho também de expressar toda a minha gratidão ao co-orientador da dissertação e orientador no INESC TEC, professor Fábio Neves-Moreira, não só pelo voto de confiança, o qual espero não desiludir, mas por toda a ajuda e tempo que disponibilizou, pelos conselhos que deu, que serviram sempre como uma excelente indicação do caminho a seguir, e pela contribuição final que teve na dissertação, sendo de enorme ajuda à sua finalização. O meu agradecimento ainda ao INESC TEC e ao CEGI pela possibilidade de lá realizar o projeto da dissertação.

No que diz respeito à vertente pessoal, começo pela minha família. À minha mãe agradeço por tudo aquilo que fez por mim, mesmo quando eu não sabia que era para meu bem, e me proporcionou, por me ensinar a fazer contas de dividir - que sem isso com certeza não estaria aqui - e pela forma como sempre encorajou (há quem diga obrigou) que desse prioridade à minha formação académica. Agradeço à minha irmã Cláudia que me foi sempre preparando, à sua própria maneira especial, para a paciência que esta fase ia exigir de mim, por estar sempre presente, mesmo estando ausente, por tudo o que já me proporcionou, pela constante preocupação comigo em várias vertentes e por obrigar os amigos a ver filmes de criança por causa do irmão (há muitos anos atrás). Agradeço à minha irmã Tânia pelas inúmeras vezes em que me ouviu a falar dos meus problemas e pela forma como naturalmente nos entendemos que não impede que façamos o outro ver as coisas de uma perspetiva diferente. Aos meus avós, que sempre estiveram presentes e mostraram entusiasmo por cada objetivo alcançado, também devo o meu agradecimento. Agradeço ao Arran por estar sempre disponível para uma sessão de "brainstorming", por ser uma pessoa com quem posso sempre contar e por tudo aquilo o que faz por mim.

Não podia deixar de particularizar o meu agradecimento à Juliana, a minha amiga mais antiga (mais de uma década), pelas horas todas que me aturou não só nos últimos meses, mas nos últimos

anos, provando sempre ser uma amiga exemplar com quem sempre pude contar. Ao Nuno também não podia faltar, que na sua inutilidade é sempre capaz de fazer a diferença em tudo a que se propõe e com quem posso ter das conversas mais interessantes sobre qualquer que seja o assunto, bem como passar horas a resolver enigmas.

À Ana, à Beatriz, à Cassandra, à Inês, às Joanas, à Maria João e às Martas agradeço-lhes por serem amigas tão presentes que fizeram com que estes últimos meses fossem passados com mais leveza e diversão.

Um agradecimento que me custa admitir é aquele que devo à Sara que, sem saber - se bem que se alguém lhe perguntar o seu amor próprio não lho permitiria negar -, fez das maiores diferenças na minha vida universitária mesmo nos momentos em que só me sabia insultar.

À Bea e ao Ricardo agradeço pela partilha de sentimentos sobre o estado do estudo ou dos trabalhos que sempre ajudou a aliviar um pouco a tensão e a sentir menos culpa e ainda à Bea pelas vezes em que devíamos estar a trabalhar, mas ficávamos antes a falar de cães ou a refletir sobre outros assuntos.

A Kali e a Clio não podiam ficar esquecidas como sendo as melhores cadelas que me trazem das maiores felicidades que eu poderia sentir e me ajudaram muito pela sua mera presença.

Por fim, se houvesse uma opção de "Todas as anteriores" quase que era correta para agradecer à Francisca, a pessoa que mais me aturou e leva com grande parte do meu mau humor, frustrações e preocupações. Obrigado por tudo aquilo que és e pela diferença que fazes na vida de todos à tua volta e especialmente na minha, pela fé incondicional que tens nas minhas capacidades mesmo quando eu não partilho do sentimento, por tudo aquilo o que fazes por mim, mesmo as coisas que não vejo tão facilmente, pelos constantes esforços para tornar a minha vida mais fácil, pelo orgulho que tens em mim,... Enfim, por tudo e mais alguma coisa.

"... the powers of instruction are of very little efficacy except in those happy circumstances in which they are practically superfluous."

Richard P. Feynman

Contents

1	Introduction	1
1.1	Motivation and Contextualization	1
1.2	Goals	3
1.3	Project Workflow	3
1.4	Dissertation Structure	4
2	Literature Review	5
2.1	Store-based Picking	5
2.2	Picker Routing Problem	7
2.3	Dynamic Routing Problem	8
2.4	Dynamic Picker Routing Problem	10
2.5	Reinforcement Learning	12
2.5.1	Introduction to Reinforcement Learning	12
2.5.2	Reinforcement Learning in the Supply Chain Context	13
2.5.3	Deep Reinforcement Learning	14
2.5.4	Transfer Learning	14
3	Problem Description	17
3.1	Store Environment	17
3.2	Store Dynamics	18
3.3	Modelling the problem as a Markov Decision Process	19
4	Methodology	23
4.1	The Problem of Dimensionality	23
4.1.1	Curse of Dimensionality	24
4.1.2	Impact on Traditional Reinforcement Learning	25
4.2	Proximal Policy Optimization	26
4.2.1	Clipped Surrogate Objective	27
4.2.2	Value Loss Function	29
4.2.3	Final Objective	29
4.2.4	Example in the Problem Context	30
4.3	Feature Engineering	31
4.4	Hyperparameter Tuning	33
4.4.1	Hyperparameters Tuned	33
4.4.2	Hyperparameter Tuning Strategy Employed in PPO	35
4.5	Parallel Environments	36

5	Results	37
5.1	Creation of synthetic stores	37
5.2	Hyperparameter Optimization	38
5.3	Impact of Layout Size	41
5.3.1	Algorithm Convergence	41
5.3.2	Performance metrics	43
5.4	Retraining Models in Increasingly Complex Environments	46
5.5	Real World Store Configuration	47
6	Conclusion	49
	References	51
A	Sustainable Development Goals	57
B	Q-Learning	59
C	Synthetic Stores Graphs	61
D	Simulation Parameters	63

Acronyms and Symbols

ADP	Approximate Dynamic Programming
A2C	Advantage Actor Critic
A3C	Asynchronous Advantage Actor Critic
BOPS	Buy Online, Pick Up In-Store
DQN	Deep Q-Networks
DRP	Dynamic Routing Problem
PP-OOS	Post Purchase Out-of-Stock
PRP	Picker Routing Problem
SFS	Ship From Store
SOP	Sequential Ordering Problem
TPE	Tree-structured Parzon Estimator
TRPO	Trust Region Policy Optimization
TSP	Travelling Salesman Problem

List of Figures

2.1	Example of dynamic vehicle routing. Adapted from Pillac et al. (2013).	9
2.2	High-level block diagram of RL. (Memarian and Doleck, 2024)	13
3.1	Store environment that represents the picker and customers doing their picking paths.	19
4.1	Example of the path taken by the agent in order to pick products in nodes 69 and 102 in that sequence.	30
5.1	Comparison of Obtained Rewards for Small Layouts between sets 1 and 4 (first 2500 episodes).	42
5.2	Comparison of Obtained Rewards for Medium Layouts between sets 2 and 5 (first 3000 episodes).	42
5.3	Comparison of Obtained Rewards for Large Layouts between sets 1 and 3 (first 3500 episodes).	43
5.4	Evolution of Customer Encounters per Product Picked.	44
5.5	Real World Store Graph Representation with Subsections.	47
C.1	Small Store Graph Configuration.	61
C.2	Medium Store Graph Configuration.	61
C.3	Large Store Graph Configuration.	62

List of Tables

5.1	Sets of PPO hyperparameters used to build training configurations.	39
5.2	Reward metrics obtained during the training procedure.	44
5.3	Performance Metrics upon Testing for 500 Episodes.	45
D.1	Parameters used to build the simulation configurations.	63

Chapter 1

Introduction

The Chapter that follows is the introduction to the dissertation going from the reasons behind it to the activities' organization during its development. Section 1.1 sets the context of the dissertation and provides a motivation to studying solutions for in-store picking given the current paradigm of e-commerce among businesses and grocery retailers in particular. It also presents how the problem will be addressed in order to reach the defined goals successfully. Section 1.2 provides the goals of the dissertation concerning the two main factors that are considered - minimum travelling distance and customer avoidance - as well as the importance of reaching those goals. Section 1.3 concerns the way the project was tackled in the sense of the organization of different activities throughout the semester that went from finding the state of the art to extracting results. Section 1.4 presents the structure of the dissertation by providing a small preview of the different chapters that follow.

1.1 Motivation and Contextualization

E-commerce is an activity that relates to having online stores and selling products through the online channel. Along with the use of new technologies spreading across a wide array of people from every background, online shopping in different kinds of markets (from clothes to groceries) is getting more popular (Caro et al., 2020). Younger people are leading busier lives and do not want to waste their free time going to buy things they can have easily delivered to their homes while some older people do not have the same ease as they used to to leave the house and carry heavy bags of groceries, for example, opting for the convenience that comes with online shopping (Rosário and Raimundo, 2021). As such, the possibility of acquiring products through the internet has been growing exponentially and it is becoming part of all kinds of businesses as a regular service instead of an extra service for which the customer needs to pay high fees for deliveries or higher prices for the same product sold online. Duval (2019) surveyed a group of people and concluded that free delivery incentivizes the purchasing behaviour and only 40% are willing to pay for a standard two day shipping. Adapting to that reality requires supply chain agility. In order to integrate that as a part of the business, companies come up with different solutions to facilitate

logistics of handling online purchases which have two main issues to tackle that are intertwined with each other: order picking and last mile fulfilment.

Order or product picking is typically done in warehouses or distribution centers that are built for it regardless of the aim being to deliver customers or to replenish physical stores' stock. Last mile fulfilment aims at serving the maximum amount of customers in the minimal amount of time by adding the minimal amount of costs on both sides (the company's and the customer's). It relates with the final process of the supply chain that is the product going from the business to the customer. Towards achieving that purpose proximity to the customer is valued. However, having a high number of warehouses in a short distance range is not feasible. (Hübner et al., 2016)

Typical grocery retailers that have been around for years with physical stores and are in the e-commerce market have a big advantage that can be exploited to get closer to the customer which is in-store picking. This relates to the activity of picking products to fulfill online orders inside the physical store where physical customers can enter as well. That way the retailers use their existent infrastructure to reduce delivery costs or the traveling distance (in the case of the customers going to the store to simply get their already picked order) by being much closer to the customers as the density of physical stores of any big grocery retailer is considerably high. This creates an order picking issue as stores are not exactly made for efficient order picking and it is how the two issues are connected: by targeting the order delivery issue to get closer to the customer, an order picking problem arises. (Dethlefs et al., 2022)

Two typical strategies that are used by retailers are the Buy Online, Pick Up In-Store (BOPS) and Ship From Store (SFS). Both consist in doing in-store product picking, but in the first the customer goes to the store to retrieve the order while in the second the order is shipped to a given address. (Jin et al., 2018; He et al., 2021)

The COVID-19 pandemic also had an impact in the growing tendency to adopt online channels as people not only were restricted from leaving their home but also did not want to endanger themselves by being exposed to the virus. As such, the pandemic created some fear among retailers as they saw physical shopping activity decrease and had to adapt the majority of their supply chain to an online sales channel that was adopted by most customers and remained even after the end of the pandemic (Bai, J., Fu, F., Guan, R., Hoffman, S., Karnani, P., Mysore, M., and Touse, S., 2021). However, during the post-pandemic period while social distancing was still a reality in the customers' daily lives and both the online channel had already gained a lot of traction and was adopted by customers and the physical channel was coming back with people leaving their houses more often the question of how adapting to that reality would look like remained (Ketzenberg and Akturk, 2021).

As such, this dissertation's theme is about tackling the in-store picking problem of a big grocery retailer in Europe, modelled as a Markov Decision Process (MDP). This is a dynamic problem as the picker needs to make decisions regarding the path to take between two fixed positions by considering different factors such as the customers occupying the neighboring positions and the travelling distances. These decisions need to be made as fast as possible so the process is not delayed given that delays translate into higher costs which influences the retailer's ability to provide

a quality service to the customer charging as low as possible. Different Reinforcement Learning (RL) methods - a machine learning technique - have shown good results in different contexts such as healthcare (Komorowski et al., 2018), finances (Deng et al., 2017) and demand response (Ru-elens et al., 2017) in providing a good performance and a fast response. As such, to address this problem a deep RL algorithm called Proximal Policy Optimization (PPO) was used. The project was developed together with INESC TEC, a Portuguese research organization.

1.2 Goals

Considering the context of this dissertation, not only the work that was developed aims at learning a picking policy with an agent that is able to navigate physical stores in an efficient way - one that minimizes the travelled distance - but also at being able to dynamically avoid customers that are present in the store. The latter not only tackles the social distancing issue if that becomes a reality ever again but also allows preserving the shopping experience of physical customers without making them feel like they are in a warehouse and constantly crossing paths with an order picking cart.

Solving these issues is important as it would bring order into the in-store picking process that remains with very little optimization creating a lot of confusion with order pickers that do not have enough experience. If they receive a random product list and know exactly where to go to optimize performance that would facilitate their task which would reduce the costs associated with order picking. Also, it would help keeping the role of physical stores as a differentiating factor among retailers where competition is high (Dennis, 2018).

1.3 Project Workflow

Regarding the workflow towards the development of the project concerning the main activities that were performed during the period of the dissertation, there were several different moments.

Initially, the priority was to find the state of the art pertaining the approached issues as to find the importance of the contribution that employing RL to accomplish the proposed goals had as well as valuable insights to the dissertation. The next step was to extend the retail store environment simulator which required learning the functioning of the existent and extensive store simulator in order to be possible to introduce modifications and understand the influence of introducing new factors. Afterwards, the main development began and based on metrics extracted from the model training the performance was assessed which allowed to do the necessary adjustments throughout the project so that each step of the development would bring relevant additions to the final goals. Finally, the final stage of extracting results took place and allowed to reflect on the goals' accomplishment.

1.4 Dissertation Structure

This dissertation is structured as follows. Chapter 2 is dedicated to the literature review about the issues relating to in-store picking and routing problems as well as the method that was used in developing the dissertation, RL and in particular PPO. Chapter 3 describes the problem that is being tackled as a MDP going through its typical elements such as the states, actions, transitions and rewards. It also describes the assumptions that were considered and the decision making power of the agent. Chapter 4 approaches the methodology that was followed during the project offering a more broad explanation on the problem of dimensionality that is faced as well as the deep RL algorithm that was used. Also, it presents the techniques that were used to reach the final results. Chapter 5 regards the findings of the project presenting metrics that assess the performance of the training that was carried out. Finally, Chapter 6 offers some conclusions from the work that was carried out.

Chapter 2

Literature Review

This Chapter aims at reviewing the state of the art regarding different subjects that relate to the development of the project. Section 2.1 is focused on in-store picking activities going through the challenges it poses as well as its main advantages and problems. Section 2.2 is based on the Picker Routing Problem (PRP) and the main focus of route optimization. Section 2.3 regards the introduction of dynamism to solve routing problems when new factors are introduced after the initial solution is reached. Section 2.4 is about the combination of the PRP with the Dynamic Routing Problem (DRP). Finally, Section 2.5 introduces RL methods as well as some uses in the supply chain context.

2.1 Store-based Picking

Recently, retailers have been taking an omnichannel approach regarding their activities by integrating the online and physical channels (Difrancesco et al., 2021). Typically, online order fulfillment depends on dedicated distribution centers, but fulfillment is more efficient when the customer is closer (Dethlefs et al., 2022). In order to reach that goal of proximity, some retailers are trying to use the same stores that satisfy demand for walk-in customers, their physical store network, to fulfill online orders as well, which allows quicker expansion without the need for high investment in infrastructures (Hübner et al., 2016; Ognibene et al., 2021). Proximity to the customer means a lower delivery cost or distance travelled for in-store pick-up which benefits the business since there is a limited willingness to pay for delivery services (Hübner et al., 2016).

Additionally, it does not add complexity to the supply chain as it does not require another network flow from the warehouse to the store (Neves-Moreira and Amorim, 2024). Indeed, even if it poses new challenges it also tackles existing ones such as the latter by eliminating it all together.

Some new challenges, however, arise, such as the inventory management problem that comes along with in-store picking, since physical stores do not hold the same inventory capacity as a warehouse would, causing a higher stock-out risk as they would have to fulfill both online and

physical demand. (Arslan et al., 2021). In turn that raises the challenge of integrating the stock levels in all fronts.

