FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

RETAILL - REtail using Technology based on Artificial InteLLigence

Leonor Marques Gomes

U. PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado em Engenharia Informática e Computação

Supervisors: Prof. Luís Paulo Reis, Prof. Brígida Mónica Faria

July 28, 2023

RETAILL - REtail using Technology based on Artificial InteLLigence

Leonor Marques Gomes

Mestrado em Engenharia Informática e Computação

July 28, 2023

Abstract

One of today's current struggles is the amount of wasted food that could be used to feed people. This pressing matter has become a Sustainable Development Goal, 12.3, which says, "By 2030, food losses along production and supply chains must be halved". One of the leading causes of food waste in the supply chain is transportation. Besides food waste, transportation also significantly negatively impacts the environment.

Thus, the main goal of this dissertation is to tackle this problem by developing route optimisation algorithms, connecting producers of fruits and veggies and consumers, and considering food waste and environmental impact. The research question connected to this is "How to improve route optimisation considering sustainability and food waste minimisation?". With this in mind, considering these factors can result in developing transportation routes that are more environmentally conscious and decreasing food waste.

First, a systematic literature review was done to gather the state of the art. During this part of the investigation, no other attempt was found to combine food waste and environmental sustainability in route optimisation. Afterward, the algorithms were developed using Hill Climbing, Simulated Annealing, Tabu Search, and Genetic Algorithm with a generated dataset. The project contains a baseline approach, a food allocation optimisation feature, an option to allocate food close to the consumption date to consumers able to receive food around the consumption date, and the chance to calculate routes for several days in a row.

Furthermore, the different modalities and algorithms were tested with different parameters, running each three times and registering the fitness, the time it took in seconds, and the standard deviation. In each algorithm, the influence of each parameter was analysed. Furthermore, a comparison was made between the four algorithms for the baseline option, and the food allocation optimisation was analysed, comparing this feature among the different algorithms and the baseline results. The option to allocate food close to the consumption date was tested as well, comparing the results between the four algorithms and the option to calculate routes for several days in a row. Finally, different variations of the tweak function were also tested to understand which one worked best. For most modalities, the best algorithm was Simulated Annealing, followed by Tabu Search, Hill Climbing, and Genetic Algorithm. The diversity and efficiency of the solution were discussed, as well as the challenges faced when developing the solution.

Finally, it was concluded that the dissertation reached its goal and could answer the research question by successfully presenting a solution incorporating an economic factor, a food waste factor, and an environmental impact factor in calculating routes from producers to consumers. Furthermore, it contributed to the field by presenting a first attempt at combining food waste and environmental impact in route optimisation, and the prospects of future work were also discussed to enrich this project.

Keywords: Vehicle Route Optimisation, Sustainability, Food Waste, Transportation, Metaheuristics

Resumo

Um dos grandes problemas atuais é a quantidade de alimentos que são desperdiçados em vez de serem aproveitados para alimentar pessoas. Esta questão urgente tornou-se um Objetivo de Desenvolvimento Sustentável, 12.3, que diz que "até 2030 as perdas de alimentos ao longo das cadeias de produção e abastecimento devem ser reduzidas para metade". Uma das principais causas deste desperdício de alimentos na cadeia de abastecimento é o transporte. Além do desperdício de alimentos, o transporte também impacta negativamente o meio ambiente. Assim, o principal objetivo desta dissertação é abordar este problema através do desenvolvimento de algoritmos de otimização de rotas, ligando produtores de frutas e legumes e consumidores, tendo em conta o desperdício alimentar e o impacto ambiental. A questão de investigação relacionada com isto é "Como melhorar a otimização de rotas considerando a sustentabilidade e a minimização do desperdício alimentar?". Ao considerar estes fatores, pode resultar no desenvolvimento de rotas de transporte mais ambientalmente conscientes e na diminuição do desperdício de alimentos. Primeiro, foi feita uma revisão sistemática da literatura para reunir o estado da arte sobre o assunto. Durante esta parte da investigação, não foi encontrada qualquer outra tentativa de combinar desperdício alimentar e sustentabilidade ambiental na otimização de rotas. Posteriormente, foram desenvolvidos os algoritmos, utilizando Hill Climbing, Simulated Annealing, Tabu Search e Genetic Algorithm com um conjunto de dados gerado. O projeto contém uma abordagem base, um recurso de otimização de alocação de alimentos, uma opção para alocar alimentos próximos à data de consumo para consumidores capazes de receber alimentos com esta caraterística, e também há a possibilidade de calcular rotas para vários dias seguidos. Além disso, as diferentes modalidades e algoritmos foram testados com diferentes parâmetros, correndo cada um três vezes e registando os resultados, o tempo que levou em segundos e o desvio padrão. Em cada algoritmo, foi analisada a influência de cada parâmetro. Além disso, foi feita uma comparação entre os quatro algoritmos para a opção base e a otimização da alocação de alimentos também foi analisada, comparando esta característica entre os diferentes algoritmos e com os resultados base. A opção de alocar alimentos perto da data de consumo também foi testada, comparando os resultados entre os quatro algoritmos, bem como a opção de calcular rotas por vários dias seguidos. Diferentes variações da função tweak também foram testadas para entender qual funcionou melhor. Para a maioria das modalidades, o algoritmo que funcionou melhor foi o Simulated Annealing, seguido pelo Tabu Search, Hill Climbing e, por fim, o Genetic Algorithm. A diversidade e a eficiência da solução foram discutidas, bem como os desafios enfrentados ao desenvolvê-la. Finalmente, concluiu-se que a dissertação atingiu o seu objetivo e foi capaz de responder à questão de investigação, apresentando com sucesso uma solução que incorporou um fator económico, um fator de desperdício alimentar e um fator de impacto ambiental no cálculo das rotas. Contribuiu para a área ao apresentar uma primeira tentativa de combinar estes fatores na otimização de rotas e foram também discutidas as perspectivas de trabalho futuro para complementar este projeto.

Palavras-chave: Otimização da rota do veículo, Sustentabilidade, Desperdício Alimentar, Transporte, Metaheurísticas

Acknowledgements

Firstly, I would like to thank Professors Luís Paulo Reis and Brígida Mónica Faria for supporting me throughout these past few months. Their help was essential in developing this project.

Furthermore, I would like to thank my parents and my family for guiding me through the years and encouraging my education. I would like to extend a special thank you to Cândido and Bianca for being the best company a student can ask for.

I would also like to give a shout out to all my teachers during my developing years that cultivated my interest and motivated me to pursue an engineering field.

Additionally, I would like to thank Board of European Students of Technology for not just being a student organisation, but a second home during four years.

Finally, I would like to thank my friends, the ones from secondary school, and the ones from FEUP, that provided me with unconditional support and companionship during this time of my life. A special thank you to Ana and Beatriz for being a second family to me.

Leonor Gomes

“Não tenhamos pressa,
mas não percamos tempo.”

José Saramago

Contents

List of Figures

List of Tables

Abbreviations

FAO	Food and Agriculture Organization
LCA	Life Cycle Assessment
API	Application Programming Interface
VRP	Vehicle Routing Problem
GVRP	Green Vehicle Routing Problem
EVRP	Electric Vehicle Routing Problem
EV	Electric Vehicle
GHG	Greenhouse Gas
FLW	Food Loss and Waste
SFSC	Short Food Supply Chain
SLR	Systematic Literature Review

Chapter 1

Introduction

This chapter alludes to the context of the dissertation, the motivation behind it, the problem at hand, the objectives, and finally, the document structure.

1.1 Context

The food industry is currently facing a substantial food waste problem that further negatively impacts the environment while, at the same time, people go hungry. 400 USD are estimated for losses between harvest and distribution of food [1]. Furthermore, according to FAO estimates, almost one-third of the food produced never makes it to the consumer.[1]. Moreover, "around 14 percent of the world's food is lost after harvesting and before reaching the retail level, including through on-farm activities, storage, and transportation"[2]. Additionally, according to the United Nations (UN), around 815 million people worldwide are struggling with hunger, and getting rid of current food losses would be sufficient to supply the caloric requirements of 1.9 billion people. [3]. Due to these issues, one of the Sustainable Development Goals(12.3) is the following "By 2030, food losses along production and supply chains must be halved." [4].

Furthermore, the transportation of goods also profoundly impacts the environment. The greenhouse gas emissions generated by vehicles, energy use, and water use by transportation negatively affect the environment. According to the International Energy Agency [5], the transportation sector is responsible for approximately 24% of global greenhouse gas emissions. Additionally, in 2019, transportation-related CO2 emissions reached 10.3 billion tonnes, with road transportation accounting for about 80% [6]. Moreover, the World Health Organization [7] estimates that outdoor air pollution from transportation causes around 1.2 million premature deaths each year. Finally, the transportation sector is one of the biggest energy consumers, with fossil fuels being the primary energy source. In 2018, transportation consumed 32% of global final energy consumption [8]. Hence, it is urgent to fight food waste and decrease the environmental impact and world hunger.

1.2 Motivation

This dissertation proposal is inserted in the project RETAILL - REtail using Technology based on Artificial InteLLigence which is within the scope of Eureka Clusters - Sustainability Call [9]. It aims at developing a modular platform flexible to most European countries food supply chains. In addition, the system will improve the food life cycle, promote circular economy and logistics efficiency, reduce the use of resources, and increase the profit of all actors in the value chain. Furthermore, it will empower and increase the bargaining power of small participants, such as producers and retailers, by creating a network between them and consumers. Finally, consumers will have access to higher-quality and safer products, determined through a freshness index for fruit quality assessment [10].

The benefits will touch the environmental, social, and economic scopes by reducing food waste, emissions, environmental noise, and overproduction in agricultural lands. Furthermore, it will increase the revenue for small and local producers and retailers, promote the remuneration of producers based on the quality of their production practices, increase the quality and safety of food products, and decentralize the food supply chain. Finally, it will reduce time and fuel spent on consumption, reduce energy for food conservation, increase the value of food products that otherwise would be disposed of, trigger food processing innovation using new food products, order optimisation for logistics companies, and optimised inventories for retailers. The features planned are route optimisation and consequently reduction of the environmental impacts and improving products' shelf life, time management between production and final client, strengthening of local food systems and selection of products according to agriculture practices, increasing the availability and affordability of healthy and, sustainable food options, making use of food waste and assessment of product life cycle. Furthermore, previous work on related projects from the Artificial Intelligence and Computer Science Laboratory at University of Porto can be found in articles [11], [12], [13], [14], [15] and [16].