According to Anshu et al. (2022), the repurchase intention of the online customer in a retailer is linked to the customer's attitude towards it - the consumer experience that was provided which influences the impression the customer has regarding factors as the service and the products. Hoang and Breugelmans (2023) state that Post Purchase Out-of-stock (PP-OOS) - which is the unavailability of stock only having happened or being noticed after the customer already made the purchase online - is a common event in online grocery shopping and customers perceive it as a drawback. Poor inventory management can lead to a situation in which the availability showed to the consumer via the online channel and the actual availability in the physical store do not match which can lead to PP-OOS and cause a bad reaction on the customer's end. Product substitution - replacing the unavailable product in the order for a similar one considering factors such as brand and flavor - is a strategy used by retailers, but it can lead to dissatisfaction. This is linked to the service the retailer provides and can affect the repurchase intention.

Stores are not tailored for picking. It is important to consider that stores have a focus on customer interaction as well as product display while distribution centers are designed with a focus on efficient order picking and packing with more available space for storage and travelling through the aisles. That will lead to higher order processing costs for in-store picking (Dethlefs et al., 2022; Hübner et al., 2016).

In-store picking also represents a downside on the customer perception regarding how walk-in customers perceive pickers in the store. According to Hosseini et al. (2014), the store atmosphere, along with other factors, has a significant impact and a positive relationship with the customer perception. Retailers do not want to disrupt the consumer experience for walk-in customers with pickers moving around the store with their carts occupying the limited space they have and making the store feel like a warehouse (Adhi et al., 2021).

There is little optimization implemented by retailers regarding the in-store picking process as it is a relatively recent approach - usually pickers receive a shopping list with items in a random or alphabetic order and then they go through the cashier, a very similar process to the walk-in customer's - despite being explored through other dimensions such as marketing and pricing strategies (Ognibene et al., 2021).

Chou et al. (2021) propose using the methodology behind the Traveling Salesman Problem (TSP) - finding the shortest route that includes every location only once - to order the product list in which the products correspond to the cities.

To tackle some of the issues regarding in-store picking, Difrancesco et al. (2021) try to determine the optimal amount of time for order batching - process of accumulating orders and picking them all at once instead of doing it as each order occurs - as well as the optimal number of pickers and packers while considering the trade-off between service level and cost and distance themselves from previous works on similar matters by using a single store scenario that deals with online and walk-in orders simultaneously in a SFS context (in-store picking with home delivery). The research found that delaying order picking - higher order batching - improves picking efficiency, but

also increases the probability of stock-outs, since walk-in customers are considered as well, and leads to a delay in the final delivery. Regarding SFS, Arslan et al. (2021) reached the conclusion that its integration led to a decrease higher than 50% in lost online sales.

BOPS has become a relevant online order fulfillment method (Jin et al., 2018). It is the process in which a customer makes the purchase online and then physically picks it up and the order itself is picked directly from the store. It differs from SFS in the sense that there is no shipment from the store to the customer involved. Yang et al. (2022) found that BOPS can lead to cross-selling opportunities and increase store traffic. Gao and Su (2017) studied the impact of BOPS on store operations and found that it may not be suitable for all products such as the ones that sell well in stores and that it helps obtaining new customers, but if the existing customers that go to the physical stores make the shift towards that alternative it leads to reduced profit margins. This strategy also improves customer confidence, because those who order online for home delivery might hesitate to make a purchase to avoid returning the product in case they need to. By allowing customers to review their order before taking it home, this approach minimizes the likelihood of returns, making sure that consumers are satisfied with their purchase before leaving the store (Li et al., 2019).

Pietri et al. (2021) aimed at improving in-store picking processes by introducing optimization. The research showed the operational performance of different approaches while doing in-store picking in a BOPS system using a no optimization scenario as baseline - given that it follows the current in-store picking paradigm - and an open TSP, a Sequential Ordering Problem (SOP) - one that introduces some constraints to the TSP formulation - and a relaxed SOP scenarios to compare performances. The study concluded that the TSP approach is better in terms of travelling time while the relaxed SOP offers a good compromise between travelling time and choosing picking sequence that obeys certain constraints pertaining factors like the product resistance or size.

Seghezzi et al. (2022) studied ways to increase the efficiency of in-store picking. The main findings point to the implementation of a picking area dedicated to storing items responsible for 60% of the online sales as a way to decrease picking time and to the consideration of batching policies within certain conditions to reduce the travelling distance per order.

This dissertation aims at reaching an efficient store-based picking model that can achieve good operational performance while not disrupting the physical customer's shopping experience as retailers would not want to jeopardize the perceived image of going to the physical stores.

2.2 Picker Routing Problem

Order picking represents a large portion of logistic efforts for companies regarding labor and time resources which makes it costly (Masae et al., 2020). It is accomplished mainly via picker-to-parts systems - the order picker walks or drives along the aisles to get the desired products (De Koster, 2008). A study conducted in 1988 in the United Kingdom showed that 55% of total operating costs in a warehouse are due to order picking activities (Tompkins et al., 2010) and despite automation being a way of reducing costs many businesses still use manual picking to maintain flexibility

dealing with the different SKU shapes and sizes, variability of demand and seasonality as well as to avoid the investment required to automate (Petersen and Aase, 2004).

The PRP is a problem that aims to minimize the total distance/ time travelled by the picker introduced by Ratliff and Rosenthal (1983) that solved it through establishing an analogy with the TSP after converting the warehouse's positions considered in the study to a Steiner graph and then finding the tour subgraph that represented the optimal path. The time the algorithm that was used took to solve the problem was proportional to the number of aisles that was considered.

Pansart et al. (2018) used exact algorithms to tackle the problem based on mixed integer programming and dynamic programming and reached satisfying results for realistic size warehouses.

Theys et al. (2010) used the Lin-Kernighan-Helsgaun TSP heuristic to solve the problem and reported an average saving of 47% in route distance.

Chen et al. (2013) proposed a new routing algorithm based on Ant Colony Optimization for a typical layout warehouse with narrow pick aisle - an example of the usage of meta-heuristics to solve the PRP.

While developing solutions and new strategies for the PRP, the primary focus often remains on optimizing the route itself. However, it is crucial to consider real-world constraints that significantly impact the routing process. For instance, in a grocery setting, frozen products should not be picked at the beginning of a route to maintain their integrity, and heavy items must be placed beneath fragile ones to prevent damage. Thus, considerations like weight, fragility, and product category are essential in influencing the routing strategy to ensure efficient and damage-free deliveries (Thomas Chabot and Renaud, 2017).

Where order picking is concerned, so are other issues such as order batching, picker assignment and batch sequencing that when integrated can lead to a high complexity difficult to solve problem. Cao et al. (2023) used a hybrid iterated local search algorithm with heuristic rules to solve that problem as well as studied the impact of such integration concluding that it can lead to better results like smaller tardiness and reduced length of the picking route.

Neves-Moreira and Amorim (2024) studied the PRP for the first time in a store, which introduces a number of new scenarios such as the store architecture that is not suitable for in-store picking and customers travelling through the store with non-optimized shopping paths. In light of that some of the graph properties that are assumed in the literature are not suitable to the store context, but for the warehouse's instead.

2.3 Dynamic Routing Problem

A DRP is a problem in which a set of locations have to be visited by an agent, but the information stochastically changes throughout the operational timeline, creating opportunities to adapt and make decisions based on the newly available information being typically studied in the vehicle routing problem. It is usually modelled as a MDP and can use states, decisions and transitions to structure the progression of decision-making and the unfolding of information (Ulmer et al., 2020).

Some new factors regarding the complexity that real world problems can have such as the concern for sustainability, population growth, e-commerce growth, the desire for speed and technology evolution (e.g., increased digital connectivity and big data) have not only enabled, but also demanded that new strategies are devised to handle the DRP (Savelsbergh and Van Woensel, 2016).

Berbeglia et al. (2010) studied a subclass of the DRP called dynamic one-to-one pickup and delivery problem - transporting objects or people (each have a given origin and destination) from one point to another dealing with real time information. In some algorithms each time a new request is received, the existing solution is updated either through an efficient insertion technique or by undergoing a complete reoptimization.

Pillac et al. (2013) identified evolution and quality of information as two important dimensions of real-world applications of the vehicle routing problem that respectively mean the information may change as the route execution progresses and the representation of uncertainty on the available information. Also, when it comes to dynamic and stochastic problems the input is partially or totally unknown and throughout the execution it is revealed dynamically and may lead to changing the initial planned route. There are several metrics to measure the degree of dynamism. Finally, the review on the existent literature found that stochastic aspects have not been considered in a big part of the work developed in dynamic routing concluding that it could lead to better fleet performance and less operating costs.

In Figure 2.1 the dynamism of the route is better shown as in t_1 after the execution of the initially planned route began and the first client had already been served two new requests appear and the route is modified to include them.

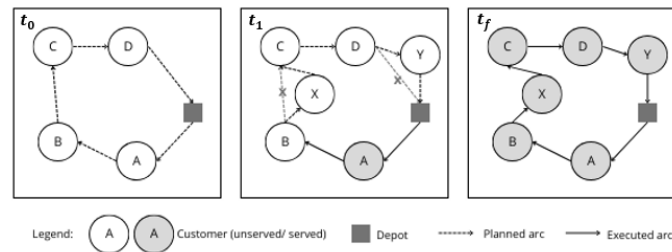


Figure 2.1: Example of dynamic vehicle routing. Adapted from Pillac et al. (2013).

Novoa and Storer (2009) studied approximate dynamic programming algorithms - rollout algorithms - for the single-vehicle routing problem with stochastic demands. A rollout algorithm is an approximate method to solve large dynamic programming problems and assumes a sub-optimal initial policy is available that the algorithm iteratively improves using simulation to estimate the value of decisions based on future possible outcomes - in the context of approximate dynamic programming the algorithm approximates the value function. They found that the optimal rollout algorithm enhances performance by approximately 8-10% compared to strategies that do not account for stochastic and dynamic factors.

Powell (2019) addresses the problem of finding the best policy - method for decision-making based on available information captured at time t by a state variable S_t . The unified framework for stochastic programming that the study proposed has played a crucial role, as researchers have increasingly adapted it for addressing dynamic issues modeled as MDPs.

Regarding dynamic programming and the MDP, the three curses of dimensionality have to be considered given that the value of the states are based on the value of the current state as well as the value of the state in the next point in time and transition decisions choose the state with the maximum value. As such, if the state space has I dimensions and each can take L possible values there may be up to L^I different states, the same goes for an outcome space with J dimensions that can take M possible values leading to M^J different outcomes and, once again, the same goes for the action space that can have up to K dimensions, take N possible values resulting in N^K possible decision vectors that can result in a lot of possibilities representing the three curses of dimensionality. Not all problems suffer with the three curses of dimensionality, some might have a state space with millions of dimensions and a very simple action space, but some problems are not as convenient and in order to reach a satisfying solution some strategies have to be deployed that mitigate the curses of dimensionality (Powell, 2007).

2.4 Dynamic Picker Routing Problem

A dynamic version of the PRP in some business contexts when new information that comes up along the route execution can improve how tasks related to picking are planned or carried out can be considered.

Gong and De Koster (2008) studied dynamic picking in the warehouses of online retailers in response to the constant need to improve customer responsiveness as well as the new information technologies that are fostering the possibility to react to changing scenarios in real time. The authors followed a stochastic polling theory approach in which picking information can dynamically change in a picking cycle. A polling model is a system of multiple queues accessed in cyclic sequence by a single server - in the research's context the server represents the order picker and the queues are the number of storage positions containing a different product. The research showed that polling-based picking systems typically result in shorter order waiting and throughput times as well as higher on-time service completion ratios, especially for high order arrival rates, than traditional batch picking systems that use ideal batch sizes.

Lu et al. (2016) introduced an interventionist routing algorithm that re-optimizes the picker route during the picking operation and were able to surpass static and heuristic dynamic order picking algorithms. An interventionist algorithm is one that actively intervenes in or changes the system it operates within based on the data it gathers in real-time. When an order picker receives updated information about the requested items of new orders during the picking operation, the algorithm determines the route to take in order to minimize the amount of distance travelled. Upon comparison with static order picking, the interventionist algorithm leads to a reduction in average order completion time and an increase on travelling distance, but the differences become smaller

when the order arrival rates increase. Against a modified largest gap heuristic (the modification is to allow intervention in the picking process), the interventionist algorithm outperforms the former with an average order completion time and average travelling distance per order 15% and 12% smaller, respectively, in the best scenario - according to the order arrival rates the improvement of the proposed approach may be smaller.

According to some authors in the literature, dynamic information in store environments usually only relates to new customers. However, in retail settings, order pickers who work online (store staff) and offline (walk-in customers) are both active, and pickers have the ability to gather positional data about customers as they go through the store. A new Dynamic In-store Picking Routing Problem (diPRP) is presented here. There is a gap in the solution to the diPRP, though, as the interaction between in-store pickers and offline shoppers is still unknown. Though a lot of research has been done on picker routing issues, not much has been done on the special conditions that in-store picking teams face, particularly with the limitations imposed by the pandemic in 2020. Managing store capacity limits was one of the new operational demands and challenges that this time period brought about. As a result, retailers may be compelled to efficiently direct customers through their shopping paths in order to speed up the shopping process rather than prolong customers' stays.

However, it is also important to note that some customers may not know the location of certain items or have a picking preference in a certain order which leads to a stochastic store state (it is not known in advance) meaning that the PRP can not solve the problem of how to find picking paths that do not disrupt the customer experience. Because of that, Neves-Moreira and Amorim (2024) considered a picker that picks orders from BOPS and SFS at the same time walk-in customers do their own picking and modelled the dynamic PRP as a MDP that is solved with a Q-Learning approach - a RL value-based technique that seeks to represent the future expected reward of taking an action in the current state. The picker's goal is to perform the route in the most efficient manner (finding optimal paths) while avoiding customers. Different RL parameters were tested in different sized synthetic stores (bigger stores take more episodes to converge) and reached picking policies that reduced the frequency of interactions with customers while still maintaining a robust order fulfillment rate.

This dissertation approaches this problem by solving the routing problem according to a shortest path problem - the first product to pick is the product that is nearest to the beginning node and from that point on the next product to pick is the the product that is nearest to the previous one - in which between two given positions the agent has the decision power to choose the route it is taking according to the weight of different factors. The shortest path problem approach appeared to be well suited for this problem as it limits the decision power of the agent by enforcing the product picking sequence which ensures a certain level of operational optimization and facilitates the learning process by removing one decision making layer. This introduces the dynamism as the same shopping list does not necessarily result in the same course of action in two consecutive situations.

2.5 Reinforcement Learning

RL differs from better known machine learning algorithms like supervised learning - it is about picking up knowledge from a training set of labeled examples that a knowledgeable outside supervisor has provided - and unsupervised learning - which usually focuses on uncovering structure within sets of unlabeled data. RL is about the interaction with the environment while an agent tries to balance exploration - trying new unexplored actions - and exploitation - repeating actions from the past that have worked well - being the closest machine learning form to the kind of learning that human beings do. It is important to consider elements like a policy - a mapping of what actions to take in certain states of the environment -, a reward signal - defines if a certain action was good or bad for the agent - and the value function - represents the long term value of a given state as in the total reward that can be accumulated over the future. (Sutton and Barto, 1998)

Subsection 2.5.1 provides an introduction to what is RL. Subsection 2.5.2 presents some examples regarding RL in the supply chain context. Subsection 2.5.4 regards a RL technique to help the learning process of an agent, transfer learning.

2.5.1 Introduction to Reinforcement Learning

Humans may find RL instinctive because it mimics how they naturally pick up new skills. According to evolutionary biology, this type of learning is essential. Humans learn appropriate behavior based on reactions from their surroundings as early as infancy. When a baby has its first interaction with a pet, for example, the baby may touch it out of curiosity. After multiple interactions, the baby gradually learns appropriate behaviors. As such, this is an iterative process that is shaped by incentives and penalties, constituting a fundamental pedagogical approach. There is still the influence of family and society that promote RL throughout life shaping the actions of a person even more. RL is a machine learning algorithm that allows an agent to learn from its experiences and actions by providing feedback in the form of rewards and punishments according to interactions with an environment. Three main components of RL can be identified as being the states - what the agent observes after each action it takes -, action - choice the agent takes after observing the environment - and reward - the compensation, if it is positive, or penalty, if it is negative, the agent receives in order to learn about its actions - that interact with each other as shown in Figure 2.2 in which the agent takes an action making it interact with the environment which in turn provides the agent with a reward and a new state to observe. (Memarian and Doleck, 2024)

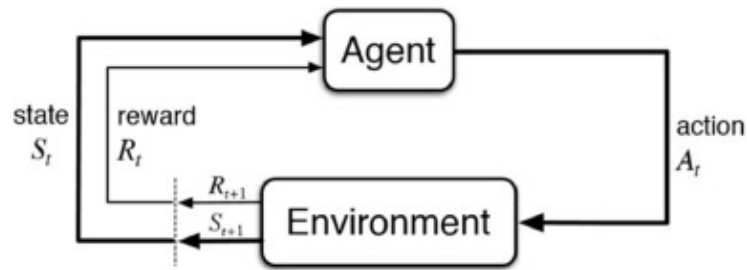


Figure 2.2: High-level block diagram of RL. (Memarian and Doleck, 2024)

RL can be classified following two different methods mostly: model-free RL - the transition dynamics and the reward function are not modelled and so the optimal policy learning process is done without them - and model-based - a learned or given model of the MDP shapes the optimal policy learning process. (Zhu et al., 2023)

Q-learning and SARSA are traditional RL algorithms studied as tabular methods and have succeeded when applied to MDPs with finite and discrete state-action spaces. (Zhang et al., 2022)

2.5.2 Reinforcement Learning in the Supply Chain Context

Zhou et al. (2023) used RL to solve a dynamic vehicle routing problem with stochastic demands from a real world situation. The procedure includes configuring a simulation in which an agent, or courier, interacts with a environment that is constantly changing conditions. The courier receives payment for fulfilling delivery and pickup requests in which deliveries, and especially those made early in the day, are rewarded with larger payments. A realistic environment is established as the simulations generates random pickup requests and service times. Changes in the state, such as the number of requests left, the new location and the time, result from the decisions the courier takes about where to go next. Throughout the morning operations, the agent engages in continuous interaction with this environment, taking actions that result in new states and rewards within a finite time horizon (morning operation).

Huang and Tan (2021) used deep RL to capture the decision making ability of RL as well as the perception ability of deep learning in a supply chain order management problem in order to extract the optimal strategy recommendations. The problem considers the order inventory price name, order shipping process, order cycle and order processing process. The authors achieved improvements in the warehouse turnover rate by obtaining a maximum increase of 28% in the fourth quarter and a minimum increase of 8% in the second quarter.

Rolf et al. (2022) identified that RL is commonly used to solve optimization problems targeting a specific supply chain driver such as customer-driven replenishment or supplier-driven replenishment.

However, due to the curses of dimensionality the problem can become too complex to be solved solely with RL leading to deep RL.

2.5.3 Deep Reinforcement Learning

Deep RL is the combination of deep learning with RL. It is obtained when using deep neural networks to approximate components of RL like the value function, policy and model. Deep distributed representations overcome the exponential challenges of the curse of dimensionality by taking advantage of the hierarchical composition of factors in data (Li, 2017).

Some deep RL algorithms include:

- *Proximal Policy Optimization (PPO)*: it was introduced by Schulman et al. (2017) as a new family of policy gradient methods (methods that work as a stochastic gradient ascent algorithm with an estimator of the policy gradient) for RL that switch between using stochastic gradient ascent to optimize a surrogate objective function and sampling data through interaction with the environment;
- *Deep Q-Networks (DQN)*: it was introduced by Mnih et al. (2013) being the first deep learning model to use RL to successfully learn control policies directly from high-dimensional sensory input, a convolutional neural network trained with a variant of Q-learning with stochastic gradient descent to update the weights and experience replay;
- *Asynchronous Advantage Actor Critic (A2C / A3C)*: it was introduced by Mnih et al. (2016) and relates to the execution of multiple agents in parallel on multiple instances of the environment replacing experience replay and does not require massively distributed architectures which means fewer resources are used.

According to Andrychowicz et al. (2018) PPO works well with discrete action spaces and the hyperparameter tuning process requires less computational efforts. Schulman et al. (2017) showed that PPO performed better than A2C on several domains. Kozlica et al. (2023) compared PPO and DQN in a discrete state-action space and found that PPO outperformed DQN and based on the training metrics that are presented the PPO algorithm showed much earlier convergence than DQN - which translates into a more stable algorithm. That led to using PPO in the development of the project.

2.5.4 Transfer Learning

Zhu et al. (2023) referred to transfer learning in RL as a method that leads to learning an optimal policy for a given target domain by leveraging exterior information from a given set of source domains as well as interior information from the target domain. The authors organize a given transfer learning approach by answering five questions: "What knowledge is transferred?", "What RL frameworks fit the transfer learning approach?", "What is the difference between the source and the target domain?", "What information is available in the target domain?" and "How sample-efficient is the transfer learning approach?". Then, they move on to a practical example using *HalfCheetah*, an OpenAI environment in which the goal is to train an agent with two legs to run fast and do not lose control. The research found that the target agent can learn from the behaviour of a

pre-trained agent. Some other transferable knowledge includes the model dynamics (RL agent can access a model of the dynamics for the source domain that is also partially relevant for the target domain).

This dissertation contributes to the field of RL by employing PPO in solving a problem not typically tackled with RL algorithms and leverages a transfer learning technique of using a pre-trained agent to boost the learning process.

Chapter 3

Problem Description

In this Chapter the aim is to describe the problem that this dissertation is tackling. That problem is modelled as a MDP since it considers elements like the states, actions, transitions and rewards at each step of the process and regards a retail store setting that fulfils both online and physical demand, given that the store receives orders from the online channel as well as physical customers can go in the store. In the store a picker is being controlled and aims at picking products to fulfil online orders while at the same time avoids physical customer encounters in order to preserve their shopping experience.

In Section 3.1 the store environment is discussed and what assumptions were taken into consideration in its simulation. Section 3.2 goes over the store dynamics such as how the physical customers and agents picking paths are computed. Section 3.3 describes the problem as a MDP going over the states, actions, transitions and rewards.

3.1 Store Environment

The store environment is one in which two kinds of demand are considered: demand from the online channel and demand from the physical channel. This creates the need to do store-based picking without undermining the customer's experience while physically shopping which can happen if the aisle passages are blocked by a picking cart or if the customers feel like they are on a warehouse instead of a store.

The store is modelled as a graph in which the nodes or vertices correspond to locations inside the store - such as the entrance, item and exit nodes, considering that the exit nodes include proper exits as well as tills - and each has a set of two dimensional coordinates assigned. In order to mimic the environment a simulation was made that takes into account several assumptions:

- Unlimited stock: no item is ever considered out of stock and subsequently no replenishment takes place in the store;
- Store and node capacity: the store has a maximum number of physical customers allowed inside it, C , as well as in a given node, m ;

- Walk-in customer arrival rate: the physical customer arrival rate follows a Poisson distribution with a mean number of clients per time period of $\lambda^{\text{walk-in customers}}$;
- Walk-in customer behaviour: the customer c goes through the store at a given speed $speed^c$ while following a certain shopping path θ_c and takes a certain amount of time picking a product while at the target location;
- Shopping path θ_c : the path followed by the customer is based on picking a certain list of products selected randomly and travelling the minimum distance between every two products;
- Online orders arrival rate: the online orders arrival rate follows a Poisson distribution with a mean number of orders per time period of $\lambda^{\text{online orders}}$;
- Online order picking positions: an order has a list of picking positions, L_o , that the picker needs to visit to retrieve the product.

3.2 Store Dynamics

The decision making that creates the problem under evaluation is essentially how the pickers decide to travel through the store aisles while considering factors such as the number of potential customer encounters as well as the distance travelled.

When an online order arrives it is assigned to an available picker that will be in charge of performing the picking process of the listed products. After that happens, two main processes take place: the sequence in which the products will get picked and the path the agent takes from its current location until its target location.

Regarding the product sequence, several factors can be considered, but in this case the distance between products will be the deciding factor, which leads to the product sequence decision consisting in solving a shortest path problem that aims the minimization of the distance the agent travels.

However, the dynamism of the problem relates to the power the agent has to choose its own path between two locations inside the store. Despite not being allowed to decide the product picking sequence, the picker can decide on the path taken to reach its target location. As such, the goal is to strike the right balance between travelling the minimum distance and avoiding the maximum number of physical customers doing their own product picking. Sometimes, it is worth travelling a longer distance considering it will allow avoiding a large number of customers while other times the number of customers the agent will encounter in a position is not high enough to justify a higher distance travelled.

In each position the picker observes its surroundings and collects information about the number of customers in the current position as well as in the neighbour visible positions.

The previously described dynamics are represented in Figure 3.1 as the environment deals with online orders and physical customers arrival rates while facing some restrictions. The physical

customers execute their shopping path while the agent does the product picking making it possible to verify that in some situations travelling a higher distance pays off because it leads to avoiding crowded positions. In other situations the number of customers present in certain locations is not high enough to set the picker in a different and less efficient course.

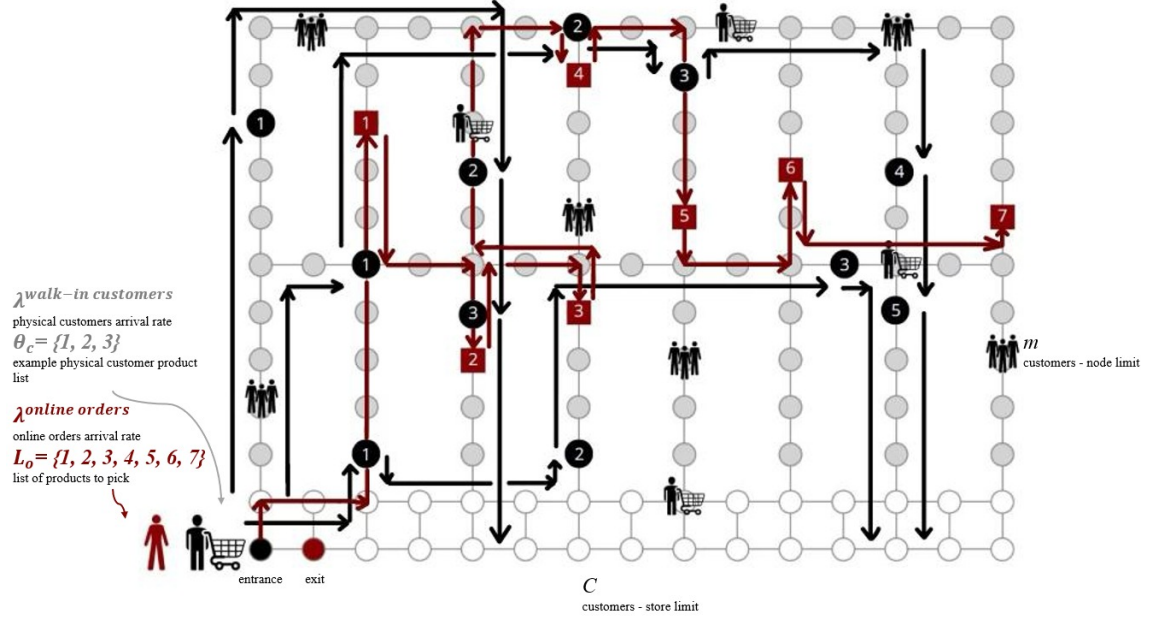


Figure 3.1: Store environment that represents the picker and customers doing their picking paths.

3.3 Modelling the problem as a Markov Decision Process

Modelling the problem as a MDP with T time periods was inspired in the work of Neves-Moreira and Amorim (2024). It requires the consideration of decision epochs $k \in \{0, \dots, T\}$. Each epoch corresponds to a state s_k and in each one the agent needs to decide what is going to do next - it needs to take an action. At epoch $k = 0$ the picker is at the starting node and at the end of the process, epoch $k = T$ the picking tour ends and the agent goes to the finishing node. New epochs are triggered upon entering a new node in which the agent collects information pertaining the state it is in. In turn, that information will guide the decision making - which action a_k to take - while the picker tries to travel the smallest possible distance and minimize customer encounters.

Typically, a MDP contains four main elements: a set of states, a set of actions, a reward function and transitions.

States

The states relate to what is perceived by the agent. In decision epoch k the system is characterized by the state s_k , a tuple $s_k = (cp_k, tp_k, d_k, n_k(1), n_k(2), n_k(3), n_k(4))$. cp_k and tp_k are the current position's coordinates and the target position's coordinates, respectively, d_k is the distance that separates the current and target locations and $n_k(1), n_k(2), n_k(3)$ and $n_k(4)$ are elements that

represent the number of physical customers nearby the agent's current position in four different directions as long as they are within visible range. Initially, the agent begins in the store entrance, $cp_0 = (0,0)$ while the other components depend on the given situation, such as the arrival rate of physical customers - which will lead to either higher or lower physical customer sightings - and online orders arrival rate - which will dictate the tp_k and d_k components not only in the beginning but at all times.

According to the state space's different components it has a total of nine dimensions and the size of each dimension will affect how large the state space is.

Actions

In order for the picker to go from its current position to a new position while travelling to reach its target position, it is necessary to take an action a_k from the action space $\mathcal{A}$. Depending on the agent's current location there is a set of possible arcs from which to choose that will lead to a new location. Based on the graph upon which the simulation is based the agent has a limit of four valid actions that may be less in some locations that provide access to fewer new locations, such as a node that is in a corner and is connected to only two other nodes. If the agent takes an action that is not possible or valid - choosing to move in a direction that is not contemplated by the current node's connections - it remains in the same position instead of moving to a new one. In short, each action represents a direction the picker takes and, as such, the transition from the current location to another - s_k to s_{k+1} - if that direction is valid considering the node connections to other nodes.

Transitions

After selecting the action a_k the state transitions to a new state s_{k+1} . While transitioning between states, the agent already knows where it is going, but only when it reaches a new state information about the number of customers it is going to encounter and is visible to is revealed, ω_{k+1} .

Rewards

Given a certain transition from s_k to s_{k+1} , the picker receives rewards depending on the conditions of the following state. The reward function considers four different elements: a negative reward for each step the picker takes, w_1 , a negative reward for each physical customer the picker encounters in its current node, w_2 , a negative reward for each physical customer the picker is visible to, w_3 , and a positive reward for each product the agent picks, w_4 . The number of steps the picker takes is given by ϕ_1 , the number of customers present in the new location is ϕ_2 , the number of customers around the new location is given by ϕ_3 and the number of products is given by ϕ_4 . Equation (3.1) describes how the reward r_k is computed based on its dependence on the current state s_k , the action that is taken a_k and the new information that is revealed ω_{k+1} through the transition that is triggered.

$$r_k(s_k, a_k, \omega_{k+1}) = -w_1\phi_1(\omega_{k+1}) - w_2\phi_2(\omega_{k+1}) - w_3\phi_3(\omega_{k+1}) + w_4\phi_4(\omega_{k+1}) \quad (3.1)$$

In addition to that reward function behaviour, another factor was considered which is a redundancy factor, which essentially accounts for the picker's sub-optimal habit of staying in the same node or continuously moving back and forth. Within the simulation environment, the last two picker positions are stored and every time the picker is in its current position and goes to one of those a penalty is applied. There might be times in which doing that follows the optimal picking policy - the picker may be trying to avoid customers or doing the shortest path and to achieve that it is possible that reaching a position and then going back is the best decision meaning that the penalty of doing that should be lower than the penalty of encountering customers or doing a longer path -, which is why that behaviour is allowed. However, to prevent the unrealistic behaviour of an agent being encouraged to stay in a given node or around for longer than would be reasonable the penalty was introduced.

Chapter 4

Methodology

In this Chapter the aim is to present how the defined problem was tackled in order to obtain an optimal picking policy compliant with the defined goals. Section 4.1 reflects upon the dimensionality issue that arises with high dimensional state spaces and how it can impact traditional RL algorithms leading to the adoption of mitigation strategies. Section 4.2 describes the algorithm that was used in the project to achieve a picking policy and provides an example of its outcome. Section 4.3 is about the features that were thought of and implemented as a way of driving the learning process in an efficient manner. In Section 4.4 the strategy to achieve optimal sets of hyperparameters for the specific project is described. Finally, Section 4.5 goes over the use of parallel processes to efficiently train the agent and get better outcomes.

4.1 The Problem of Dimensionality

In the context of a RL problem modelled as a MDP, dimensionality refers to the number of variables or elements that characterize each state along with the number of possible values each variable can take as well as the number of actions available to the agent. A simple representation of the concept in the context of the current problem can be achieved by considering the state s_k and the variables that characterize it, the target position coordinates variable tp_k , the current position coordinates variable cp_k , the distance between the current and target locations d_k and the four variables that relate to the number of customers surrounding the agent $n_k(1), n_k(2), n_k(3)$ and $n_k(4)$. tp_k and cp_k are bi-dimensional with as many values as there are nodes representing items inside the store, I , in the case of the tp_k , and with as many values as there are nodes in the store, N , in the case of cp_k . The distance is fixed from the moment tp_k and cp_k are defined. Each element among $n_k(1), n_k(2), n_k(3)$ and $n_k(4)$ can take as many values as there possibilities for the number of customers around the agent in each direction, M . This leads to a state space dimension of $I \times N \times M^4$. Introducing new variables can make the state space increasingly large. Understanding dimensionality is important in order to tackle the problem in retail store environments as it impacts the efficiency and feasibility of RL since with increased dimensionality the computational efforts required to extract policies and make decisions also increase which makes it harder to train

real world scenarios where quick decision making is crucial. The trade-off between exploration and exploitation is also important to consider as it becomes harder to ensure exploration of the environment in higher dimensions leading to the agent learning sub-optimal behaviours.