1.3 Problem

As seen by the context, transportation in the food supply chain not only negatively impacts the environment but also causes a significant amount of food waste. Therefore, there is an opportunity to tackle both these dimensions while transporting produce. The research question to fit the problem created by the context is "How to improve route optimisation considering sustainability and food waste minimisation?".This research question explores innovative approaches and strategies integrating sustainability considerations and food waste minimization into route optimisation models and algorithms. By explicitly incorporating these crucial factors, it can develop more environmentally conscious and socially responsible practices within the transportation and logistics sector.

Existing studies on route optimisation have primarily focused on minimizing distance, reducing delivery time, or optimising vehicle utilization, mostly neglecting the sustainability aspects and the specific challenges related to food waste reduction. However, given the global concern for climate change, resource conservation, and the urgency to tackle food waste, it is imperative to extend the traditional route optimisation frameworks to embrace these critical dimensions. This question bears economic and resource efficiency considerations due to the money lost through food waste, fuel lost in inefficient routes, and social responsibility by addressing needed societal changes.

1.4 Objectives

The main goal of this dissertation is to contribute to the project by building a route optimisation system tailored for the transportation of fruits and vegetables from food producers to consumers, with a specific emphasis on addressing food waste and environmental variables. More specifically, this dissertation proposes to study route optimisation algorithms to identify the most suitable approaches for transporting perishable goods, such as fruits and vegetables. Furthermore, it proposes exploring ways to minimize the environmental impact of the transportation process and investigate strategies such as optimising routes to reduce fuel consumption and emissions and considering alternative transportation modes. Additionally, exploring opportunities for utilizing food waste generated within the supply chain is an objective. The final step is developing a comprehensive route optimisation system that incorporates the findings from the previous objectives. The system should efficiently calculate optimised routes, considering the specific constraints of food producers and consumers while incorporating food waste and environmental variables.

1.5 Document Structure

Besides the Introduction, the document comprises the State of the Art chapter, which illustrates a literature review on the topic, and the Methodology chapter, explaining how the proposed solution was developed. Furthermore, the Experiments and Results chapter presents the experiments carried out and their results and the Discussion chapter analyses these results and some implementation details. The Conclusion summarizes the work and reflects on the contributions and future work. Finally, the References have everything cited and referred to in this document.

Chapter 2

State of the Art

The State of the Art chapter introduces the methodology used to gather information, the current technologies for the problem and the systematic literature review. This review tackles route optimisation algorithms, sustainable route optimisation, electric vehicle route optimisation, food waste, and sustainable agriculture.

2.1 Methodology

A systematic literature review was performed to understand the topic in question better. The review focused on five subtopics: Route Optimisation Algorithms, Sustainable Route Optimisation, Electric Vehicle Route Optimisation, Food Waste, and Sustainable Agriculture. Furthermore, research on the current technologies was made outside of the systematic literature review. Below, there is an explanation of the procedure of the systematic literature review.
The first step carried out was to define the research question. It was decided that "How to improve route optimisation considering sustainability and food waste minimisation?" was a pertinent question. Next, the databases, keywords, and search strategies were decided. In the search strategy, it was concluded that the best approach was to look for review articles, thus having more prominent access to further articles and a better understanding of the scope of the problem. Additionally, the subsections of the systematic literature review were defined. The databases used for this research were Scopus and Google Scholar, and the keywords were:

- sustainable "route optimisation" AND (LIMIT-TO DOCTYPE , "re"))
- "food waste" AND "route optimisation" AND (LIMIT-TO (DOCTYPE , "re"))
- "green vehicle routing problem" AND (LIMIT-TO (DOCTYPE ,"re")

After this step, the exclusion criteria were decided. The exclusion was carried out in two stages. The first stage was during the results of the databases, and articles were selected based on the first impression of their relevance, analysing their title, keywords, and abstract. The second stage was done afterward, considering the number of citations and further relevance by reading

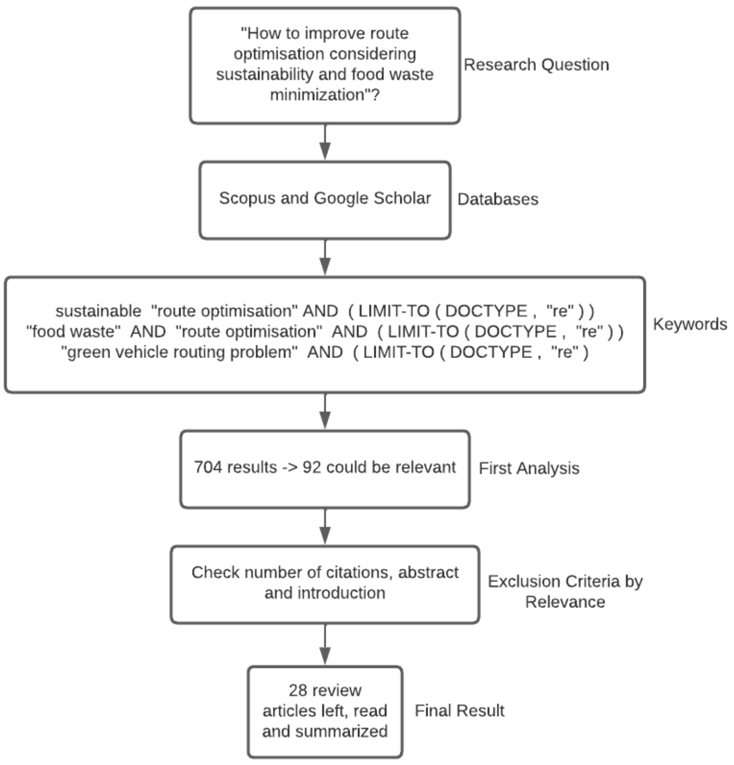

"How to improve route optimisation considering sustainability and food waste minimization"?
Research Question
Scopus and Google Scholar
Databases
sustainable "route optimisation" AND (LIMIT-TO (DOCTYPE , "re"))
"food waste" AND "route optimisation" AND (LIMIT-TO (DOCTYPE , "re"))
"green vehicle routing problem" AND (LIMIT-TO (DOCTYPE , "re")
Keywords
704 results -> 92 could be relevant
First Analysis
Check number of citations, abstract and introduction
Exclusion Criteria by Relevance
28 review articles left, read and summarized
Final Result

Figure 2.1: Diagram of the systematic literature review process

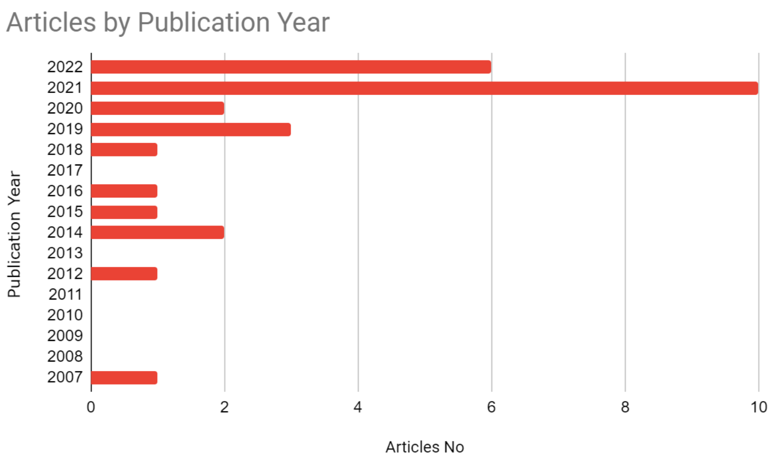


Figure 2.2: Number of review articles read by year

the abstract and introduction. From the first analysis, out of 704 articles, 92 were considered they could be relevant. The second analysis was performed, and in the end, 28 articles were considered relevant, read, and summarized. The process is described in figure 2.1.

Furthermore, the year each article was published in figure 2.2 and the countries where the authors of the articles worked in figure 2.3 were analysed.

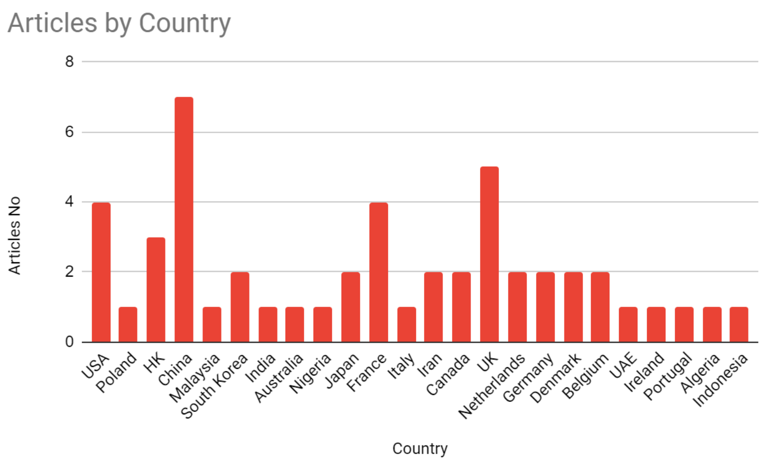


Figure 2.3: Number of review articles read by country

Regarding the countries for each article, it was taken into account the different countries where the authors worked and added to the count as one. For example, if an article had three authors, two from different institutions in France and one from the UK, the article would count as one for France and one from the UK.

2.2 Current Technologies

In order to understand what is already on the market, an exploration of the current platforms was made. The goal was to find valuable tools that helped farmers connect with the consumers, improve sustainability in food production and transportation, and optimise routes in transportation. Moreover, tools that helped with food waste and life cycle assessment software were assessed as well. The research on current technologies focused on e-commerce platforms related to agriculture, route optimisation services, food waste platforms, and LCA software.