4.1.1 Curse of Dimensionality

The curse of dimensionality was a term first introduced by Richard Bellman (Powell, 2007) and it describes the exponential effects that increasing data dimensions has as well as the increased computational efforts required to process the data and extract satisfying results.

By considering the previous example, assuming the fact that retail stores have different sizes, one store could easily have twice, three or four times the size of another store which could lead to a proportional increase in the number of item nodes I and nodes in general N as well as the usual flow of physical customers visiting the store which would increase the number of customers around the agent M in each of the four directions. Considering the three mentioned scenarios it is shown that with the increase of the store dimension the state space grows exponentially:

- Store twice the size: $2I \times 2N \times (2M)^4 = 64I \times N \times M^4$;
- Store triple the size: $3I \times 3N \times (3M)^4 = 729I \times N \times M^4$;
- Store quadruple the size: $4I \times 4N \times (4M)^4 = 4096I \times N \times M^4$.

The previous example considered the variable M increase proportional to the store size increase for simplification and demonstration purposes.

This shows that increases in the dimensions of the variables that make up the state space increase exponentially the dimension of the state space itself. Adding new variables to the state space has a similar effect. By considering variables with D possible values, if one is added to the previously defined state space its size would increase D times while if two were added its size would increase D^2 times.

As such, the computational efforts required also increase exponentially. There is increased efforts as the larger the environment the larger the state and action spaces become increasing the memory needed for their storage. In the case of a large retail store environment, storing a policy that accounts for all the state action combinations is demanding from a memory point of view. Updating the value estimates or the policies will require calculations and modifications across high dimensional arrays which creates the need for higher processing power.

Dimensionality also has its effects on convergence. Convergence is how the algorithm progresses over time to stabilize and reach the a solution - this essentially means that as the number of episodes that occurred increase, the changes in the policy or value estimates decrease. A policy extracted from a RL algorithm that converges will carry out the best policy from its learned experience. In high dimensional environments, as is the case of a retail store environment, achieving a thorough exploration of the state space is challenging because visiting the same state action pair becomes less likely as the state space dimension increases. That leads to sub-optimal behaviours in which the agent performs better while executing the less common more visited state situations

and poorly in the rest of them. From a perspective with a limited number of training episodes, convergence would be harder for a higher dimension environment and maybe would not even happen, and from a perspective with an unlimited number of training episodes, convergence would be slower for a higher dimension environment, but with more training episodes arises the need for more computational power. Overfitting to regular patterns as well as variance in the followed strategy become a source of concern.

4.1.2 Impact on Traditional Reinforcement Learning

To better understand the effects of dimensionality on traditional RL algorithms it might be useful to dive deeper into one, Q-Learning, that is better explored in Appendix B.

Equation B.2 from Appendix B relies on the Bellman equation based on a Q-function that is given by Equation 4.1:

$$Q(s, a) = R + \gamma \max_{a'} Q(s', a') \quad (4.1)$$

Every time an action is taken the algorithm executes an update to the value-function. The value-function can be perceived as a Q-table that represents the value for each state-action pair - initially, as the table needs to have values to start with, there is a random assignment of values.

The dimension of the Q-table essentially defines the computational effort that Equation 4.1 demands and as the state space grows so does the Q-table. That growth leads to a very large number of values that have to be stored and constantly accessed and updated whenever an action is taken.

Given a high dimensional state space as it was previously mentioned it is very likely for an agent to learn sub-optimal behaviours by simply repeating the paths that it knows work. That would lead to several values in the table being left with a very small number of updates. If a high number of values remain very close to the initial values the extracted policy will highly depend on those which is almost equivalent to following a random policy.

As the table initial values are assumed to be far from its true values, the algorithm faces a big challenge with convergence. That can have a very slow progress when new states are constantly being encountered as would be the case of a real-world retail store environment.

Considering how dimensionality can affect RL traditional algorithms, even reaching a point that makes them impractical, some mitigation strategies come up that include employing strategies that ensure a thorough exploration, such as ϵ -greedy approach that makes the agent take a random action upon a given probability so that it does not always choose the highest value one and keep a very high focus on exploitation or the upper confidence bounds approach in which confidence intervals are computed around the values estimates and the agent takes the action with the highest upper confidence bound leading to the highest value action or one whose value is very uncertain (Kumaraswamy et al., 2018).

Another valid strategy is taking the leap to deep RL and leveraging the use of neural networks which can significantly reduce the need to estimate a value for every possible state action pair by identifying patterns in similar states and approximating their values.

Among DQN, A2C and PPO, some common deep RL algorithms, the latter was chosen.

4.2 Proximal Policy Optimization

PPO is a deep RL algorithm - it combines deep learning with RL. This allows the use of deep neural networks which approximate functions that are potentially hard to compute. PPO was introduced as an upgrade to Trust Region Policy Optimization (TRPO). TRPO is a more complicated and less scalable method. (Schulman et al., 2017)

This method is an actor-critic method as it combines a policy network that guides the action taking as well as a value network to estimate state values.

PPO is characterized by being:

- Model-free: the agent learns based on the experience built upon interacting with the environment - the next state is not predicted based on the current state - which is important in the context of a retail store environment because it is stochastic with factors like unpredictable customer movements;
- On-policy: the policy the agent uses to guide its behaviour is the same policy that is being optimized - the policy updates are based on the same values that that policy is generating in its current interaction with the environment - which helps the agent continuously refine its actions in a retail environment where changes are constantly happening and quick adaptation is crucial.

This algorithm is a policy gradient method. Policy gradient methods are the ones that focus on parameterizing and optimizing the policy itself - mapping from states to probability of taking a given action - instead of optimizing the value function that will after lead to choosing the highest value action and define the policy (value-based methods). In a large state space as a store this helps with exploring the different states initially when lower value actions can lead to better outcomes and keeps the agent from fully stopping the exploration of new courses of action even when the learning process is reaching the end. This is important as a store is a very dynamic environment and new valuable states can be found at every stages. These techniques are effective, but they frequently run into problems regarding the policy stability and efficiency - specially in situations in which rewards vary a lot. PPO is an innovation comparing to fundamental policy gradient structures focusing in overcoming these issues as it adds more complex procedures to control policy changes and make sure the learning progress remains stable. Otherwise, the agent would go from episodes in which it picked products successfully to another where it was not able to pick products or where product picking was due to random movements.

This methods work through the execution of an estimator of the policy gradient in a stochastic gradient ascent algorithm. Equation 4.2 represents the most commonly used gradient estimator

form for PPO.

$$\hat{g} = \mathbb{E}_t [\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \hat{A}_t] \quad (4.2)$$

where:

- $\mathbb{E}_t[\dots]$ is the average computed over a certain amount of time-steps;
- $\nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$ is the gradient of the logarithm of the stochastic policy function π_{θ} , considering its parameters θ , in which the policy provides the probabilities of taking action a_t in state s_t and the logarithm transforms multiplications between the policy probabilities into sums of logarithms;
- $\hat{A}_t$ is the estimator of the advantage function at time-step t - which is essentially the difference between the future expected rewards after taking action a_t in state s_t , action-value function, and the future expected rewards if the current policy is followed, value-function.

Unlike value-based methods, policy gradient algorithms naturally develop stochastic policies, leading to smoother search spaces and addressing the exploration challenge by enhancing environmental knowledge - easier due to probability distribution over actions - for policy optimization. These methods facilitate gradual modifications in the policy throughout the learning phase, potentially improving convergence characteristics as updates directly influence the policy by small amounts, which reduces the likelihood of drastic changes which would lead to unstable learning (Sutton et al., 2000).

Policy gradient methods use function approximation to model the probability of taking any action. This makes them more suitable for high dimensional state spaces in which exploration is needed - action choosing based on probabilities facilitate it.

In PPO, the policy gradient estimator plays an important role in figuring out the optimization of future rewards policy parameters adjustment. It computes gradients that indicate how adjustments to the policy's parameters θ can maximize the expected rewards guiding the learning process towards overall improvement of the policy as a whole.

4.2.1 Clipped Surrogate Objective

Before understanding what the clipped surrogate objective is about, it is important to know that PPO aims at the maximization of the surrogate objective (Equation 4.3).

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t [r_t(\theta) \hat{A}_t] \quad (4.3)$$

where:

- $L^{CPI}(\theta)$ represents the surrogate objective function - the algorithm aims at maximizing the advantage function weighted by the shown ratio with the final purpose of improving the policy's performance effectively and maximizing future expected rewards;

- *CPI* stands for Conservative Policy Iteration a founding algorithm of ADP and its fundamental concept is to moderate the greediness by using stochastic combinations of successive policies (Vieillard et al., 2020);
- $\pi_\theta(a_t|s_t)$ and $\pi_{\theta_{\text{old}}}(a_t|s_t)$ are the probability of taking a certain action a_t in state s_t under the current parameters of the policy and under the old parameters of the policy, respectively;
- $r_t(\theta)$ represents the ratio between the previously mentioned probabilities of taking action a_t under different circumstances.

However, maximizing that function without adding a constraint would lead to very big policy updates (Schulman et al., 2017) which would make learning less stable and create the need to stop the new policy from deviating too much from the previous one which is why the need for a clipped surrogate objective comes up that basically introduces a minimization within a maximization framework as it is shown in Equation 4.4. By introducing a clipping mechanism, the clipped surrogate goal potentially changes the policy gradient estimations.

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t [\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] \quad (4.4)$$

where:

- ϵ is a hyperparameter in the interval $[0, 1]$ usually set to 0.2;
- $\text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t$ is the clipping mechanism that is used and it changes the surrogate objective by clipping the probability ratio and thus making it stay within the interval $[1 - \epsilon, 1 + \epsilon]$ - then, the minimum value between the original and modified objectives is used leading to the conclusion that the clipping method is only active for situations in which the clipped objective is worse than the old one;

When the ratio takes values outside of the interval it is replaced by the value to which it is closer, $1 - \epsilon$ or $1 + \epsilon$. In the context of store navigation if the agent is performing well and suddenly has an incident in which performance has a large decrease, clipping makes it so that incident's influence in the policy is not high enough so that it undermines the learning process delaying it by creating the need for the agent to re-learn how navigate between two positions.

This also depends on the signal of the advantage function.

- $\hat{A}_t > 0$: the objective is to encourage actions that are better than expected, but not excessively better, so $r_t(\theta)$ will be clipped at $1 + \epsilon$ when it surpasses that value - focus on increasing $r_t(\theta)$;
- $\hat{A}_t < 0$: the objective is to discourage actions that are worse than expected, while not encouraging overly aggressive improvements, so $r_t(\theta)$ will be clipped at $1 - \epsilon$ when it goes below that value - focus on decreasing $r_t(\theta)$.

4.2.2 Value Loss Function

The value loss function (Equation 4.5) has influence in the advantage function estimator because it is used to update the value network in PPO - expected future return from a given state. Consequently, value predictions will be more accurate, which means the advantage function will be more accurate as well and it is used for updates in the policy.

$$L^{VF}(\theta) = \mathbb{E}_t \left[(V_\theta(s_t) - \hat{R}_t)^2 \right] \quad (4.5)$$

where:

- $V_\theta(s_t)$ is the value of state s_t predicted by using the value network with its parameters θ ;
- $\hat{R}_t$ is the real return that was achieved in state s_t computed as the sum of discounted future rewards.

The mean squared error is used in order to provide an incentive for smaller deviations as bigger ones will have a higher impact than lower ones.

The value loss function improves the accuracy of the advantage estimations by reducing the variance between expected and actual returns, which in turn improves the computation of the policy gradient.

4.2.3 Final Objective

The final objective that this method aims at maximizing (Equation 4.6) combines the policy surrogate objective and the value loss function as well as adds an entropy element with the goal of enhancing exploration.

In RL entropy measures the amount of exploration the agent does while being trained, which is essentially how diverse its actions can be to get to know new states instead of following behaviours that have worked in the past.

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right] \quad (4.6)$$

where:

- c_1 and c_2 are the value and entropy coefficients set to scale the value loss function and the entropy bonus, respectively;
- $S[\pi_\theta](s_t)$ is the entropy bonus used to incentive exploration and to which is given a certain weight through the assignment of a coefficient c_2 .

Through the signals of each element in Equation 4.6 it is possible to conclude that PPO aims at maximizing the clipped surrogate objective and the entropy and at minimizing the value loss function.

The architecture of PPO is intended to balance the dynamism of updates throughout its value and policy networks. The clipping of the policy updates reduces the probability of an aggressive

policy update. This works towards keeping the training process stable. In store environments where the conditions change so rapidly the aforementioned control helps guide the store navigation learning process. At the same time, the value loss function estimation constitutes an important beginning for calculating the advantage. That guides the direction of policy modifications. This is complemented by the entropy bonus, which makes sure there is proper exploration of the environment and helps stopping convergence to sub-optimal behaviours. Exploration is vital in a retail store environment as there is a high dimensional state space and the agent needs to be familiar with a high number of paths making it easy to fall into sub-optimal picking policies in case they show improvement upon previous behavior.

4.2.4 Example in the Problem Context

After training an agent in a retail store environment with physical customers doing their own product picking, a picking policy is extracted and by examining Figure 4.1 it is possible to reach the conclusion that the path followed by the picker is different from the the path if it was a shortest path policy. In the figure the nodes' colours and sizes represent an average density of customers in a given node during the moment the picker went from the entrance until reaching node 102 passing through node 69 - larger nodes with colours closer to red means that there are more customers in that node while smaller nodes with colours closer to green means less customers in the node. The sequence of nodes $[0, 69, 102]$ is mandatory while the other nodes the picker visited are arbitrarily chosen while the picker is trying to visit those nodes and at the same time travelling the minimum distance and encountering the minimum amount of customers. In some cases, the picker travels a higher distance in order to go through less crowded nodes.

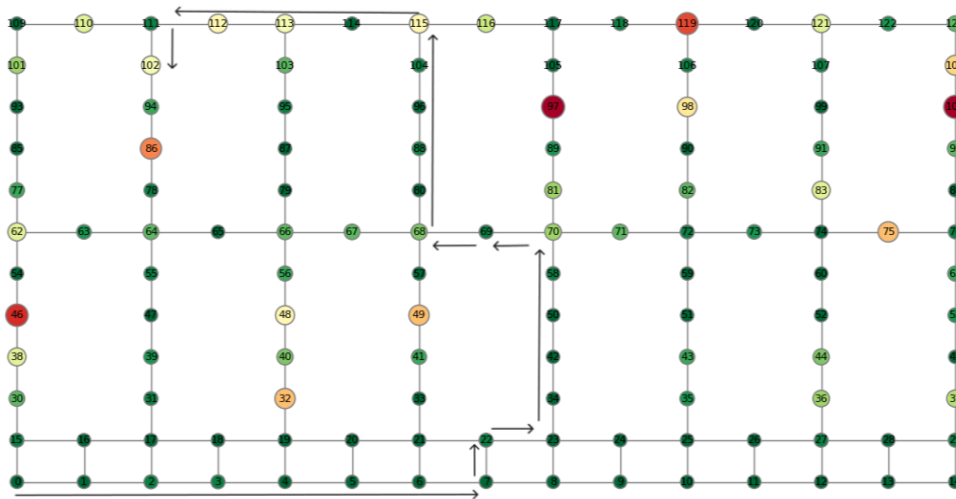


Figure 4.1: Example of the path taken by the agent in order to pick products in nodes 69 and 102 in that sequence.

4.3 Feature Engineering

The objective of the project is to extract picking policies that teach the agent how to go through the environment in a meaningful way - a way that promotes efficient travelling from one position to another, meaning the agent going through the minimum distance that weighs in the physical customers' shopping experience.

In RL the agent needs to observe the environment and learn from it and update the policy based on the reward it receives from taking an action under certain conditions. While this is putting it relatively simple, as there is more to updating a policy than the immediate return of an action since the future expected rewards are always taken into consideration, that lays out a need which is what information or observation should be provided to the agent such that it is relevant and at the same time not redundant - since unnecessarily increasing the state space would be a waste of computational efforts. The mentioned relevance refers to the information's potential to drive learning in an efficient way that does not lead to poor agent behaviour.

In this context, the features are the observations that are passed to the agent at each step it takes meaning that feature engineering in this case refers to the selection, modification or creation of different features using knowledge about the specific setting in which the agent is. The performance and convergence of the algorithm largely depends on how well the environment is represented through the states which constitutes the importance of feature engineering.

Retail stores can vary a lot regarding their size and layout, which means that the agent needs to be adaptable to different stores. Having that in mind raises the importance of generalization. The features that are used should not be features that are repeatable only on that specific environment, but rather features that help the agent learn actions that are valuable in different situations. Otherwise, overfitting may happen leading the agent to perform very well in certain tasks but poorly in tasks that vary slightly.

The focus of the feature engineering performed on this work was on the creation and modification of features.

Besides the features that follow, other features such as whether the agent had picked a product or not at its current position, providing the target direction and informing about the picking route were considered. However, along with the implemented features those features would either be redundant or useless. As the target position and current position features along with the reward structure already convey information relating to product picking, since it is considered that the product is picked if the picker is at the target position, and the target direction, as it is a direct extrapolation from those two features, providing those features does not add relevant information. Also, since the picker is not allowed to change the picking sequence it is not worth it to provide it with the picking route. As such, those features were not implemented after careful consideration of the trade-off between additional gains they could give against the additional computational efforts and learning difficulties the increase of the state space would bring.

- **Current Position and Target Position**

The current position and target position features were obtained by transforming the correspondent nodes from the graph in coordinates.

As the store is modelled as a graph the agent positions can be accessed at any moment, but it is given as node, like node 10. That raises the concern of node numbers being too specific of a given situation and do not exactly reflect the relative positions of two nodes - for example, in Figure 4.1 it is possible to see that node 109 is closer to node 0 than node 76, which is applicable to that store and may not be applicable to other stores.

However, coordinates are general, which means that in any store the position with a given coordinates will be the same as in other store. This means that if the agent learns how to navigate between two points by observing its coordinates in a certain situation it will be able to repeat that behaviour in a different situation - for example, the points with coordinates (0,0) and (4,2) will be equally distant from each other in any store regardless of its size or layout.

- **Target Distance**

The target distance is the measure of the euclidean distance between the current position and the target position.

The picker is walking towards a goal, but in the middle it has to go through nodes in between that do not return any positive reward which means it is important for the picker to get a sense of whether it is going in the right direction or not - if moving from one node to another makes the picker get closer to its target. As such, the distance from the picker's current position and the target position is computed and provided to the picker at every state.

- **Physical Customer Counts**

The physical customer counts feature counts the number of physical customers that are in the neighbour nodes to the current position providing a value for each possible direction.

One of the main goals to achieve in the picking policy that the project aims to extract is physical customer avoidance. Considering that it is important for the agent to know how many customers it is visible to from each possible direction it can take in the position it is in. That way the objective would be to learn how to avoid the more crowded paths. Towards achieving it, the count of customers to which the agent is visible in the four possible directions is provided as an observation - at some positions, the agent can only move in two or three directions so the physical customer count in the other two or one directions, respectively, will be zero, like when the agent is in one aisle and can only move through it and has only two possible actions or directions to take.

4.4 Hyperparameter Tuning

Hyperparameters are parameters used as an input to machine learning algorithms like PPO that will define different factors in the training process like the value that is given to exploration or the value that is given to future rewards.

Hyperparameters influence the model performance and can be the source of the model's success or failure as they are typically not updated during the training phase. For example, some hyperparameters can define what importance is given to exploring the existent state action pairs or to long term rewards over short term rewards. As opposed to model parameters that are learned internally and can not be configured as they change while the training is executed.

That raises the issue of hyperparameter tuning which is optimizing the set of hyperparameters that is used in the initialization of the model in order to get the best performance out of it. The issue with that optimization is that the hyperparameter space of a model can be very large and include continuous variables as well as discrete ones which creates a large set of combinations to be found, leading to an extensive and prolonged process.

As many other machine learning algorithms, PPO faces the common issue of balancing the trade-off between exploration and exploitation, which can be improved by properly tuning its parameters.

Convergence is essential to achieve good results and having a training process that is stable and efficient helps with model convergence. Certain hyperparameters limit the policy updates preventing from occurring large updates that can destabilize the training process, so if they are optimal it can help converge the algorithm.

Each environment has its own specific tasks and properties, making exploration more important in certain environments than others, for example, and that means it is hard to reach one set of hyperparameters that follows a "one fits all" approach. That highlights the importance of adapting hyperparameters to the specificities of the environment.

Hyperparameters can also play a role in the computational efforts required meaning that there may be a set that works well for an environment, but also another set that works equally even better and achieves better computational performance.

Overall, the final expectations of doing hyperparameter tuning is to find a set that will guide the training of the model towards an optimal picking policy as fast and efficiently as possible.

4.4.1 Hyperparameters Tuned

Given the time constraints relating to training models in complex environments, the hyperparameter tuning this project is focused on does not include all possible parameters the PPO algorithm can take. Since the default parameters the algorithm uses are already obtained through some optimization and are supposed to provide an average performance for a given model, the main goal was to tackle the hyperparameters that seemed to have a greater influence on the more relevant factors that influence the specific problem in the PPO setting such as the exploration and the importance of long term rewards.

- **Learning Rate (α)**

The *learning rate* defines the size of the step the algorithm takes during the learn phase in order to reach the optimal policy. While aiming for optimal solutions and escaping local optima the algorithm takes steps with a certain size that should not be so small that it extremely increases the complexity of the training process and should not be so large that it stops the algorithm from reaching the optimal policy. As an example, if someone is trying to reach the highest point in a field while taking steps of a certain size if the steps are too big the person may skip the highest point while if the steps are too small it will take a lot of time to reach that point. Ideally, it is important to explore the state space in the beginning and decrease the exploration towards the end. However, it was chosen to balance the number of sets of hyperparameters that were used to find the best sets against the amount of time dedicated to testing each set which made finding feasible initial and final learning rates as well as an ideal decrease rate a difficult task considering that scalability issues due to the different number of set testing episodes and training episodes that would arise. As such, implementing a scheduling to the learning rate remains as a possibility of future work to be done in the project.

- **Discount Factor (γ)**

The *discount factor* (γ) relates to the weight that is given to long term satisfactory results - expected future rewards - in the moment the agent is making its decision. The higher it is - given that it falls within the range $[0, 1]$ - the more importance is given to future rewards as opposed to immediate ones. In a retail store environment the agent will only receive its reward when it gets to the target location, so it is important that it is able to go through some penalization in order to achieve the final reward that may be after being penalized 20 times for example. It has to be able to take immediate steps that will only get penalization so that it reaches the target location that will result in a positive reward.

- **Clipping Parameter (ϵ)**

The *clipping parameter* (ϵ) relates with the prevention of large policy updates in order to maintain training stability and help with the model convergence. It stops the ratio of the new policy to the old policy from being outside a certain interval $[1-\epsilon, 1+\epsilon]$. Larger clipping parameters will lead to larger allowed updates to the policy. In order to maintain consistence in the movements within the store and to prevent from the agent's movement patterns being constantly changing while aiming for an optimal picking policy that fulfils the goals of the project it is important to tune the clipping parameter.

- **Entropy Coefficient**

The *entropy coefficient* controls the trade-off between the exploration and exploitation as the higher it is the more exploration it encourages. It is crucial to prevent the model from converging too soon into a sub-optimal policy. Given the dimension of a retail store it is

important that the picker explores enough but does not get lost in all possibilities to explore while never reaching an optimal policy.

- **Batch Size**

The *batch size* is an important hyperparameter as it influences both the computational efforts during the training as well as the stability given that it defines the number of data samples that are used to update the policy - more stability comes with higher batch sizes which also bring more memory usage, while faster convergence is related to smaller batch sizes as well as more variance in the updates.

- **Number of epochs**

The *number of epochs* refers to the number of times the collected trajectories performed by the agent are processed by the algorithm during the training meaning that it can help and harm the performance as it can improve the learning phase but also lead to overfitting.

- **Network Architecture**

The *network architecture* reflects how the neural network is structured in the deep learning model that is used in PPO. That hyperparameter plays a big role in feature representation since it defines how the input information is processed and turned into meaningful information which in a retail store environment with a high dimension is important as that is essential for the agent to learn the optimal policy. A well structured network architecture can also help with transfer learning since it helps the algorithm reach generalization and better adapt in unseen states. If a model is trained in a small store and then retrained in a medium store generalization can help the agent reach better results faster without requiring extensive training. It also influences function approximation and the more accurate it is the better performance the algorithm will have. Finally, the complexity of the neural network also influences the computational efforts required.

4.4.2 Hyperparameter Tuning Strategy Employed in PPO

There are several strategies that can be used to optimize the hyperparameters. In this project the Optuna optimization framework was used - a recent optimization software that aims at simplifying the tuning process by solving some problems that existed in previous software that could easily lead to a waste of efforts in the tuning process (Akiba et al., 2019). The tuning is made based on trials and each trial is essentially the choosing of a set of hyperparameters and evaluation of its performance through an objective function that in this case is the mean reward obtained with that set of hyperparameters. Optuna leverages the information collected in previous trials.

The sampling method that was used was the Tree-structured Parzen Estimator (TPE), a Bayesian approach. It uses a combination of two Gaussian processes to model the objective function in which one relates to how likely it is to improve the performance and the other how likely it is to worsen the performance. The sampling decision TPE does is then based on the ratio of those two

probability density functions which will guide the search towards hyperparameters that will get the best performance.

This method is more scalable than others like Grid Search or Random Search as it gets feedback from previous trials which helps navigate larger hyperparameters spaces avoiding redundant and harmful decisions and improving efficiency. (Falkner et al., 2018; Belete and D H, 2021)

4.5 Parallel Environments

Dealing with a deep RL training method for a complex high dimensional environment creates challenges related to the computational efforts required as well as achieving a satisfactory policy for the specific problem that is being tackled - like the highly computational complexity, large amounts of data are needed, training stability and convergence to optimal policies. The goal is that the picking policy creates a distribution of probabilities for each action such that it allows the agent to achieve the optimal balance between avoiding customers and travelling the shortest path while picking orders in a retail store - this creates a lot of information that needs to be handled efficiently in order to escape sub-optimal policies and drive the agent towards a realistic behaviour. Given this context, sample efficiency becomes a problem - the collected experiences throughout the training are hard to obtain given the high dimensionality of the environment and the high amount of experience the agent needs to generate. Overcoming that obstacle requires a long training process and that is where parallelizing the environments comes in.

In the context of RL algorithms, parallelism means the simultaneously training of different environments in order to achieve results faster and more efficiently as there are multiple iterations of the environment running at the same time sharing the learned experiences with each other. (Nair et al., 2015)

Towards implementing parallelism in the project the tool *SubprocVecEnv* from the *stable_baselines3* library in Python was used. It was used to facilitate the creation of multiple instances of the retail store that was used and each one runs on its own sub-process enabling them to generate their own data. Only the number of environments were specified.

This speeds up the process as there are multiple instances running and gathering different experiences at the same time that can be batched and together contribute to the overall picking policy. This requires communication between processes that is handled by Python's *multiprocessing* library. This library has several mechanisms to control data sharing between processes that allow data to be shared amongst processes without a process interfering with the other - as they run independently. That allows scalability for computationally demanding environments.

The used tool plays a large role in the task learning exercise that is training an agent in a high dimensional space.

There is another tool to achieve parallelization which is *DummyVecEnv*. However, it runs all the environments in a single process which does not make full use of multi-core processors and results in a slower process and does not achieve true parallelization because the training is sequential with multiple environments instead of parallel.

Chapter 5

Results

The following Chapter presents the main findings of the proposed approach. Section 5.1 regards the creation of synthetic stores and what was considered in doing so. Section 5.2 presents the best sets of hyperparameters that were found for the different store configurations. Section 5.3 presents several results regarding algorithm convergence and performance metrics during training and testing. Section 5.4 is about the retraining of an agent in a more complex environment and the outcome of that. Section 5.5 regards the implementation of the devised strategy in a real world scenario.

5.1 Creation of synthetic stores

The models were trained in stores with different sizes in order to compare metrics and the model convergence between the stores as well as the training progress across a certain number of episodes.

Three different sizes were considered:

- **Small Store:** as it is shown in Figure C.1 from Appendix C it has five aisles and no side aisles;
- **Medium Store:** as it is shown in Figure C.2 from Appendix C it has seven aisles and two side aisles;
- **Large Store:** as it is shown in Figure C.3 from Appendix C it has fifteen aisles and two side aisles.

The length between different positions or nodes in the graph - the edge length - is the same among all stores while the aisle length differs: the aisle length of the small store is half the length of the large store aisle's length and the medium store aisles are two thirds the size of the large store aisle's length. The choice to use three different store sizes serves as way to validate the approach under varied circumstances and represents stores with realistic sizes that could go from a boutique store to a supermarket - while the layouts are not as realistic given their regular shapes - allowing

to verify the scalability of the model. Also, it provides insights into the impact of the layout size in the operational performance of the picker.

Regarding the simulation configuration - what was used to simulate customer paths in the stores and other important details to approximate real life environments - the same one was used in the training of the different stores. Several variables were considered in configuring the simulation such as the online order arrival rate that was set to 0.2, the physical customer arrival rate that was set to 2, the maximum number of customers in the store as well as in each node which were considered as 50 and 5 customers, respectively, the picker speed which was set to $1m/s$ and the product picking time was set to 15s. The number of products in each online order is uniformly distributed between 1 and 25 items. The store operation time is 8h. These values aim at achieving a realistic simulation and one that focuses the computational efforts in getting the agent to learn the store navigation by using a relatively low product picking time - operation time is more used in navigating the store - and orders that can have a high number of items - less focus in the order processing activity and more in the product picking.

Regarding the reward system, for each product that is picked a reward of 100 is assigned which is the only positive reward the agent gets. The penalization for taking a step is 1, the redundant move penalization - when the agent returns to one of the two previously visited positions - is 1 and for each customer in the current picker position's neighbouring nodes and in the current node a penalization of 1 and 3 is applied, respectively. All the parameters used to build the simulation configurations are summed up in Table D.1 from Appendix D.

The customer paths inside the stores were obtained by creating random lists of items and the number of products is uniformly distributed between 1 and 25 products. The customer travels the minimum distance between every two products.

In this project, towards achieving parallelism four sub-processes were created which means that the training process had four environments training simultaneously a balance that leads to a good use of the machine's CPU cores while considering the memory usage.

The computations were run on Intel Core i7-8565U CPU @ 1.80GHz processing units.

5.2 Hyperparameter Optimization

Aiming for achieving the best set of hyperparameters for training different store layouts - sets that were tailored to drive learning in specific store configurations -, the hyperparameter tuning process was done accross 500 episodes for each trial - i.e., each set was tested during that number of episodes - and 50, 75 and 100 trials were performed for small, medium and large stores, respectively, to account for the different complexity of each layout different amount of resources were used. Each trial consists in training a model with a specific set of hyperparameters during a given number of episodes so that after the specified number of trials the different performance are compared and the hyperparameter sets are chosen. The sets were chosen based on the average episode reward from each trial and the five best sets from each layout were chosen to carry on to the training process. The results for the chosen sets - considering the hyperparameters that were

tuned - for each store layout are presented in Table 5.1 which shows the five best hyperparameter sets that were used in the trials that obtained the highest average reward across all trials performed for each store layout.

Table 5.1: Sets of PPO hyperparameters used to build training configurations.