2.2.1 E-commerce Platforms

Part of the research made focused on e-commerce agricultural platforms. The following characteristics were taken into account:

- Help managing inventory - provides tools to manage the inventory of farms;
- Connect sellers and buyers - the platform allows for consumers to buy products from sellers;
- Help with business and marketing - assists on a business and marketing level;
- Information on sales data - shows data on how sales are going;
- Product Delivery - takes care of delivery to consumers;
- Sustainable measures in pre-harvest - applies sustainable measures before harvesting;

- Fair-trade between farmers and industrial buyers - guarantees fair-trade;

Furthermore, the following **platforms** were considered:

- 1000 Ecofarms - an online marketplace of local natural foods, connecting farmers/restaurants and consumers [17];
- Barn2Door - a software solution that helps farmers with their business [18];
- CSAware - a product that assists farmers in managing their business [19];
- EatFromFarms - farm store builder allowing farmers to connect with customers and improve sales [20];
- Farmers Web - tool to help farmers manage sales, customers, and records;
- Harvie - online subscription service for buying groceries and products from local farmers and producers [21];
- FaST - digital service platform aiming to make available capabilities for agriculture, environment, and sustainability based on space data and other public and private data [22];
- Community Supported Agriculture - "a partnership between farmers and consumers in which the responsibilities, risks, and rewards of farming are shared" [23];
- Agri Marketplace - a fair-trade platform that connects farmers to industrials worldwide [24].

The table 2.1 is a review of the characteristics each platform has.

Table 2.1: E-commerce Platforms

Platform	MI	CSB	BMKT	SD	PD	SPH	FT
1000 EcoFarms	x	✓	✓	✓	x	x	x
Barn2 Door	x	x	✓	x	✓	x	x
CSAware	✓	✓	✓	✓	✓	x	x
EatFromFarms	✓	✓	✓	✓	✓	x	x
Harvie	x	✓	x	x	x	x	x
FaST	x	x	x	x x	✓	x	
CSA	x	✓	x	x	x	x	x
Agri Marketplace	x	✓	x	x	✓	x	✓

MI - Manage Inventory; CSB - Connect Sellers and Buyers; BMKT - Business and MKT; SD - Sales Data; PD - Product Delivery; SPH - Sustainability Pre-Harvest; FT - Fair Trade

The most common characteristics by order are connecting sellers and buyers (6), help with business and marketing (4), product delivery (4), information about sales data (3), managing inventories (2), fair trade (1), and sustainable measures pre-harvesting (1).

2.2.2 Route Optimisation Services

There are, currently, several route optimisation services available. For this research, these features were considered:

- Optimise delivery routes - route optimisation;
- Estimated time of arrival (ETA) - calculates ETA;
- API Engine - provides an API;
- Customer notifications - sends notifications to costumers;
- Live tracking - has live tracking of the transportation
- Delivery analytics - has data on delivery;
- Proof of delivery - has an implemented method to prove delivery ;
- Workload balance - takes into account the workload of drivers;
- Real-time (RT) adjustments - allows real-time adjustments;
- Driver preferences - takes into account driver preferences;
- Electric vehicles (EV) charging - calculates route considering EV are part of the fleet and their need for charging;
- Alternative routes - shows alternative routes.

Regarding the services considered, these were the ones studied:

- Routific - a software platform that provides route optimisation solutions for delivery fleets and field service businesses[25];
- OptimoRoute - a cloud-based route optimisation platform designed to help businesses with delivery and mobile workforce management by providing advanced algorithms for planning and scheduling optimised routes [26];
- RoadWarrior - a cloud-based route planning and optimisation software that provides features such as multi-stop route planning, real-time traffic updates, and customizable route optimisation algorithms for businesses and individuals [27];
- AntsRoute - a fleet management software that offers route optimisation, real-time tracking, and customizable data visualization tools for businesses with delivery, collection, and service fleets [28];
- Verizon Connect - platform that offers a range of features including vehicle tracking, fuel management, driver behavior monitoring, and route optimisation for businesses with mobile workforces [29];

- Onfleet - cloud-based delivery management software that offers route optimisation, real-time driver tracking, and customer communication tools for businesses with local delivery operations [30];
- Route4me - fleet management software that offers route planning and optimisation, real-time tracking, and mobile app integration for businesses with delivery, sales, and service fleets [31].

These services are all route optimisation businesses.
The next table 2.2 encompasses what each service provides in terms of features.

Table 2.2: Route Optimisation Services

Service	ODR	ETA	APIE	CN	LT	DA	PoD	WB	RTA	DP	EVC	AR
Routific	√	√	√	√	√	√	√	x	x	x	x	x
Optimo Route	√	√	√	√	√	√	√	√	√	x	x	x
Road Warrior	√	x	√	x	x	√	x	x	√	√	x	x
AntsRoute	√	x	x	√	x	√	√	x	x	x	√	x
Verizon Connect	√	√	√	x	√	√	x	x	√	x	x	√
Onfleet	√	√	√	√	√	√	√	x	√	x	x	x
Route4me	√	x	√	√	√	√	x	√	x	x	x	x

ODR - Optimise Delivery Routes; ETA - Estimated Time of Arrivel; APIE - API Engine; CN - Customer Notifications; LT - Live Tracking; DA - Delivery Analytics; PoD - Proof of Delivery; WB - Workload Balance; RTA - Real Time Adjustments; DP - Drive Preferences; EVC - EV Charging; AR - Alternative Routes

The most popular features by order are optimise delivery routes (7), delivery analytics (7), API engine (6), customer notifications (5), live tracking (5), estimated time of arrival (4), proof of delivery (4), workload balance (2), driver preferences (1), electric vehicles charging (1) and alternative routes (1). After taking this into account, it becomes easier to know which features are essential such as the optimisation of delivery routes and delivery analytics, and the ones that can make a service stand out, such as considering electric vehicle charging.

2.2.3 Food Waste Platforms

Currently, there are a lot of platforms on the market that help with food waste. An analysis of said platforms was carried out, and the following characteristics were considered:

- Sell food close to consumption date - sells food that is close to consumption preferred date but it is still safe to eat;

- Sell surplus food - sells excess food that would otherwise get wasted;
- Donate food - donates food to people in need;
- Include transportation - has a transportation system to the final consumer;
- Share information - uses their platform to educate on food waste;
- Check consumption limit of products - alerts when food products are close to the consumption limit;
- Plan household food consumption - provides assistance with for example shopping lists, recipes with specific ingredients, etc;

The difference between selling food close to the consumption date and selling surplus food is in the context. Restaurants/markets/bakeries can have surplus food at the end of the day (for example, prepared meals), and supermarkets/markets can have food near the end of the preferred consumption date (for example, yogurts).

The platforms taken into account were:

- Too Good To Go - a mobile application that connects restaurants and markets with surplus food with customers[32];
- Goodafter - online supermarket that sells products that are close to the end of the preferential consumption period or surpassed [33];
- Olio - a mobile application that lets users upload food that they won't need so it gets donated [34];
- FlashFoods - a mobile application that connects consumers with supermarket products that are close to the best-before date with discount prices [35];
- Food Rescue US - a platform connecting food donors from local stores that donate their excess food with people in need [36];
- Karma - a platform that enables users to rescue fresh food from restaurants, bakeries, and cafes that would have been thrown away [37];
- FoodCloud - a platform with the goal of connecting businesses with surplus food with charities and community groups that need it [38];
- NoWaste - a mobile application that checks consumption dates and helps with planning household food consumption [39];
- nosh - a mobile application that checks consumption dates and helps with planning household food consumption [40];

- Kitche -a mobile application that checks consumption dates and helps with planning household food consumption [41];
- Fridgely - a mobile application that checks consumption dates and helps with planning household food consumption [42];

The following table 2.3 shows the features included in each product.

Table 2.3: Food Waste Platforms

App	SCCD	SS	D	T	I	CCD	PHC
Too Good to Go	x	✓	x	x	✓	x	x
Goodafter	✓	x	x	✓	x	x	x
Olio	x	x	✓	✓	x	x	x
Flashfoods	✓	x	x	x	x	x	x
Food Rescue US	x	x	✓	✓	x	x	x
Karma	x	✓	x	x	x	x	x
FoodCloud	x	x	✓	x	x	x	x
NoWaste	x	x	x	x	x	✓	✓
nosh	x	x	x	x	x	✓	✓
Kitche	x	x	x	x	x	✓	✓
Fridgerly	x	x	x	x	x	✓	✓

SCCD - Sell Close to Consumption Date; SS - Sell Surplus; D - Donate; T - Transportation; I - Info; CCD - Check Consumption Date; PHC - Plan Household Consumption

From these tables, it can be concluded that there are two types of food waste apps: the ones that sell/donate excess food and the ones that help manage food waste from a household perspective. Too Good to Go, Goodafter, Olio, Flashfoods, Food Rescue US, Karma and FoodCloud belong to the first type of apps and NoWaste, nosh, Kitche, and Fridgerly to the second type of apps. The most common features of the first type of app are donate food (3), include transportation(3), sell surplus food (2), sell food close to the consumption date (2), and share information (1). The second type of app checks the consumption limit of products and helps plan household food consumption in all examples.

2.2.4 Life Cycle Assessment

Life Cycle Assessment (LCA) is applied to evaluate the environmental impact of a product over its life cycle. It can be used to analyse the contribution of each life cycle stage to the overall environmental load and, afterward, be able to prioritise improvements. Furthermore, it can be used as a comparison method between products [43].

An LCA study is divided into 4 stages [43].

Stage 1: Goal and Scope determines how much of the product life cycle will be considered in the evaluation and how the evaluation will be carried out. It also specifies the standards used to compare systems.

Stage 2: Inventory Analysis describes the materials and energy flows of the system, the interactions with the environment, mainly raw materials usage and emissions.

Stage 3: Impact Assessment details the impact of each category and assesses it by normalizing or weighting methods.