Layout	Set	Learning Rate $\times 10^{-4}$	Discount Factor	Clipping Parameter	Entropy Coefficient $\times 10^{-2}$	Batch Size	Number of Epochs	Network Architecture
Small	1	6.421	0.958	0.159	3.461	128	5	[512, 256, 128]
	2	2.191	0.954	0.108	1.707	64	6	[512, 256, 128]
	3	5.835	0.957	0.137	0.889	256	8	[1024, 512, 256]
	4	8.819	0.950	0.202	0.646	512	5	[128, 256, 128]
	5	8.529	0.951	0.179	2.223	512	5	[512, 256, 128]
Medium	1	7.273	0.968	0.128	2.103	64	9	[256, 256, 256]
	2	4.933	0.962	0.100	3.647	256	9	[128, 256, 128]
	3	6.025	0.974	0.166	4.417	64	5	[256, 256, 256]
	4	6.783	0.972	0.171	2.667	64	7	[256, 256, 256]
	5	8.534	0.979	0.151	3.555	64	10	[256, 256, 256]
Large	1	5.612	0.974	0.148	5.883	64	9	[512, 256, 128]
	2	8.202	0.974	0.235	6.203	64	9	[1024, 512, 256]
	3	11.835	0.971	0.198	4.449	128	8	[512, 256, 128]
	4	7.619	0.985	0.231	3.105	64	10	[512, 256, 128]
	5	6.679	0.970	0.126	5.233	64	8	[512, 256, 128]

The *learning rate* was searched within the interval [0.0001, 0.0015] considering the 0.0003 default value. The search for the *discount factor* was in the range [0.95, 0.99]. The *clipping parameter* was tuned within the interval [0.1, 0.3] which is centered around the typical value of 0.2. The *entropy coefficient* was tuned within the range [0.005, 0.2] which mainly tested values above the default of 0.01 given the dimension of the state space. The *batch size* was tuned within the range [32, 512] - the default is 64 - and the increments are multiples of 32 so that it prevents redundant testing by ensuring that there is a significant difference between two different batch sizes when comparing both results. That allows verifying different trade-offs between the computational effort and the policy results. The default value for the *number of epochs* is 4 and it was tuned with values that fall under the interval [3, 10] which sounds appropriate given the dimension of a retail store environment. The tuning of the *network architecture* was based on choosing an option within the set $\{[128, 128, 128], [256, 256, 256], [128, 256, 128], [512, 256, 128], [1024, 512, 256]\}$ in which each has three hidden layers to capture the complexity of a retail store and variable number of neurons which performs the feature representation, while the number of layers allow to create a hierarchy of low level and high level data based on the input allowing the agent to learn more complex behaviours.

Regarding the *learning rate*, it is possible to notice that a wider range of learning rates work well for the small layouts going from a lower (2.191×10^{-4}) to a higher (8.819×10^{-4}) value while

regarding the other store layouts relatively higher learning rates with the minimum value for the medium and large stores being more than double than the latter's and the maximum value being similar between the medium and small stores but approximately 34% higher in the large store than the maximum value between the other two layouts. The consistently higher values along with the increasing layout dimension and consequently state space dimension aligns with how the learning rate works as taking large steps in less complex environments would more easily lead to achieving a sub-optimal solution while taking too small steps in more complex environments can increase the learning process difficulty and stop the algorithm from learning useful knowledge in a reasonable timeframe. In this context the steps the algorithm takes are the incremental adjustments made to the algorithm's parameters and its internal model.

Concerning the *discount factor*, the same tendency is shown as previously in which the value given to future rewards is higher as the store layout complexity increases. Given that the order lists are being equally generated across the different layouts, the distance between two picking locations will be higher as the store size increases which means that the picker has to travel higher distances to pick the product and go through more penalization to get the reward. That helps drive the agent towards the reward - as sometimes the next product to pick is near the previous one and sometimes it is further, which emphasizes the importance of the agent being able to wait to get the reward.

The *clipping parameter* did not show a specific tendency across the three layouts. However, the *clipping parameter* showed higher values for the large store which may indicate that this hyperparameter's influence is more noticeable for bigger differences between the layouts' complexities while for closer complexities like the small and medium stores' case it shows a similar influence.

The *entropy coefficient* as it determines the importance given to exploring the different state action pairs, noticeably increased along with the layout size between small and medium or large as well as medium and large stores which was expected since the state space increases exponentially with the increase of the store dimension in this particular context which means that there are more state action pairs to learn from and more exploration is needed.

The *batch size* was noticeably higher in the small store than in the medium and large stores, but it also showed more variability as all possible values for that hyperparameter are present at least one time except for a batch size of 32 which shows that for small stores the batch size does not have as much influence as for larger stores such as the medium and large ones in which for both a batch size of 64 was the best one in four of the best sets. That leads to a less stable training process despite reaching algorithm convergence faster, which given the complexity of bigger layouts is important so that it does not represent a strain in computational efforts.

The *number of epochs* shows the same tendency as previous hyperparameters for increasing values along with increasing store layouts reflecting the learning difficulty that comes with them as the algorithms needs to resort more times to learned experience in order to achieve better results.

Finally, the *network architecture* shows higher complexity for the large stores which aligns well with the higher number of states that are present meaning that more unseen states are found and each state is less explored as there are other to explore. As such, more generalization and better

feature representation are beneficial. However, that hyperparameter showed a higher complexity than expected for small stores and lower than expected for medium stores which can indicate that the robustness of the network architecture used for small stores was possibly high which may lead to overfitting while in the case of medium stores it might have been low which can cause the model to fail in capturing the complexity of the environment. (Zhang et al., 2018)

5.3 Impact of Layout Size

In order to reach results for the different layouts, each hyperparameter set for each layout was trained for 5000 episodes.

5.3.1 Algorithm Convergence

One of the goals when training a machine learning model is for the algorithm to converge, as such that was one of the factors that was assessed for each store layout and hyperparameter set combination. Also, it is important to consider the expectation that as the store size increases the maximum reward that can be expected for a similar number of episodes decreases given that there is a higher state space to explore and the agent needs to spend more time navigating the store in order to pick an order as well as that it takes longer to surpass the positive reward threshold the larger the store layout is.

Two examples from each layout will be discussed in which it is possible to verify two different performances that depict different scenarios regarding algorithm convergence.

Figure 5.1 represents the algorithm convergence for the small store layout across the first 2500 episodes. Set 1 shows convergence to a higher reward then set 4 and does so faster. While the latter appears to have a more linear growth, the first has an exponential growth and ends up with higher rewards throughout the rest of the training.

Comparison of Obtained Rewards for Small Layouts Across Different Hyperparameter Sets

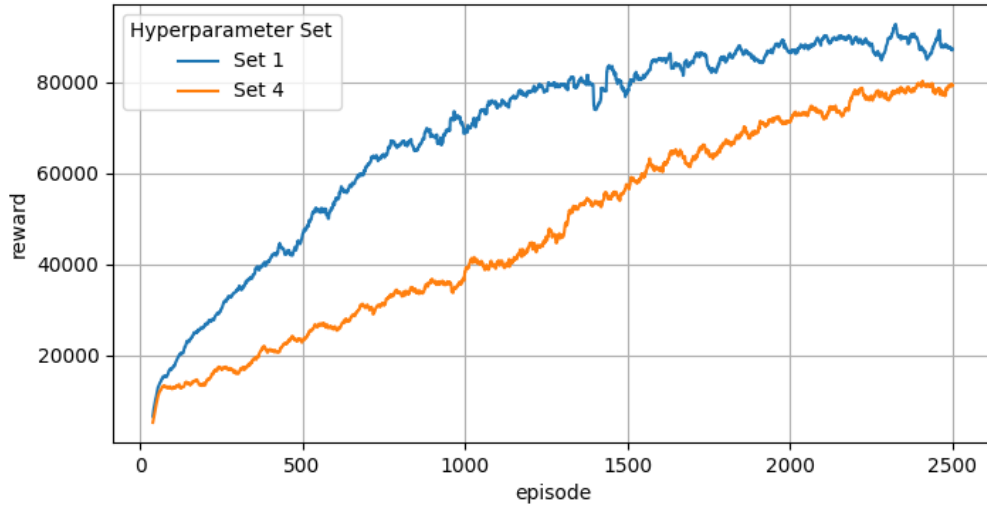


Figure 5.1: Comparison of Obtained Rewards for Small Layouts between sets 1 and 4 (first 2500 episodes).

Figure 5.2 represents the algorithm convergence for the medium store layout across the first 3000 episodes. As in the previous example, a similar situation is present in this case since set 5 converges faster than set 2 and reaches a higher value.

Comparison of Obtained Rewards for Medium Layouts Across Different Hyperparameter Sets

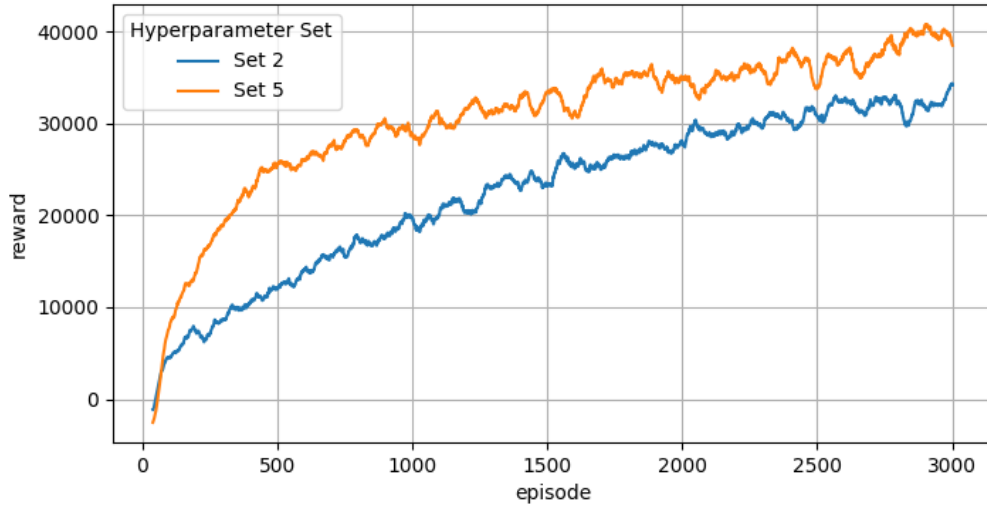


Figure 5.2: Comparison of Obtained Rewards for Medium Layouts between sets 2 and 5 (first 3000 episodes).

Figure 5.3 represents the algorithm convergence for the large store layout across the first 3500 episodes. In the case of the large store layout, both examples are sub-optimal. On one hand, set 1 appears to have converged to a poor value which means that it was not able to escape a local optima. On the other hand, while set 3 performed better it still was not able to converge showing

more instability towards the end with a tendency to keep improving the rewards. Given the higher complexity of the large store layout the decision was made to keep training the models for that configuration for an additional 2500 episodes.

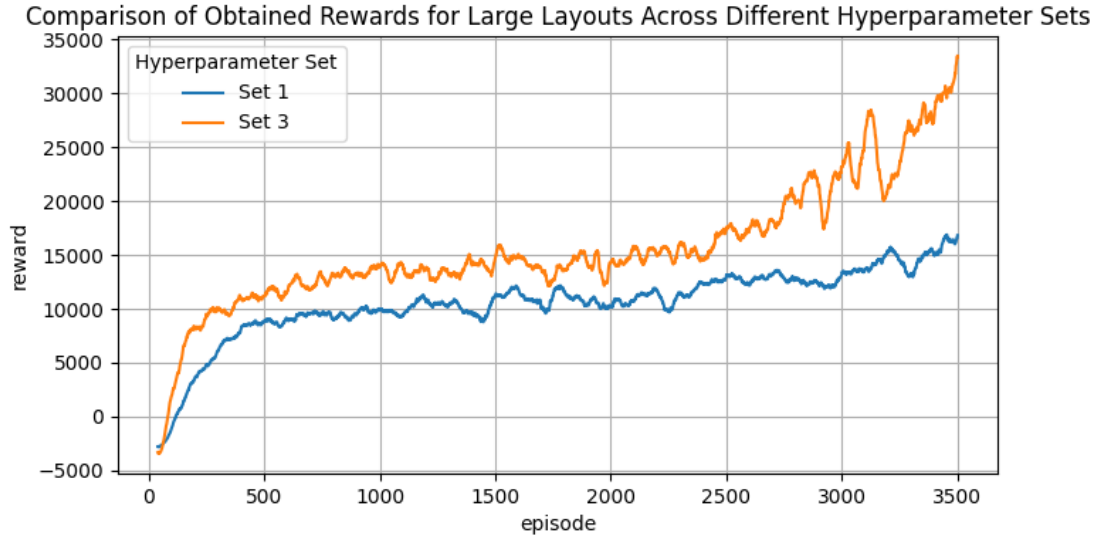


Figure 5.3: Comparison of Obtained Rewards for Large Layouts between sets 1 and 3 (first 3500 episodes).

Table 5.2 was obtained after training the layouts and already consider the additional training that was performed in the large stores.

By analysing Table 5.2 it is noticeable that the average and maximum rewards decrease as the layout of the store becomes more complex to navigate in which was expected. Regarding the minimum reward it did not show a clear tendency depending on the store layout despite the medium and large stores showing two discrepant values which are likely due to initial exploration of large state spaces. However, it was expected that the minimum reward would decrease as well, which did not happen. The coefficient of variance was computed for the final 100 episodes, showing that small store configurations converge more easily than the other configurations - the same happens with the medium store configuration in comparison with the large store configuration. As shown by that metric, large store configurations would require a more extensive training process to converge.

5.3.2 Performance metrics

As the goal of the work is to learn a picking policy that not only teaches an agent how to efficiently pick products but also to avoid physical customers, the metrics that were evaluated relate with that purpose.

As product picking and customer avoidance are the objectives, analysing how the ratio of customer encounters per product picked is important. Figure 5.4 shows that metric for the three different store configurations trained with a given hyperparameter set that after analysis proved to

Table 5.2: Reward metrics obtained during the training procedure.

Layout	Set	Min.	Avg.	Max.	CV (last 100 episodes)
Small	1	-4971.3	82637.3	117674.5	0.08
	2	-5088.7	76892.4	116559.2	0.10
	3	-6088.1	83100.7	123196.4	0.07
	4	-4437.0	69385.7	114600.3	0.11
	5	-5450.5	70172.0	112146.6	0.08
Medium	1	-4235.8	44299.8	105630.7	0.25
	2	-4262.1	28298.5	52365.0	0.09
	3	-3922.6	32143.4	56070.8	0.11
	4	-36206.9	35565.3	66022.1	0.10
	5	-4144.3	39965.6	77736.6	0.06
Large	1	-3750.5	20325.1	45827.6	0.23
	2	-3321.6	21802.6	46707.5	0.14
	3	-6301.3	23650.3	66673.5	0.16
	4	-11903.2	16165.0	51726.2	0.32
	5	-4395.5	24746.6	58057.6	0.16

be the best for each considering the length of the training process. For all stores layouts there is a rapid decrease of the ratio that after the initial 500 to 1000 episodes starts to stabilize but still showing some decrease, which corroborates that the agent is learning how to efficiently travel the store while avoiding customer encounters. As the customer arrival rate and allowed number of customers inside the store and in each node did not change for different layouts, as the store size increases the ratio can reach lower values more easily given that the customers are more distributed across the store.

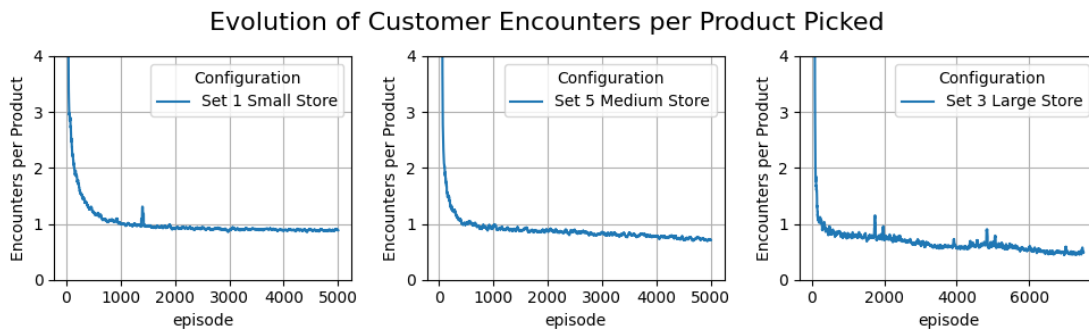


Figure 5.4: Evolution of Customer Encounters per Product Picked.

The results for the testing phase of the trained policies in the different layouts that took place for 500 episodes are presented in Table 5.3 and allow reaching certain conclusions.

Firstly, there is a clear difference between the metrics when comparing across different store layouts. A measure of good performance has different standards when the store size increases

given the increase of the travel time between two positions as well as the increase of computational efforts required to gain knowledge about the state space.

Regarding the small store, it is possible to conclude that performance is not widely affected by the hyperparameters as performance is very similar across the different hyperparameter sets.

The performance difference for different sets of hyperparameters is more clear within the medium and large stores. Given the larger state spaces that are inherent to those configurations tailoring the hyperparameters becomes more relevant to guide the learning phase.

In the medium store, the model trained with the hyperparameter set 5 stands out positively as it not only outperforms the others regarding operational performance but also regarding the average encounters per product picked. The main differentiating hyperparameters in that set are the learning rate and the discount factor, with the performance pointing to having found the ideal values to those hyperparameters.

Regarding the large store, the model trained with hyperparameter set 4 has a bad performance when comparing with the others which may be due to the entropy coefficient that was considered being too low and by not allowing the agent to explore a large state space it fell into a sub-optimal policy. Moreover, it is noticeable that while hyperparameter set 3 achieved the best operational performance, hyperparameter set 1 achieved a slightly lower ratio of customer encounters per product.