Stage 4: Interpretation reviews the results, determines data sensitivity, and presents results.

Calculating the LCA of a product can assist with searching for life cycles with a minimal negative impact on the environment, taking decisions, deciding on the focus and priorities for strategic planning, creating new products or modifying existing processes, and, finally, marketing / eco-labeling [43].

2.2.5 LCA Software

Five software tools that calculated the evaluation of a product's life cycle were analysed. The first one is Simapro, a software tool that provides insights into the environmental performance of products. It models and analyses complex life cycles, measures the environmental impact of products and services across all life stages, and identifies the hot spots of every link in the supply chain [44]. The second one is Sphera which offers a holistic view of the product's energy consumption, ecological hot spots, and the resulting environmental impacts. Moreover, it calculates the product's life cycle and compares it with alternative scenarios. It also provides a database that details the costs, energy, and environmental impact of sourcing and refining materials. Finally, it presents greener alternatives for manufacturing, distribution, recycling, pollution, and sustainability [45]. Furthermore, openLCA is an open-source tool that calculates sustainability and life cycle assessments, showing analysis results and identifying main drivers through the life cycle. Life cycle costing and social assessment are also integrated [46]. In addition, Ecochain offers two solutions to estimate the life cycle assessment. Ecochain Helix measures the impact of large product portfolios, estimating each LCA and analysing the environmental insights on multiple levels. Ecochain Mobius, however, measures a product's environmental footprint, compares different products and materials, and facilitates the design of more sustainable products [47]. Finally, iPoint is a software tool that offers solutions for the entire product life cycle, including carbon footprint, life cycle assessment, circular economy, design for sustainability and compliance solution, and digital product passport [48].

2.3 Route Optimisation Algorithms

In this section, an introduction to the Vehicle Routing Problem will be presented, including its variants. Additionally, the methods used to solve this specific problem will be showcased.

Previous work conducted by the Artificial Intelligence and Computer Science Laboratory at University of Porto focused in generation and optimisation of inspection routes for economic and food safety, partnering with the Portuguese Food and Economic Safety Authority (ASAE). Findings can be found in articles [49], [50], [51], [52], [53]. In the case of the current project, the goal is to optimise routes as well but from producers of fruits and vegetables to consumers taking into account food waste and environmental factors, taking inspiration from this previous work in developing the solution.

2.3.1 Introduction to the Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a linear optimisation problem, common in logistics and transport, that has the goal of figuring out how to use a fleet of trucks with different capacities to cater to a group of customers who are geographically spread out around a central depot [54] [55]. It is one of the most widely studied problems in the literature of Operations Research and, over the years, the approaches have become more advanced in order to accommodate real-life complexities [54].

2.3.2 VRP Variants

In article [56], a summary of various Vehicle Routing Problem (VRP) variants was presented. These variants represent different problem settings and constraints in the VRP. For instance, the Capacitated VRP variant explicitly considers the capacity of each truck [57], while the Time-Dependent VRP variant takes into account both the distance and the time of day to predict travel time [58, 59]. The Pickup and Delivery Problem has paired pickup and delivery points, and there are several variants of this problem [60, 61]. The Multi-Depot VRP variant contains more than one depot, and vehicles are assigned to them [62, 63]. In contrast, the Stochastic VRP variant assumes that some elements such as travel times are random [64].

Other variants include the Location Routing Problem, which decides whether to open one or more depots and calculates the route after that [65, 66]. In contrast to the Dynamic VRP, which is a dynamic operation where client requests are published throughout the planning period and assigned in real-time[67, 68], the Periodic VRP version finds the most efficient path and schedules the tours for specific days of the week [69, 70]. The Inventory Routing Problem variant guarantees that there are no stock-outs at each customer [71, 72]. The Fleet Size and Mix VRP variation also selects the fleet's most cost-effective combination of vehicles[73, 74].

The Generalized VRP variant divides customers into clusters, and only one customer in each cluster is visited [75]. The Multi-compartment VRP variant deals with goods that are inhomogeneous and non-inter-mixable, so they need to be in different compartments of the vehicle [76, 77]. The Site-dependent VRP version limits the amount of vehicle types that can visit each site to a subset rather than all of them [78]. The Split-delivery VRP variation enables multiple vehicles to cater to each client [79, 80]. In addition, the Fuzzy VRP variation formulates subjective, unclear, and imprecise parts using fuzzy logic [81, 82].

The Open VRP variation doesn't demand vehicles to come back to the depot once the route has been completed [83, 84], the VRP with Loading Constraints variant calculates the best paths within packing restrictions [85, 86], the Multi-echelon VRP variant studies the movement of flows in a multi-echelon distribution strategy [87], and the VRP with Time Windows variant takes into account service time intervals and due dates, being divided in Hard Time Windows (hard constraint) and Soft Time Windows (soft constraint with penalty) [88], [89].

2.3.3 Algorithms used in VRP

Either exact or approximate algorithms are used to solve VRP problems [56],. Exact algorithms reach the optimal solution however they can only tackle problems on a small scale. Comparatively, approximate algorithms tend to be used in practice as they can quickly provide satisfactory answers to complex issues. Exact algorithms are classified into three categories, direct search tree algorithms, dynamic programming, and integer linear programming [90]. Approximate solutions include classic heuristics and metaheuristics. The categories for classical heuristics are saving algorithms, sequential improvement algorithms, sweep algorithms, petal algorithms, and Fisher and Jaikumar two-phase algorithms [91] [92] .

Metaheuristics, compared to classic heuristics, perform a deeper search of the solution space, thus being able to reach better solutions, in spite of bigger computation costs [92]. Metaheuristics can be classified in two types[56], local search and population search. Examples of local search algorithms are tabu search (TS), simulated annealing (SA), greedy randomized adaptive search procedure (GRASP), variable neighborhood search (VNS), and large neighborhood search (LNS). For population search, some examples are genetic algorithms (GA) and ant colony optimisation (ACO).

2.3.4 Other Technlogies

Apart from exact and approximate algorithms, other technologies are starting to be applied to these types of problems. For example, article [93] developed a state-of-the-art review on the applications of blockchain and AI in transportation systems, including the advantages and challenges. Furthermore, article [94] discussed various applications of AI in transportation systems, including knowledge-based systems (KBS), neural networks (NN), and fuzzy systems. It also arrived at the conclusion that AI technologies are expected to improve transportation safety and environmental compatibility in the future.

2.4 Sustainable Route Optimisation

Over the years, the interest in more sustainable approaches has increased and route optimisation was no exception. The demand for environmental protections and the responsible use of natural resources has motivated companies to take into account environmental concerns in their decisions [95]. Thus, the interest in green route optimisation has increased. In addition, one of the biggest impacts caused by transportation is the over-exploitation of resources, such as energy and water. The use of energy specifically increases pollution with the release of greenhouse gases, contributing majorly to climate change and global warming. This article [96] has also added air pollution, noise, land use and safety hazards including accidents as negative impacts of transportation. Thus, this section will present an introduction of the Green Vehicle Routing Problem (GVRP), the environmental impact factors, and different instances of GVRP.

2.4.1 Introduction to Green VRP

The Green Vehicle Routing Problem (GVRP) has the goal of harmonizing both the environmental and economic costs by implementing effective routes to meet environmental concerns, decreasing its negative impact, and financial objectives [56].

2.4.2 Environmental Impact Factors

In transportation, one of the factors that influences the environmental impact is fuel consumption. Furthermore, the study [97] details different emission models. There are "numerous energy and emission models available and they differ in their modeling approaches, modeling structures, and data requirements" [98]. The initial study [97] categorizes fuel consumption models in increasing levels of complexity. First, only basic fuel consumption techniques are included in the factor models. Then, the microscopic models estimate immediate vehicle fuel use and emission rates at a more precise level, while the macroscopic models estimate network-wide emission levels using average aggregate network variables.

This study [97] then concludes that vehicle speed is quite crucial for fuel consumption and it's critical to travel at a speed that minimizes fuel consumption. Furthermore, medium-duty cars should not be chosen over light-duty vehicles and medium-duty should be preferred over heavy-duty vehicles. At last, a positive road gradient should be considered when planning a route due to the fact fuel usage rises as a result of it. Finally, the article [97] reiterates the suggestion of using multi-objective optimisation in green road transportation considering the several factors that impact the environment.

2.4.3 Different Instances of Green VRP

In article[56], there are three categories for GVRP. The first one is Green-VRP which optimises the energy consumption of transportation [99]. A vehicle dispatching plan with reduced pollution, specifically a decrease in carbon emissions, is settled upon by The Pollution Routing Problem [100], [101]. Finally, VRP in Reverse Logistics deals with the distribution aspects of reverse logistics. Reverse logistics is the process of organizing, implementing, and managing backward streams of raw supplies, in-process stock, packaging, and final products from a manufacturing, distribution, or application point to a point of recovery or a point of proper disposal [102], [103].

2.4.4 Algorithms for Green VRP

The article [104] analysed the algorithms used for Green VRP in a universe of 309 articles and checked which strategies were more popular. It was concluded the most common strategy is Metaheuristics (41%) followed by Exact algorithms (18%), Exact Solver (16%), Hybrid (14%), Heuristic (7%) and Software Application (4%). As stated by study [105], metaheuristic approaches receive more attention since they are a quicker method compared to exact methods, that fail to reach a good solution in a reasonable amount of time.

Additionally, the article [104] also checked the variables that were used in Multi-Objective strategies for Green VRP. The most common variables taken into account are Emissions (21%), Cost(19%), and Social Factors(17%). As an example, in study [106], they have three goals, that are decreasing the operational costs of travel, the fuel consumption based on the distance traveled and taking into account the load of the vehicle and increasing customer satisfaction. Another example is the article [107], that wants to minimize the distance traveled and the CO2 emissions. In this case, the estimation of CO2 also takes into consideration the state of the vehicle, whether it is empty, lowloaded, halfloaded, high loaded and fully loaded.

2.5 Electric Vehicle Route Optimisation

Electric vehicles are a sustainable solution to use for transportation compared to fossil fuel vehicles. There has been a rise in interest in applying this technology and it's directly connected with green route optimisation. This section will go over the relevance of electric vehicles, the Electric Vehicle Routing Problem (EVRP) and its variants.

2.5.1 Relevance of Electric Vehicles

Electric cars feature various eco-friendly characteristics, including fewer GHG emissions, improved energy efficiency, and decreased noise pollution[108]. Thus, the adoption of these vehicles provides a good chance to improve the environment [108], [109], [110], [111].

The transportation industry emits a considerable amount of CO2, impacting negatively the environment. The use of electric vehicles and, consequently, the decrease of CO2 emissions compared with other vehicles, will help reduce the negative impact on the environment. The article also noticed an increase in interest in purchasing electric vehicles. Hence, one can conclude that betting on this technology as a tool for a greener transportation industry is a good idea [112].

However, it should be noted that there are some cons compared to typical fossil fuel vehicles. For example, the article [113] mentioned the long charging time of EV batteries, the low energy density of batteries, and the scarcity of public and/or private charging stations for EVs. Additionally, the article [114] also concluded that the use of electric vehicles for transportation should "not be automatically associated with profitability, but rather with exploration and preparation in anticipation of expected future developments".

2.5.2 Introduction to EVRP

The main difference between EVRP and VRP is the need to consider the electric vehicle's mileage, battery capacity, charging/swapping station time window, and the vehicle characteristics and power performance as well as the fuel consumption constraints of the VRP [115]. These added constraints make the EVRP more challenging than the traditional VRP. Additionally, while developing a solution for EVRP, researchers need to take into account additional factors such as vehicle capacity/load capacity, vehicle energy usage, accessible charging stations, charging technology employed, charging station capacity limitation, grid capacity constraint, immediate price of electricity and charging cost [108].

2.5.3 EVRP Variations

In article [108], three variations of EVRP were introduced. The first one is EVRP with charging where it takes into account the charging location, with the possibility to charge at the place of the customers, depots and charging stations. It also considers the capacity of the charging stations. The second variation is EVRP with a time window (EVRP-TW), both with hard and soft time windows. Finally, the last variation is EVRP with a mixed fleet (EVRP-MF). This can comprise identical types of EVs and fossil-fuel vehicles, mixed types of EVs and fossil-fuel vehicles and mixed types of EVs.

2.6 Food Waste

Food Waste has become a relevant problem for today's world. Sustainable Development Goal (SDG) 12.3 states that by 2030, global food losses should be reduced by half throughout the entire food value chain, including post-harvest losses and the end consumer [4].

Food loss is "the decrease in the quantity or quality of food resulting from decisions and actions by food suppliers in the chain, excluding retail, food service providers and consumers". Food waste, on the other hand, is "the decrease in the quantity or quality of food resulting from decisions and actions by retailers, food services and consumers"[116], [117]. However, some authors don't make any distinction between them [118].

Preventing a surplus of food or loss throughout production and consumption along the food value chain is the first step in achieving the goal of reducing food waste. Reuse encompasses the actions done to give surplus food to those in need. The recycling stage includes using exceeding food for animal feed or composting and the recovery stage uses the surpluses to produce energy. Finally, the disposal takes care of eliminating the surpluses in a landfill [116]. Prevention has been the least promoted solution however it is the most environmentally favorable option to handle this problem and for long-term sustainable consumption and production as well [119], [120], and [121].

As explained in the food waste hierarchy, food surplus doesn't need to go to waste. For example, food donation can contribute to FLW reduction and social and environmental advancements

[122], [123]. Additionally, due to the uncertainty of demand and supply, fresh products could be frozen when possible to reduce food waste. However, it leads to a decrease in the value of the product [122], [124]. This section will be explaining the current impacts of food waste and the causes of food waste in transportation.

2.6.1 Impact of Food Waste

According to assessments from the Food and Agriculture Organization of the United Nations (FAO), about one-third of the food produced fails to reach consumption[116], [1]. Furthermore, food waste is an issue since it not only fails to feed those in need but also consumes a significant amount of resources [116], [125]. Additionally, halting ongoing food losses would be sufficient to satisfy 1.9 billion people's caloric needs at a time when, according to the United Nations (UN), 815 million people are hungry worldwide[116], [3]. Therefore, it is widely acknowledged that, rather than a lack of production, distribution problems cause the issues in the global food supply chain[116], [126].

Furthermore, the article [127] summarizes the environmental, social and economic impacts of food waste. For environmental impact, it notes the rise of methane emissions from greenhouses, the ineffective usage of agricultural land and water and the waste of non-renewable energy. For social impact, it reffers to the reduction in nutrition level, the greater susceptability to diseases caused by micronutrient deficits, the food prices rise, making access more difficult and the decrease in both salaries and output of labor. Regarding the economic causes, it considers the reduction in reveneu, the increase in expenditure for disposal and treatment of waste and the decrease of funds available for additional investment.

2.6.2 Causes of Food Waste in Transportation

In the shipment, storage, distribution, and wholesale stage, the specific causes of food loss in the fruits and vegetable sector are variations in temperature (strawberries, tomatoes, potatoes, etc), storage pushing (lettuce, apples, potatoes), weather-related sales variability (lettuce), demand forecast inaccuracy (lettuce, tomatoes, broccoli), storage deterioration (onions and broccoli), destructive quality control (onions and avocados), distance traveled (citrus), and, finally, shipping delays (bananas) [122], [124]. Furthermore, perishable products, like fruits and veggies, need management approaches and models that take into account additional challenges such as temperature controls, quality decay, or waste reduction methods [128], [129], [130].

During transportation, FLW is "influenced by the type of transport, infrastructures,distance between pickup and delivery and duration of moving and handling" [119]. The causes of FLW during transportation are expired products in transit, improper handling, inadequate packaging, interruption of the cold chain, inappropriate type and condition of transport, lack of protection of the products in the cargo space, presence of ramp and slipways on the routes and finally, lack of cold chain facilities [119] [131].

The article [131] continues by saying that it is critical to tweak the routes since poor product placement combined with intense vibration during transportation can result in damaging the packages. Additionally, a prolonged transportation process leads to the product having a shorter lifespan when it finally reaches the customer. This could be restrictive, particularly for foods that are easily perishable, like fruits and vegetables. In addition, the articles [132], [133] consider transportation the most critical step through the food journey due to the problems affecting the products during shipments and storage activities, especially in international and intercontinental trips.

2.7 Sustainable Agriculture

This section will briefly elaborate on the impacts of agriculture and two examples of alternative food systems.

2.7.1 Impact of Agriculture

The articles [134], [135] forecasted that the transport industry contributes to nearly half (48.5%) of the emissions in the alimentary production supply chain when a complete "farm to fork" analysis is made, which takes into consideration agriculture, food processing, distribution, and consumption. Furthermore, approximately 30% of the world's energy is used for food production. Additionally, eutrophication of water (more than 50%)[134], [136], biodiversity loss (60%) and land conversion (60%) as well as nutrient overload (70%) and climate change (30%) [134], [137] are all directly impacted by global food systems and the food and beverage businesses which sustain them.

Furthermore, given the majority of food needs to be refrigerated during transit, such refrigeration systems can consume up to 40% of the diesel. This may help clarify why freight transportation uses over 25% of all petroleum globally and contributes more than 10% of all fossil fuel-related carbon emissions [134], [138]. However, it has been concluded that delivering orders to homes through automobiles is more environmental friendly than having a lot of people go to the store individually by car. The delivery services go to all delivery points in one round-trip in the quickest path to each residence. Hence, the efficiency of the routes reduces the carbon footprint of food shopping [134], [139].

2.7.2 Regional Food Systems

One can then conclude that modern food supply systems face major environmental and social problems. Regional food systems, in which consumers favor nearby food suppliers, could contribute to finding a solution [140]. Additionally, while the majority of locally produced food is provided to stores, some of it is sold directly to consumers. The articles [140], [141], [142], [143] further add the perceived reasons why consumers would prefer regional foods, them being lower prices, fresher food, safer and more nutritious food, decreased dependence on oil and gas and long-distance travel, as well as the potential to boost the regional economy. This increase of interest for regional foods has the potential to be highly advantageous for small farms, which make up 85% of all regional food producers and have an annual income of less than $75,000 [140], [144].

Nonetheless, there are some disadvantages of direct producer-consumer contact, such as the preference from consumers to buy food in grocery stores and a lot of farmers struggle with the cost of transporting their produce to farmer markets [140], [144], [145]. In order for food regional systems to unlock their full potential, the respective producers must find a way to scale their operations and, by using institutional and retail outlets, they may reach a wider spectrum of clients with their products.

It should be also taken into account that, even though regionally produced foods travel shorter distances, due to the scaling that may be accomplished by long-distance freight movement of entire truckloads, regional food systems are typically less effective [140], [146]. The best practices for improving the efficiency of these regional systems are utilizing vehicles effectively, minimizing vacant backhauling, selecting the right vehicle, making regular and timely deliveries, utilizing third-party logistics providers' services, and establishing transportation partnerships.

2.7.3 Short Food Supply Chain

Short Food Supply Chain (SFSC) is "a supply chain involving a limited number of economic operators committed to cooperation, local economic development, and close geographical and social relations between producers, processors, and consumers" [147]. SFCS represents an alternative food system that is more sustainable due to the fact that low food miles (the distance food travels) of local food systems "imply lower fuel consumption, lower CO2 emissions, and fewer distribution phases" [148], [149], [150], [151].

As previously mentioned, the short-distance advantage cannot be enough to decrease the environmental impact if there are other factors present, like a "lack of an efficient coordination in transport logistics, the increasing use of private cars or the frequent journeys with low load factors required" [147], [152], [153], [154].