Table 5.3: Performance Metrics upon Testing for 500 Episodes.

Layout	Set	Avg. #Orders	Avg. #Products	Avg. #Encounters	Avg. #Steps	Avg. Steps per Product	Avg. Encounters per Product	Avg. Rewards per Product
Small	1	88.48	1060.58	941.89	3269.78	3.08	0.89	91.61
	2	88.03	1054.59	925.04	3303.02	3.13	0.88	92.02
	3	90.12	1078.25	944.92	3351.87	3.11	0.88	92.27
	4	86.27	1035.69	922.72	3259.29	3.15	0.89	91.50
	5	85.31	1020.80	916.38	3203.74	3.14	0.90	91.47
Medium	1	57.81	691.13	489.42	3715.57	5.38	0.71	89.40
	2	42.42	504.94	395.67	2916.97	5.78	0.78	88.07
	3	41.26	491.62	386.94	2803.56	5.70	0.79	88.05
	4	47.59	566.41	420.59	3133.64	5.53	0.74	88.76
	5	66.80	812.72	533.41	4222.39	5.20	0.66	89.96
Large	1	32.20	380.80	224.71	3827.52	10.05	0.59	83.66
	2	32.55	383.92	230.56	3873.80	10.09	0.60	83.59
	3	45.44	523.85	317.55	5368.69	10.25	0.61	82.66
	4	16.80	197.27	145.06	2244.35	11.38	0.74	78.94
	5	44.85	518.90	317.24	5320.16	10.25	0.61	82.98

5.4 Retraining Models in Increasingly Complex Environments

One of the purposes of training an agent to navigate a store is to achieve generalization making it possible to leverage transfer learning skills and use an agent that is already trained in performing a task to learn a new and more complex task. Instead of training the agent from the beginning to perform the task it is intended to which makes the learning process more complex and difficult. As the small store is essentially a sub-graph of the medium store and the same with the medium and large stores, the expectation is that by learning how to navigate a small or medium store the agent will then more easily learn how to navigate a medium or large store, respectively.

Following that rationale, the policy that was learned in the small store configuration with hyperparameter set 1 and in the medium store with hyperparameter set 5 were used to train a new agent in a medium and large stores, respectively, for an additional 2500 episodes. The hyperparameter sets that were used from each configuration were chosen based on the stability of the training in those configurations as well as on the rewards achieved.

In both cases, the retraining strategy was characterized by a less stable training process and an initial confusion of the agent reaching rewards in the order of -50000 which in comparison to the minimum rewards obtained in the training process of agents in different layouts without prior knowledge shown in Table 5.3 is a low value. That can be explained by the fact that that specific model was trained for a certain configuration and is now suddenly exposed to a different one in which there is a higher state space dimension and is faced with more unknown scenarios. However, the agent was able to adapt and leverage its initial experience.

- **Medium Store Configuration:** in this configuration, the results of retraining a model from a small store were similar to the ones obtained through the training of a medium store using hyperparameter set 4, which were essentially average performance results for that configuration;
- **Large Store Configuration:** in this configuration, the results obtained by retraining a model from a medium store showed improvement over the training without prior knowledge by reaching an average of picked products during testing of 657.33 products translating into an improvement of 25.48% over the best operational results obtained for large stores and an average of encounters per product of 0.55 encounters per product improving 6.78% over the best trained model in that regard.

The improvements that happened in the large store configuration for the same number of total training episodes as the initial training - 5000 episodes that built the model from the medium store and the additional 2500 episodes that built the final retrained model in the large store - show that the agent can transfer skills from completing a less complex task and then moving on to a more complex one.

5.5 Real World Store Configuration

The culmination of developing this project was to obtain the proposed goals in a real world store from a big retailer.

Figure 5.5 represents the real world store graph from an European retailer that inspired this work. The gray nodes represent the product positions, the white nodes are nodes without differentiation the agent can use to reach the product positions', the black nodes are divided between the tills and the exits and the red node is the entrance.

The difference between the architecture of the real store and the synthetic stores is noticeable. Firstly, the nodes are not equally distant meaning that the agent needs to travel different distances considering the positions in which it might be. Secondly and more importantly, the layout is not rectangular. It is characterized by an irregular shape mainly around the corners. The agent's movements are more irregular in the real store due to the node connections and the store's shape making it harder to learn valuable actions given that along with that the state space has a high dimension.

The same configuration that was used in the synthetic stores to create the simulation environment is used in the real world store. That allowed to keep the learning process more understandable as it leads to a less chaotic environment and validate whether the method is viable or not to drive learning in a real store. Considering the size of the real store comparing to the synthetic stores it leads to a learning process that focuses more in learning efficient store navigation than customer encounters avoidance.

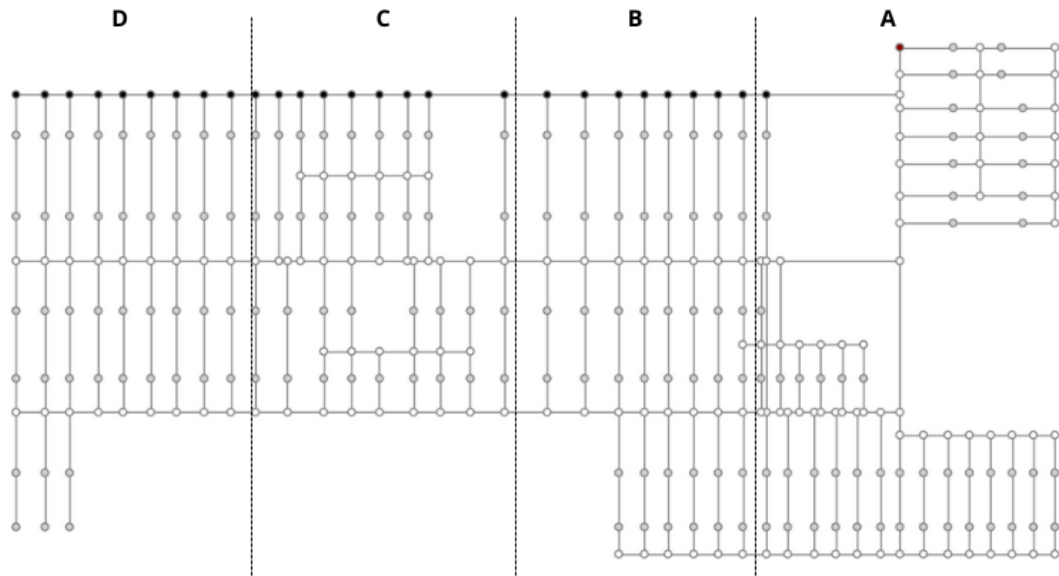


Figure 5.5: Real World Store Graph Representation with Subsections.

Given the previous strategy that was followed with synthetic stores showed good results when retraining a model from a medium store in a large store, the same strategy was followed to train

an agent in the real store configuration.

The hyperparameters that were used were based on an adaption of the hyperparameters tailored to the large store configuration given that its the closest size to the real store but disregarding the fact that it has a regular shape - hyperparameter set 3 was used, except for the network architecture which was replaced for $[1024, 512, 256]$ to capture the higher complexity and the discount factor which was set to 0.985 to value more the expected future rewards given the high dimensional state space.

Figure 5.5 is divided into four subsections that were used to build up the complexity of the store in which the model was trained. Initially, the model was trained for 5000 episodes in each of the following settings: subsection A, followed by adding B to the configuration and then C was added as well. Finally, subsection D was added making up the full store and the model trained for 10000 episodes. The first conclusion that was drawn was that despite the initial adaption difficulty when retraining a model was present when going from subsection A to subsections A and B, but afterwards when subsections C and D were added the agent had a better initial performance showing that it would do well even in the presence of unseen states. Secondly, one of the main problems that affected the performance relates to the agent remaining in the same position instead of moving to different positions. Finally, the model did not undergo testing as its performance did not approach an optimal one - considering there was too much time wasted in the same position which does not show realistic behavior.

However, the results observed during training validate the notion that it is possible to use the devised methodology to drive learning in a real world store. By analyzing the best performance of the agent during training it shows that when the picker is effectively moving it moves efficiently. The best performance featured a number of orders served of 16, a number of products picked of 225, a number of encounters of 214, 1916 steps and a reward of 17389. This translates into 0.95 encounters per product, 77.28 reward per product and 8.51 steps per product. The agent can indeed learn how to navigate the store and even avoid customers, but this results were far from optimal due to the previous described behavior despite the redundant move penalization that was implemented.

Tailoring the hyperparameters and the rewards and penalizations to the specific conditions of the real world store would be essential to achieving better results and reaching an optimal picking policy as well as performing several training scenarios to compare results and decide upon the best policy.

Chapter 6

Conclusion

To sum up, this dissertation addressed the challenges that arise in the present times - regarding the growing online channel demands as well as the customers expectations that are always increasing due to the highly competitive market that is the retailing industry - by tackling in-store picking operations and how those can be addressed within a simulated store environment. In doing so, PPO, a deep RL technique, was leveraged to control an agent during the navigation of the store environment modelled as a MDP.

During the development of the project the main goal was to devise a strategy that led to training an agent that was capable of navigating a grocery store in an efficient way, such that the total travelled distance was minimized, and avoiding the physical customers present in the store that are currently navigating their own shopping paths.

This project follows the concern of how physical stores are not tailored for order picking such as warehouses are which can lead to inefficiencies and increase operational costs as well as how seeing pickers and crossing paths with them in the store's aisles can affect the physical customer's perception of the store by feeling that he/she is not in a store but rather in a warehouse.

Obtaining a picking policy that aims at addressing those issues was the goal of training agents using PPO. To do so, the hyperparameters that serve as an input to the model before the training process begins were tuned. The hyperparameter tuning for the synthetic stores configurations showed more relevance for medium and large stores and allowed training an agent capable of navigating the store and learning from the observations it got.

In the training process, the agent was capable of following the proposed goals and navigate the store while avoiding customers - considering the weight given to each factor when assigning the rewards and penalizations that prioritizes store navigation as it directly increases costs if it is inefficient.

The retraining of an agent in a more complex environment - that used the models that were already trained - proved to be effective between the medium and large stores.

The next challenge was to achieve a picking policy in a real world store of an European retailer. Training a real world store is a much more complex task due to its inherent irregularity mainly due to the outer shape of the store and also the connections between nodes - namely, some portions

of the stores are only accessible through two nodes and changing positions does not change the coordinates in the same way every time as the distances between nodes are not the same for every two nodes.

The results for real world stores lack further training with different hyperparameter sets specifically tailored to that scenario as well as simulation parameters that resemble a real world environment.

Despite only achieving a sub-optimal policy for the real world store, those results along with the results for the synthetic stores validate the followed approach.

The proposed goals were achieved when considering the synthetic stores with the agent successfully learning how to balance the trade-off between product picking and customer avoidance making it a promising strategy to employ in a real world scenario. In the case of the real world scenario, the proposed goals were not achieved as only preliminary actions were taken, but the obtained results validate the strategy and motivate further work into making it work. In this scenario, subsections of the graph were sequentially added to the training process and as the picker is presented with unseen states when adding the two final subsections and maintaining its performance - despite sub-optimal - it shows that it is able to generalize the previously learned experience. That is important because there is a high number of stores and employing the same agent by adding minimal additional training between different stores would reduce implementation costs.

The dissertation had limitations and left room for future work to be developed. The work development was limited by the computer's capacity that was used to run the simulations that did not always overcome the computational efforts required causing sometimes sudden failures in the program. Besides, a successful run of an entire training process is a prolonged process.

Regarding future work, there is plenty that can be done still. Firstly, finalizing this approach to the problem by doing hyperparameter tuning for the real world store and implementing real world parameters as well as running more extensive training processes to achieve an optimal picking policy which would allow testing the agent in different scenarios to validate the policy and implementing it.

Secondly, there are other possible algorithms such as A2C that can be tested and compare the performance against PPO.

Thirdly, the features implemented in the training process create a large state space which brings more complexity to the agent's learning process. Further studying the implementation of other possible features that can better represent the state the agent is in or the performance of the policy by reducing the number of feature by using techniques such as Principal Component Analysis can help reducing the state space or improving its efficiency.

References

- Adhi, P., Calais, S., Hough, G., Lange, T., and Lenzen, C. (2021). Reimagining the role of physical stores in an omnichannel distribution network. *McKinsey & Company*. Available at: <https://www.mckinsey.com/industries/retail/our-insights/reimagining-the-role-of-physical-stores-in-an-omnichannel-distribution-network/>. Accessed: 2024-04-15.
- Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.
- Andrychowicz, M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J. W., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. (2018). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39:20 – 3.
- Anshu, K., Gaur, L., and Singh, G. (2022). Impact of customer experience on attitude and repurchase intention in online grocery retailing: A moderation mechanism of value co-creation. *Journal of Retailing and Consumer Services*, 64:102798.
- Arslan, A. N., Klibi, W., and Montreuil, B. (2021). Distribution network deployment for omnichannel retailing. *European Journal of Operational Research*, 294(3):1042–1058.
- Bai, J., Fu, F., Guan, R., Hoffman, S., Karnani, P., Mysore, M., and Touse, S. (2021). A path toward healthy, sustainable growth in e-commerce. *McKinsey & Company*. Available at: <https://www.mckinsey.com/industries/retail/our-insights/solving-the-paradox-of-growth-and-profitability-in-e-commerce>. Accessed: 2024-03-27.
- Belete, D. and D H, M. (2021). Grid search in hyperparameter optimization of machine learning models for prediction of hiv/aids test results. *International Journal of Computers and Applications*, 44:1–12.
- Berbeglia, G., Cordeau, J.-F., and Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15.
- Cao, Z., Zhou, L., Lin, C., and Zhou, M. (2023). Solving an order batching, picker assignment, batch sequencing and picker routing problem via information integration. *Journal of Industrial Information Integration*, 31:100414.
- Caro, F., Kök, A. G., and Martínez-de Albéniz, V. (2020). The future of retail operations. *Manufacturing & Service Operations Management*, 22(1):47–58.
- Chen, F., Wang, H., Qi, C., and Xie, Y. (2013). An ant colony optimization routing algorithm for two order pickers with congestion consideration. *Computers & Industrial Engineering*, 66(1):77–85.

- Chou, X., Loske, D., Klumpp, M., Gambardella, L. M., and Montemanni, R. (2021). In-store picking strategies for online orders in grocery retail logistics. *AIRO Springer Series*.
- De Koster, R. (2008). Warehouse assessment in a single tour. *Warehousing in the Global Supply Chain: Advanced Models, Tools and Applications for Storage Systems*.
- Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. (2017). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28:653–664.
- Dennis, S. (2018). Physical retail isn't dead. boring retail is. *Forbes*. Available at: <https://www.forbes.com/sites/stevendennis/2018/03/19/physical-retail-is-not-dead-boring-retail-is-understanding-retails-great-bifurcation>. Accessed: 2024-03-17.
- Dethlefs, C., Ostermeier, M., and Hübner, A. (2022). Rapid fulfillment of online orders in omnichannel grocery retailing. *EURO Journal on Transportation and Logistics*, 11:100082.
- Difrancesco, R. M., van Schilt, I. M., and Winkenbach, M. (2021). Optimal in-store fulfillment policies for online orders in an omni-channel retail environment. *European Journal of Operational Research*, 293(3):1058–1076.
- Duval, J. (2019). Shipping is critical to keeping online shoppers happy. *Forbes*. Available at: <https://www.forbes.com/sites/forbestechcouncil/2019/08/27/shipping-is-critical-to-keeping-online-shoppers-happy/>. Accessed: 2024-04-02.
- Falkner, S., Klein, A., and Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale.
- Gao, F. and Su, X. (2017). Omnichannel retail operations with buy-online-and-pick-up-in-store. *Management Science*, 63(8):2478–2492.
- Gong, Y. and De Koster, R. (2008). A polling-based dynamic order picking system for online retailers. *IIE transactions*, 40(11):1070–1082.
- He, Y., Xu, Q., and Shao, Z. (2021). “ship-from-store” strategy in platform retailing. *Transportation Research Part E: Logistics and Transportation Review*, 145:102153.
- Hoang, D. and Breugelmans, E. (2023). “sorry, the product you ordered is out of stock”: Effects of substitution policy in online grocery retailing. *Journal of Retailing*, 99(1):26–45.
- Hosseini, Z., Jayashree, S., and Malarvizhi, C. (2014). Store image and its effect on customer perception of retail stores. *Asian Social Science*, 10:223–235.
- Huang, H. and Tan, X. (2021). Application of reinforcement learning algorithm in delivery order system under supply chain environment. *Mobile Information Systems*, 2021:1–11.
- Hübner, A. H., Kuhn, H., and Wollenburg, J. (2016). Last mile fulfillment and distribution in omnichannel grocery retailing: a strategic planning framework. *International Journal of Retail & Distribution Management*, 44(3).
- Hübner, A., Wollenburg, J., and Holzapfel, A. (2016). Retail logistics in the transition from multi-channel to omni-channel. *International Journal of Physical Distribution I& Logistics Management*, 46:562–583.