2.8 Implications and Future Directions

The study conducted gathered the current context of the Vehicle Routing Problem applied to sustainability and where food waste can be implemented. Considering the articles reviewed, route optimisation taking into account both food waste and sustainability hasn't been properly explored yet. Therefore, an opportunity arises in this field where the development of these algorithms could be useful for the industry. Furthermore, this chapter has shown the importance and the impacts of food waste in today's world. Factoring in the information presented, one can conclude that although the literature on green vehicle route optimisation is extensive and the interest in turning transportation more sustainable is increasing, there isn't yet a focus on putting together efforts to at the same time decrease food waste and environmental impacts of transportation. Hence, one of the future steps in the literature would be to develop route optimisation algorithms that included both food waste and environmental impact factors. Given the increase of interest in making operations more sustainable and reducing food waste, there is a huge potential in this field of research to develop these algorithms.

Chapter 3

Methodology

This chapter presents the methodology employed to define the problem, how the proposed method generates a solution, the features added to complement the solution, and the algorithms applied.

3.1 Problem Definition

This section thoroughly explains how the solution proposes the representation of the problem. The Classes Definition subsection showcases how each class is organized and what it represents and the Problem Creation subsection brings light on how the information comes together.

3.1.1 Classes Definition

In order to represent the problem, several classes were created. The FoodType class represents a type of produce, for example apples or oranges. It contains:

- id: integer;
- name: string;
- no_days: int - number of days since collected it takes to get too ripe;
- volume: float - individual volume of one product (example: one apple - m^3).

The Consumer class represents a consumer and has:

- id: integer;
- latitude: float;
- longitude: float;
- request: list - list of tuples with the structure (FoodType: FoodType, quantity:integer, date:datetime.datetime) being quantity the amount of products from FoodType and date the date that the consumer made the request;

- too_ripe: boolean - if it is true, it means that the consumer can receive close to end of consumption date produce.

The Producer class represents a producer and, similarly with the Consumer class, has:

- id: integer;
- latitude: float;
- longitude: float;
- stock: list - list of tuples with the structure (FoodType: FoodType, quantity:integer, days_rippen: integer, date:datetime.datetime) being quantity the amount of products from FoodType, days_rippen being the amount of days passed since it was ripped and date the date the product is available for transportation.

The Vehicle class represents the vehicles of the problem that transport the food and has:

- id: integer;
- fuel_type: string;
- consumption: float - consumption of fuel per meter;
- fuel_price: float;
- fuel_emissions: float - emissions of the fuel type per unit of consumption;
- capacity: float - the volume capacity of the car (m^3)

The Problem class represents the whole problem and contains:

- producers: list;
- consumers: list;
- vehicles: list;
- food_types: list.

The Food Delivery class represents a food delivery between a producer and a consumer and what the producer delivered to the consumer. It has:

- producer: integer - id of the producer;
- consumer: integer - id of the consumer;
- food_list: list - list of tuples with the structure (FoodType: FoodType, quantity: integer, days_rippen: integer), similar to the one in stock in Producer object;
- total_volume: float - the total volume corresponding to the food_list.

The CarRoute class represents a route that a car will do in one day. It contains:

- vehicle: Vehicle - vehicle that is performing the route;
- food_delivery_list: list - list of FoodDelivery objects that correspond to the ones being taken care of by the vehicle;
- prod_cons: dictionary - dictionary with the keys as producers (their id) and the values as a list of ids of consumers that the producers will deliver food;
- route: list - list of strings that represents the order where the car will pick up or deliver food.

Finally, the RouteSolution class represents a single solution, with the information of the CarRoutes. It has:

- car_routes: list - a list of the CarRoutes;
- vehicles_info: list - list of the Vehicles of the problem;
- producers_info: list - list of the Producers of the problem;
- consumers_info: list - list of the Consumers of the problem;
- food_type_info: list - list of the FoodTypes of the problem;
- dist_dict: dictionary - dictionary with the key as a tuple with (lat1: float, lon1:float, lat2:float, lon2:float) and the value dist:float that is the distance between points (lat1, lon1) to (lat2, lon2);
- date: datetime.datetime - date of the solution;

The figure 3.1 represents the classes of the proposed solution.

3.1.2 Problem Creation

With the use of the Problem class and a problem creation function, the program reads the information of the problem and creates the objects for Producer, Consumer, Food_Type, and Vehicle, the ids for producer and consumer are assigned to be sure there are no repeated ids, and assembles the lists containing the information. The Problem Class allows access to the information required to solve the problem efficiently and is then used to generate RouteSolution solutions.

3.2 Dataset

The first step to understanding which dataset to use was defining which variables were needed for the problem. Because this is a route optimisation problem taking into account food waste and environmental sustainability, not only was it vital to have information about the location of producers and consumers, but also information about the food that was going to be transported,

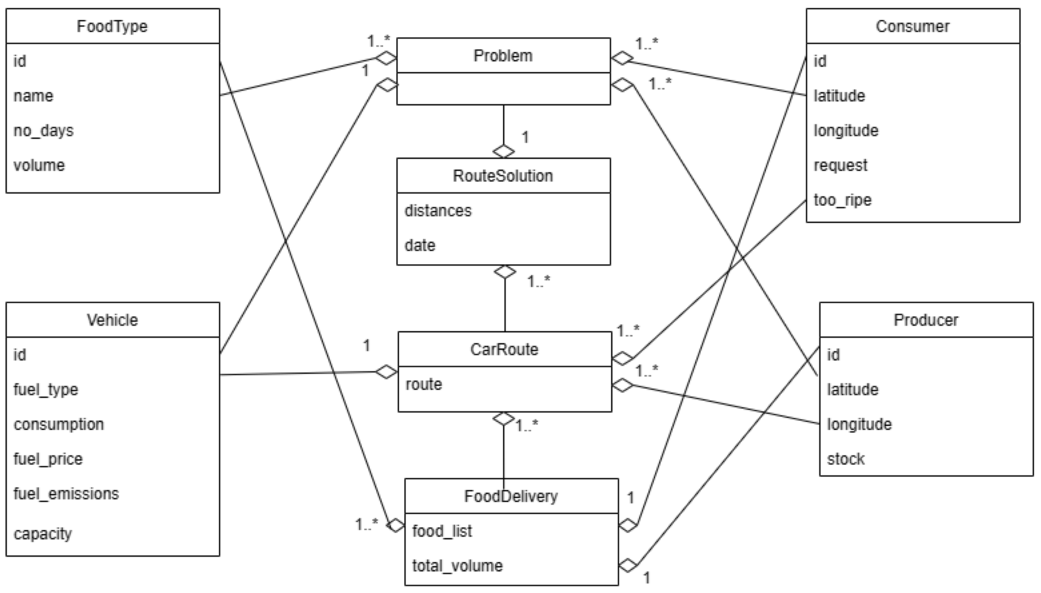


Figure 3.1: Diagram of the classes

the stock of the producers, the requests the consumers have, information about the cars and its emissions. The different variables needed for the problem reflect on the characterization of the classes of the problem. So, for consumers, one needed the location, if they could receive food that was too ripe, and the requests with the date on it. For producers, there was also a need for the location and its stock with the dates. For FoodTypes, besides the product's name, there was the number of days it took for the product to get spoiled and the volume of each unit occupied. Finally, for the vehicles, it was also essential to refer to the type of fuel, the fuel price, the CO2 emission rates, and the vehicle's capacity. As seen from the variables needed, a simple VRP dataset would not be enough. Hence, an extensive search for a dataset was carried out.

3.2.1 Search for Datasets

Two dataset websites were used to find possible datasets [155] and [156]. For dataset [155], the keywords "food delivery", "food delivery transportation", "transportation of perishable products", "green vehicle routing problem" and "vehicle emissions" were used. However, there were no datasets that could be used for the problem.

[156] is a website that contains only VRP datasets and the following keywords were used: "GVRP", "Green", "G-VRP", "food waste", "environment" and "perishable products". Seven datasets that looked promising were found. However, some of them were repeated, and the latitude and longitude were the only helpful information since the nature of the problems was different. Some datasets contained only electric vehicles not considered in this solution, others did not have different types of vehicles, and none had information regarding food waste.

Additionally, the article [157] contains the benchmarks datasets for VRP that was also consulted. Unfortunately, the datasets mentioned in this article suffered the same problems as dataset

[156]. Due to the difficulty in finding appropriate datasets that could fit the problem, the dataset used was generated.

3.2.2 Initial Dataset

The first dataset used to solve the problem was written by hand and contained four text files: "consumers.txt", "food_types.txt", "producers.txt," and "vehicles.txt". The locations used for the consumers and the producers were chosen in northern Portugal. The consumers' file has ten consumers, and each line represents a consumer. The producers' file has three producers. Each line represents a producer. Likewise, the vehicles file also has three vehicles. Finally, the food types file has ten fruit types and is represented by each fruit in one line.

Example of a consumer line:

```
0 41.193125 -8.660231 1 watermelon-1-22/05/2022,apple-3-23/05/2022
```

Example of a producer line:

```
0 41.242969 -8.648129 orange-20-2-22/05/2022,strawberry-40-4-22/05/2022
```

Example of a vehicle line:

```
0 diesel 15 1.507 2.6 500
```

Example of a food type line:

```
0 orange 10 0.1
```

3.2.3 Dataset Generation

Since the initial dataset was rather small and no dataset was found online, the dataset for testing was generated. First, 60 coordinates from the North part of Portugal were chosen for the consumer locations, and 18 coordinates from the North of Portugal were chosen for the producer ones. The locations were registered in a file for the consumer locations and another for the producer locations. Furthermore, more produce variety was added to the food types file, with 20 fruits and vegetables now. For both the producer and consumer generation, the number of days had to be indicated as the initial date. A function for request generation was written that would generate a list of requests in a determined day. The generator randomly selects the size of the request list between 1 and 5, and then a sample of food types would be randomly chosen with the size of the request list, so each request for the day had a different type of produce. The quantity was also randomly chosen between 1 and 10. The consumer generation function generated a request list for each consumer location and each day and randomly chose whether the consumer could receive extra food. Each line of the new consumer file generated corresponded to the consumer's information and the one-day request list. Similarly to the request generation function, a stock generation function was also developed. This function generates a stock list and randomly chooses a quantity between 30 and 100 for each food type. It also selects the number of days since ripped between the values of one and the number of days it takes for the food type not to be appropriate for consumption. The producer generator function generated a stock for each location and day. Finally, a vehicle information generator function was also created. Four fuel types were introduced, gasoline, diesel, hybrid, and LPG, as list strings. The usual interval of consumption values regarding each fuel type was also in a list, and the index of an element of this list was the same for the index of the type of fuel correspondent. A fuel emissions list with the values for each fuel type was also added. Since the fuel price is always changing, the function randomly generates four prices between the values of 1 to 2.5. Finally, the number of vehicles for the problem was chosen randomly between 12 and 20. For each vehicle, the type of fuel was randomly selected, and then consequently, the values of the price and the emissions were gathered, and the consumption was randomly chosen between the typical values of the fuel type. The vehicle's capacity was also generated between 300 and 1000. Each line of the new vehicle document corresponds to a vehicle.

The figure 3.2 shows the data flow of the files to the Problem class.

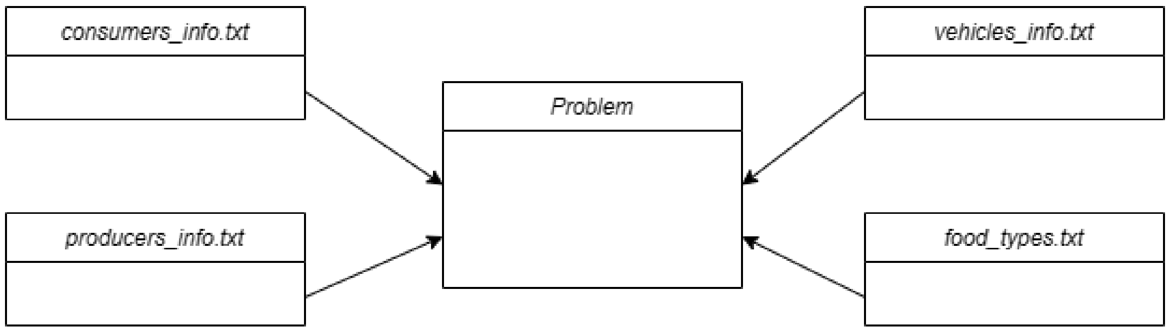


Figure 3.2: Diagram of the data flow

3.3 Generation of a Solution

As mentioned, the RouteSolution class represents a solution with the corresponding CarRoutes. The function that generates a solution requires an instance of the Problem class, with the information about the problem as a parameter, the distances dictionary to calculate the distances of the routes, the current date, and two boolean parameters, one that represents if the food should be optimally allocated concerning food waste and if the solution should allocate extra food. The respective sections will explain the cases where the food should be optimally allocated or if the solution should allocate extra food close to consumption date.

For the case where the food is not optimally allocated, the function will first allocate which producer will provide the food for each consumer request. It will call a function, allocate_food, demonstrated in listing 3.1.

```
function allocate_food(problem):
  for each consumer in consumers do:
    for each request in consumer request:
        producer <- choice(producers)
        if producer possible_to_supply request:
            food <- producer update_supply request
            add food to list fl

        add consumer id to list of producer id in visits dict

  return visits, fl
```

Listing 3.1: Allocate Food Function

As seen in listing 3.1, for each request, a random producer is chosen, and with possible_to_supply, it will check if the producer has the product that fulfills the request. If so, the update_supply function updates the producer's stock with the values of the request and returns a FoodDelivery class with the request as a food_list. The visits dictionary keeps the producers as keys and the list of the consumers that will receive produce from the producer as values. Afterward, the food delivery list is uniformed by reducing the objects with the same producer and consumer and adding the lists. Once this is concluded, the function will decide which car will care for which producer and respective consumers, like in listing 3.2.

```
  for each producer id in visits keys:
    car <- choice(vehicles)
    add producer id to list of card id in car_assignment dict
```

Listing 3.2: Initial Car Allocation

A random vehicle will be allocated to each producer with a car_assignment dictionary where the key is the car's id, and the value is a list of producers the car will transport its produce. The final step is creating the CarRoute objects in listing 3.3.

```
  for each vehicle in vehicles:
     sub_fl <- sub_food_list of list of producers from vehicle id in car_assignment
          dict from new_food_list
     route <-  create CarRoute with vehicle and sub_fl
     add route to list routes
  solution <- create RouteSolution with routes, problem, distances and date
```

Listing 3.3: Create CarRoute Objects

For each vehicle, the sub_food_list function is called and will create a new list of FoodDelivery objects that only have the producers of the car as producers, and a CarRoute object is created. Another section explains the generation of the routes mechanism. To conclude, figure 3.3 represents the flow of the function.

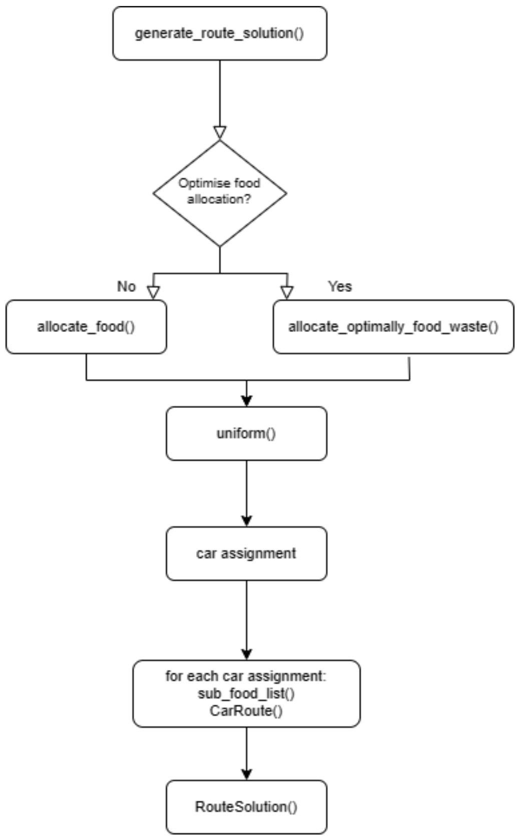


Figure 3.3: Diagram of the generate_route function flow

3.3.1 Fitness Function

The fitness function is calculated with three factors, the economic factor, the emissions factor, and the food waste factor. These parts are added to the fitness, and the goal is to minimize the fitness function. Therefore, the smaller the fitness result is, the better. The economic factor is the sum of the economic factor of each car route that's calculated like the following:

$$economicfactor = distance * vehicle.consumption * vehicle.fuel\_price \tag{3.1}$$

This way, it calculates the price of each route and adds the total. The emissions factor is also the sum of the emissions factor of each car route and is calculated like this:

$$emissionsfactor = distance * vehicle.consumption * vehicle.emissions \tag{3.2}$$

Finally, the food waste factor adds the food waste of each food in the food list in all the food deliveries of all car routes and is calculated like this:

$$foodwastefactor = food.no\_days - food.days\_since\_harvested \tag{3.3}$$

It calculates the difference between the days the product has until it goes rotten and the days the product has. This way, it prioritizes the produce closer to the consumption date and takes a step at preventing food waste. However, in case the product has been harvested longer than the days it takes to rot, the calculation goes like this:

$$foodwastefactor = food.no\_days + food.days\_since\_harvested \tag{3.4}$$

This technique penalizes the solution if the algorithm has, by chance, chosen produce that is too old for direct consumption. Both the emissions factor and the economic factor are divided by 100000000 and the food waste factor is divided by 1000 so each value has similar order of greatness and consequently similar importance. Furthermore, the value of the fitness is easier to read and interpret.

3.3.2 Distances Calculation

An API, called Openrouteservice [158], calculates the distances between two points, represented by their latitude and longitude. The calculation takes into account the distances of what the cars need to tavel. Other APIs were taken into consideration as well, such as Google Maps API [159], Bing Maps API [160] and Mapbox API [161], however the Openrouteservice was free and with a bigger limit of operations using the student version. The table 3.1 shows a quick comparison between the features of each API.

Table 3.1: APIs Comparison

Feature	OpenRouteService	GoogleMaps	BingMaps	Mapbox
Geocoding	✓	✓	✓	✓
Directions	✓	✓	✓	✓
Distance Matrix	✓	✓	✓	✓
Traffic Data	✓	✓	✓	✓
Open Source	✓	x	x	x

3.3.3 Optimisation of Food Allocation

When creating a RouteSolution object, there is the option of allocating the producers to the requests optimally, considering food waste, which means that the algorithm will choose the producer with the oldest produce that hasn't gone over the ideal consumption date.

```
for each request in consumers request:
    for each producer in producers:
        for each index i in list stock:
          if request name equals stock[i] name and request quantity  smaller or
              equal stock[i] quantity:
              if current_producer stock[current_index] days_since_harvested
                  smaller stock[i] days_since_harvested  and stock[i]
                  days_since_harvested smaller or equal stock[i]no_days:
                current_index <- i
                current_producer <- producer
```

Listing 3.4: Allocate Food Optimally

In listing 3.4, for each request, it will check the producers with the stock and the corresponding oldest produce without going over the consumption date. In this way, delivering the oldest produce instead of the more recent one will decrease the chances of food waste. Afterward, the FoodDelivery object is created, and the producer's stock is updated.

3.3.4 Route Representation

The route in a CarRoute object is a list of strings where, in the case of a producer, it is the string version of its id. However, in the case of a consumer, the string in question is in the format of "consumer_id - producers_id. For example, if the consumer has the id seven and, in that stop, the vehicle will deliver produce from producers 1 and 2, the element will be represented by "7-1,2". Here is an example of a simple route:

```
    ["0", "1","2", "4-1", "6-0", "8-2", "7-1,2", "9-0,2"]
```

3.3.5 Generation of Routes

When initialising a CarRoute object, the define_route function in listing 3.5 is called to create the route taking into account the producers and the consumers.

```
route <- generate_route()
probability <- random()
if probability smaller 0.5:
   optimize_route()
```

Listing 3.5: Define Route

The generate_route() function returns a list corresponding to the route of the CarRoute, and the optimize_route function will be explained in its section. The generate_route() in listing 3.6 is divided in two possible strategies: generate_prod_cons_route and generate_prods_first_route.

```
function generate_route():
    current_route <- [] #empty list
    probability <- random()
    if probability < 1/2:
      current_route <- generate_prod_cons_route()
    else:
      current_route <- generate_prods_first_route()
```

Listing 3.6: Generate Route

The generate_prod_cons_route strategy, for each producer of the CarRoute, puts first the producer and then the consumers after the list of consumers is shuffled. Additionaly, the generate_prods_first_route puts all the producers at the beginning of the list and then shuffles the consumers. The generate_complete_random_route randomly was initially used as well and it generates randomly the order of the route and it keeps generating until the order constraint is respected, however, when testing with the bigger dataset, this approach would take too long so it was removed.