- Jin, M., Li, G., and Cheng, T. (2018). Buy online and pick up in-store: Design of the service area. *European Journal of Operational Research*, 268(2):613–623.
- Ketzenberg, M. and Akturk, M. S. (2021). How “buy online, pick up in-store” gives retailers an edge. *Harvard Business Review*. Available at: <https://hbr.org/2021/05/how-buy-online-pick-up-in-store-gives-retailers-an-edge>. Accessed: 2024-03-23.
- Komorowski, M., Celi, L. A., Badawi, O., Gordon, A. C., and Faisal, A. A. (2018). The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine*, 24:1716–1720.
- Kozlica, R., Wegenkittl, S., and Hiränder, S. (2023). Deep q-learning versus proximal policy optimization: Performance comparison in a material sorting task. pages 1–6.
- Kumaraswamy, R., Schlegel, M., White, A., and White, M. (2018). Context-dependent upper-confidence bounds for directed exploration. In *Neural Information Processing Systems*.
- Li, H., Chen, T., Teng, H., and Jiang, Y. (2019). A graph-based reinforcement learning method with converged state exploration and exploitation. *Computer Modeling in Engineering & Sciences*.
- Li, Y. (2017). Deep reinforcement learning: An overview. *ArXiv*, abs/1701.07274.
- Lu, W., McFarlane, D., Giannikas, V., and Zhang, Q. (2016). An algorithm for dynamic order-picking in warehouse operations. *European Journal of Operational Research*, 248(1):107–122.
- Masae, M., Glock, C. H., and Grosse, E. H. (2020). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, 224:107564.
- Memarian, B. and Doleck, T. (2024). A scoping review of reinforcement learning in education. *Computers and Education Open*, 6:100175.
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783.
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning.
- Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., Maria, A. D., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., Legg, S., Mnih, V., Kavukcuoglu, K., and Silver, D. (2015). Massively parallel methods for deep reinforcement learning.
- Neves-Moreira, F. and Amorim, P. (2024). Learning efficient in-store picking strategies to reduce customer encounters in omnichannel retail. *International Journal of Production Economics*.
- Novoa, C. and Storer, R. (2009). An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European journal of operational research*, 196(2):509–515.
- Ognibene, N., Chou, X., Loske, D., Klumpp, M., and Montemanni, R. (2021). The buy-online-pick-up-in-store retailing model: Optimization strategies for in-store picking and packing. *Algorithms*, 14.

- Pansart, L., Catusse, N., and Cambazard, H. (2018). Exact algorithms for the order picking problem. *Computers & Operations Research*, 100:117–127.
- Petersen, C. G. and Aase, G. (2004). A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, 92(1):11–19.
- Pietri, N. O., Chou, X., Loske, D., Klumpp, M., and Montemanni, R. (2021). The buy-online-pick-up-in-store retailing model: Optimization strategies for in-store picking and packing. *Algorithms*, 14(12).
- Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11.
- Powell, W. B. (2007). Approximate dynamic programming - solving the curses of dimensionality. In *Wiley Series in Probability and Statistics*.
- Powell, W. B. (2019). A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3):795–821.
- Ratliff, H. D. and Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations research*, 31(3):507–521.
- Rolf, B., Jackson, I., Müller, M., Lang, S., Reggelin, T., and Ivanov, D. A. (2022). A review on reinforcement learning algorithms and applications in supply chain management. *International Journal of Production Research*, 61:7151 – 7179.
- Rosário, A. and Raimundo, R. (2021). Consumer marketing strategy and e-commerce in the last decade: a literature review. *Journal of theoretical and applied electronic commerce research*, 16(7):3003–3024.
- Ruelens, F., Claessens, B., Vandael, S., Schutter, B. D., Babuvska, R., and Belmans, R. J. M. (2017). Residential demand response of thermostatically controlled loads using batch reinforcement learning. *IEEE Transactions on Smart Grid*, 8:2149–2159.
- Savelsbergh, M. and Van Woensel, T. (2016). 50th anniversary invited article—city logistics: Challenges and opportunities. *Transportation science*, 50(2):579–590.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.
- Seghezzi, A., Siragusa, C., and Mangiaracina, R. (2022). Enhancing in-store picking for e-grocery: an empirical-based model. *International Journal of Physical Distribution & Logistics Management*, 52:301–323.
- Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 9:1054–1054.
- Sutton, R. S., Singh, S. P., and McAllester, D. A. (2000). Comparing policy-gradient algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 30(4):467–477.
- Theys, C., Bräysy, O., Dullaert, W., and Raa, B. (2010). Using a tsp heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, 200(3):755–763.

- Thomas Chabot, Rahma Lahyani, L. C. C. and Renaud, J. (2017). Order picking problems under weight, fragility and category constraints. *International Journal of Production Research*, 55(21):6361–6379.
- Tompkins, J. A., White, J. A., Bozer, Y. A., and Tanchoco, J. M. A. (2010). *Facilities planning*. John Wiley & Sons.
- Ulmer, M. W., Goodson, J. C., Mattfeld, D. C., and Thomas, B. W. (2020). On modeling stochastic dynamic vehicle routing problems. *EURO Journal on Transportation and Logistics*, 9(2):100008.
- Vieillard, N., Pietquin, O., and Geist, M. (2020). Deep conservative policy iteration.
- Yang, L., Li, X., and Zhong, N. (2022). Omnichannel retail operations with mixed fulfillment strategies. *International Journal of Production Economics*, 254:108608.
- Zhang, C., Vinyals, O., Munos, R., and Bengio, S. (2018). A study on overfitting in deep reinforcement learning. *arXiv*.
- Zhang, T., Lin, Z., Wang, Y., Ye, D., Fu, Q., Yang, W., Wang, X., Liang, B., Yuan, B., and Li, X. (2022). Dynamics-adaptive continual reinforcement learning via progressive contextualization.
- Zhou, C., Ma, J., Douge, L., Chew, E. P., and Lee, L. H. (2023). Reinforcement learning-based approach for dynamic vehicle routing problem with stochastic demand. *Computers & Industrial Engineering*, 182:109443.
- Zhu, Z., Lin, K., Jain, A. K., and Zhou, J. (2023). Transfer learning in deep reinforcement learning: A survey.

Appendix A

Sustainable Development Goals

Nowadays more than ever, it is important to keep sustainability in mind. Not only environmental sustainability but also economical and social sustainability. Growth can be achieved in such a way that does not exhaust available resources, considering paths that not only add value to a given business but also to society.

Considering that, the United Nations adopted a set of 17 goals in 2015 called Sustainable Development Goals that aim the improvement of the planet and society as a whole until 2030. The goals are intertwined with each other and assume the influence each has in the others.

A great part of fulfilling those goals is in the hands of innovation through research and the development of innovative solutions that can, for instance, reflect a way of tackling problems within businesses and institutions relating with improving the processes while following values that aim the stakeholders of those businesses. Engineering is precisely about devising innovative solutions that translate theoretical concepts into applicable ones.

As such, this Chapter is about the Sustainable Development Goals with which this dissertation is connected.

Among the existent goals, the ones with which a direct or indirect relation was found are as follows.

Goal 8: Decent Work and Economic Growth

This goal relates with reaching full and productive employment as well as decent work for men and women equally and sustainable economic growth.

The optimization of in-store picking activities that this dissertation aims leads to more efficient logistic operations which reduces the costs related with that and in turn promotes economic growth. Also, preserving the customer's shopping experience maintains their loyalty towards the retailer which contributes to economic growth. Besides, it leads to more productivity as the picker will not need to have an extensive knowledge about the store layout in order to navigate through it which could lead to wasted time thinking where to go next or how to get there. That promotes productive employment.

Goal 9: Industry, Innovation and Infrastructure

This goal targets the improvement of technological capabilities of all industry sectors encouraging the enhancement of scientific research through the increase of the number of research and

development workers as well as the development of sustainable and quality infrastructure to support economic development accessible to all.

The proposed method that focuses on the use of PPO to tackle the in-store picking challenges towards the improvement of logistical operations promotes that goal as it is a relatively unexplored method contributing to innovating and advancing the retail industry's technological capabilities.

Goal 12: Responsible Consumption and Production

This goal aims the responsible use of resources and supporting developing countries shifting towards more sustainable patterns and directly relates with reducing the food waste in retailers and consumers in order to create more sustainable supply chains.

The dissertation aim in optimizing product picking and order fulfillment can help reduce the lead time between placing an order and receiving it which allows retailers to move their products quickly which is important in the case of perishable goods that easily go to waste. Also, by not depending so much on subjective workers qualities relating with product picking a better stock management can be achieved which prevent overstocking and understocking, reducing the excessive use of resources (which contributes to the goal) and maintaining customer satisfaction, respectively.

Goal 13: Climate Action

This goal aims the integration of climate change measures into national policies, strategies and planning and adopting low-carbon development as well as support developing countries in the process in order to fight the climate change.

This dissertation has a less direct impact in this goal but it is still worth mentioning. By adopting home deliveries (like with SFS) one vehicle serves the needs of many people that would leave their homes and use their cars to go shopping for groceries which reduces carbon emissions directly by reducing the number of vehicles that are being used and replacing them with one but also indirectly as the spreading of home delivery can reduce traffic in urban areas which leads to lower gas consumption due to stopping and starting the vehicles less times. Also, by reducing the number of customers in the store the energy consumption can also decrease.

To sum up, while this represents only a reflection on how this dissertation contributes to the Sustainable Development Goals it can be further developed to measure the impact of some of the aspects that were discussed.

Appendix B

Q-Learning

Q-Learning is a model-free off-policy value-based RL algorithm. Regarding model-free algorithms, the agent does not require a model of the environments making it learn based only on experience collected through iterative observation of the consequences of its actions. An off-policy algorithm refers to one where the agent learns the optimal policy - value function - while following a different policy - behaviour policy. The value-based dimension of the algorithm refers to its focus in learning the value different actions have, if taken, in each state - action-value functions, $Q(s_k, a)$ which shows the future expected rewards of taking each action in each state, meaning that in each state s_k the chosen action a_k will be the action a that maximizes the future expected rewards. The action a_k that the agent takes is the action $a \in \mathcal{A}(s_k)$ that maximizes the value function which is based on the Bellman equation (B.1).

On one hand, the Bellman equation for the value function $V^*(s_k)$ is given by:

$$V^*(s_k) = \max_a \{r_{k+1} + \mathbb{E}[V(s_{k+1}) \mid s_k, a_k]\} \quad (\text{B.1})$$

where:

- $V^*(s_k)$ is the value of state s_k - the maximum expected rewards from starting in that state;
- r_{k+1} is the immediate reward received after taking action a in state s_k ;
- $V(s_{k+1})$ is the value of state s_{k+1} - the maximum expected rewards from starting in that state.

On the other hand, $V^*(s_k)$ is given by $\max_a Q^*(s_k, a)$.

Q-Learning approximates Q^* through an iterative process by executing Equation B.2:

$$Q^{\text{new}}(s, a) = Q(s, a) + \alpha \left(R + \gamma \max_{a'} Q(s', a') - Q(s, a) \right) \quad (\text{B.2})$$

where:

- $Q^{\text{new}}(s, a)$ is the updated value compared with the current value $Q(s, a)$ of the action-value function for state s and action a ;
- α is the learning rate and it determines how much weight is given to new information;
- R is the reward received after taking action a in state s ;
- γ is the discount factor and it represents the importance that is given to future expected rewards when already in the new state s' ;

- $\max_{a'} Q(s', a')$ is the maximum future expected rewards when at the next state s' - after taking action a in state s - over all possible actions a' that can be taken there.

Appendix C

Synthetic Stores Graphs

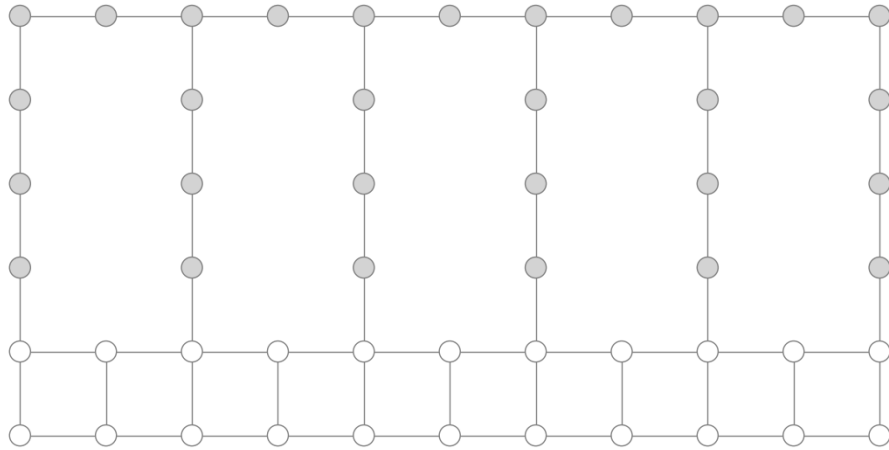


Figure C.1: Small Store Graph Configuration.

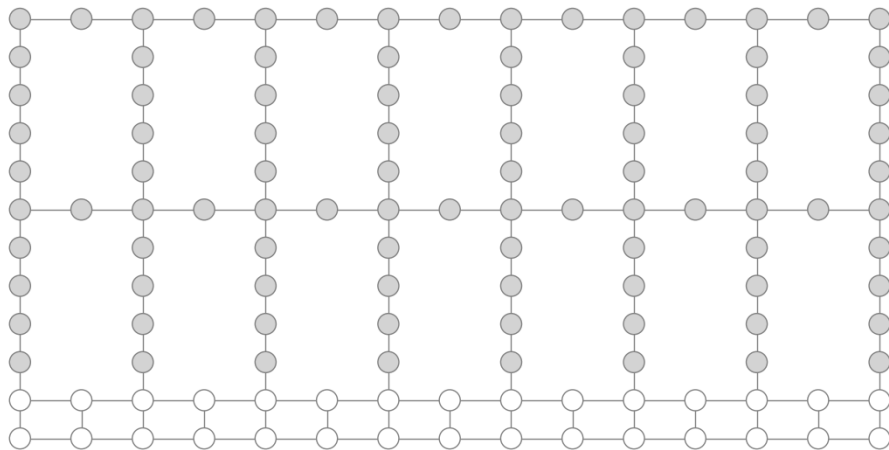


Figure C.2: Medium Store Graph Configuration.

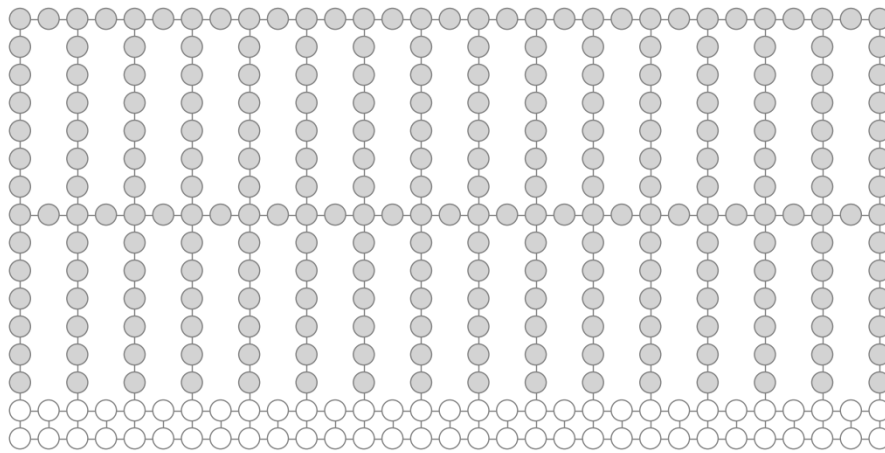


Figure C.3: Large Store Graph Configuration.

Appendix D

Simulation Parameters

Table D.1: Parameters used to build the simulation configurations.

Store layout	{ Small, Medium, Large }
Physical customer arrival rate	{ 2 }
Online order arrival rate	{ 0.2 }
Open time (h)	{ 8 }
Maximum number of customers in the store	{ 50 }
Maximum number of customers in each node	{ 5 }
Product picking time (s)	{ 15 }
Walking speed (m/s)	{ 1 }
Customer encounter penalty	{ 3 }
Customer visibility penalty	{ 1 }
Step penalty	{ 1 }
Redundant move penalty	{ 1 }
Picking reward	{ 100 }