3.3.6 Order Constraint

When generating or tweaking the route list, the proposed solution considers the order of the elements since the producer needs to go before the respective consumer so that the vehicle can pick up the product and then deliver it. Therefore, an order constraint is in place. The function in listing 3.7 implements this constraint and returns True if the order makes sense and False if it doesn't, therefore, in the case that at least one consumer is before the respective producer in the order of the stops.

```
    for each stop in route:
      if stop is_consumer:
        for producer in get_producers_list of stop:
          if producer not appears_after stop in route:
            return False
    return True
```

Listing 3.7: Order Constraint

For each route element, the function checks if it is a consumer. In that case, for each producer related to that element, in get_producers_list, it will check if the producer is before the consumer in the order of the route by calling the appears_after function. This function returns True if the consumer appears after the producer in the route order and False if otherwise.

3.3.7 Optimisation of Routes

When generating a route, there is a probability of optimising it. The idea of optimising the generated route came up since, in a CarRoute, the same consumer could receive products from different producers, and the routes generated two stops for the same consumer. Therefore, the optimize_route goes over the route, and if there are two stops on the same consumer, it deletes the first stop and adds the producer of the first stop in the second stop, reducing the number of stops of the vehicle and making it more efficient.

3.4 Solution Tweaks

The program includes three tweak functions to increase the diversity of the solutions, where each tackled a different part of the solution and slightly changed it. One of the functions slightly alters the food allocation of the solution. One slightly alters the order of the routes, and the other the car allocation.

3.4.1 Food Allocation Tweaks

The Food Allocation Tweak strategy changes at random the producer of a FoodDelivery object as shown in listing 3.8.

```
while not sanity_check:
   route1 <- choice(car_routes)
   route2 <- choice(car_routes)
   food_delivery <- choice(route1.food_delivery_list)
   new_producer <- choice(list(producers from route2))
   for each request in food_delivery.food_list:
     if new producer not possible_to_supply request:
       sanity_check = False
 consumer <- consumer from food_delivery
 give_back_stock to producer from food_delivery
 take_out_food_delivery food_delivery in route1
 add_food_delivery food_delivery in route2
```

Listing 3.8: Food Allocation

The function randomly plucks two CarRoutes from the car_routes list and chooses a FoodDelivery from the first CarRoute. Additionally, it chooses a new producer from the second CarRoute, and for each request in the food_list of the FoodDelivery instance, the function checks if the new_producer can fit that demand. If impossible, the cycle starts again by randomly selecting two CarRoutes. When it finds a fitting switch, it calls the give_back_stock function to update the stock of the first producer that is no longer catering to the requests and calls the take_out_food_delivery function to take out the FoodDelivery object from the CarRoute and makes the necessary changes. Afterward, the stock of the new_producer is updated, and the food_list from the FoodDelivery is also updated, considering the difference of days that passed since the producer harvested the produce. Finally, the object is updated, and the second CarRoute is also adjusted with the add_food_delivery function.

3.4.2 Route Order Tweaks

This function chooses a CarRoute from the solution and alters its route slightly. The listing 3.9 showcases part of the function.

```
  while not sanity_check:
    idx1 <- random(0, len(route)-1)
    idx2 <- random(0, len(route)-1)
    val1 <- route[idx1]
    val2 <- route[idx2]
    new_route <- route
    new_route[idx1] <- val2
    new_route[idx2] <- val1
    sanity_check <- call order_constraint on new_route
```

Listing 3.9: Route Order Tweak

It chooses two random indexes and switches the values. It repeats the process until the route fulfills the order constraint.

3.4.3 Car Allocation Tweaks

For the car allocation tweak, if the CarRoutes use all cars, the function randomly chooses two CarRoutes and switches their vehicles. However, if the solution is not using cars, it chooses a random CarRoute and switches the vehicle with an unused vehicle. One can have a better overview in listing 3.10

```
cars_unused <-  get_cars_unused()
if cars_unused is empty:
  switch_cars <- sample 2 cars from car_routes
  vehicle1 <- switch_cars[0].vehicle
  vehicle2 <- switch_cars[1].vehicle
  switch_cars[0].vehicle <- vehicle2
  switch_cars[1].vehicle <- vehicle1

else:
  rand_car <- choice(cars_unused)
  rand_car_route <- choice(car_routes)
  rand_car_route.vehicle <- rand_car
```

Listing 3.10: Car Allocation Tweak

3.5 Volume Constraint

The volume constraint comes from the finite space in the vehicles transporting the food. Therefore, there is a function in listing 3.11 that checks if the vehicle is transporting excess volume at any given time in the route.

```
current_volume <- 0
for each index i in list route:
  if route[i] is_consumer:
    subtract from current_volume volume_consumer_stop from route[i]
  else:
    add to current_volume volume_producer_stop of route[i]
  if current_volume bigger vehicle.capacity:
    return False
return True
```

Listing 3.11: Volume Constraint

The curr_volume variable keeps count of the volume the vehicle is currently transporting. It iterates over the route, and if the element is a consumer, it takes out the total volume of the delivery scheduled for that spot. If the element is a producer, it calls the get_volume_producer_stop function that retrieves the total volume that the vehicle will receive at that stop. This function considers multiple stops at the same producer, so it only calculates the total volume between the current producer stop and the next one if existent. In each element of the route, it checks if the curr_volume goes over the vehicle capacity. The solution breaks the constraint, and the function returns False. If the solution respects the constraint at the end of the iteration, it returns True.

In case the route does not respect the volume constraint, the rearrange_stops_capacity function, representation in listing 3.12, changes the route so that the volume constraint is respected.

```
producer_with_bigger_capacity <- get_prod_w_bigger_capacity()
if producer_with_bigger_capacity is empty:
  route <- generate_prod_cons_route()
else:
  for each producer in producer_with_bigger_capacity:
    take_producer_from_route producer
    sub_route <- get_prod_subroute of producer
    route <- route add sub_route
```

Listing 3.12: Rearrange Stops Capacity

First, the function will use the function get_prod_w_bigger_capacity to list the producers whose total volume of products is more significant than the vehicle's capacity. If no producer has that amount of food to sell, the route will just be recalculated to have first the producer and then the consumers, as explained in generate_prod_cons_route. If any producer needs to deliver more volume than the vehicle's capacity, the take_producer_from_route function is called and takes out the producer and respective consumers from the route. With get_prod_subroute, it will calculate a subroute just for the producer and its consumers and add a new producer stop when necessary. Finally, it adds the calculated subroute at the end of the previous route. This way, the route will not break any capacity constraints.

3.6 Allocation of Food Close to Consumption Date

As previously mentioned, there is the option of allocating the food close to the end of consumption date so that it is not wasted. In that case, the generate_route function will call the allocate_extra_food function, represented in listing 3.13.

```
extra_food <- get_food_too_ripe()
for each producer in extra_food keys:
  sorted_distances <- get_closer_consumers of producer
  close_consumers[producers] <- [first 2 elements of sorted_distances]
for each producer, food_list in extra_food:
  for each food in food_list:
    consumer <- choice(close_consumers[producer])
    producer_consumer_extra[(producer, consumer)] add food
```

Listing 3.13: Allocate Extra Food

In this function, the get_food_too_ripe function is called and returns a dictionary where the keys are the producers and the values are a list of tuples corresponding to the stock of the producer with the product that is one day from the limit of the consumption date, on the limit of the consumption date or older than the consumption date. This function also takes these products from the producer stock. Afterward, for each producer, the get_closer_consumers function will return a list of the consumers eligible to receive food close to the consumption date sorted by the closest consumers to the producer. The close_consumers dictionary then has the producer's key and the values as a list of the two closest consumers that can receive extra food. Then, each extra food of the producer will randomly allocate to either of the closer consumers. Finally, it will create the FoodDelivery objects corresponding to the producer and consumers and return a list of them.

```
extra_food <- allocate_extra_food()
for each food in extra_food:
  if food.producer not in extra_producers:
    add food.producer to extra_producers list
for each route in car_routes:
  for each producer in extra_producers:
    if producer in vehicle list from car_assignment dict:
      extra_food_producer <- list of food in extra_food with producer
      add_extra_food with producer and extra_food_producer to route
```

Listing 3.14: Extra Food Allocation

After getting the FoodDelivery list, the function gets a list of producers delivering the extra food. Then for each CarRoute of the solution, if the producer is allocated to that CarRoute, a sublist of the extra_food list is created only with the FoodDelivery with the producer as producer. The add_extra_food function is called, where it adds the FoodDelivery objects to the CarRoute, and if the consumers are already not visited by the vehicle, they are added to the route at the end of it. This logic can be seen in listing 3.14

3.7 Route Planning for Several Days

There is also a possibility to plan routes for several days. The get_next_x_routes function creates a list of RouteSolution elements where each corresponds to the solution proposed for each day. It is represented in listing 3.15.

```
function get_next_x_routes(problem, starting_day, number_of_days):
  for each day i in range(number_of_days):
    date <- starting_day + days*i
    sub_problem <- get_sub_problem of problem until date
    solution <- generate_route_solution of sub_problem
    add solution to routes list
    update_problem problem with sub_problem
    clean_up_update problem
```

Listing 3.15: Get Next X Routes

In the parameter section, the problem is the real problem with all the information from the files, the starting_day is the first day of planning, and x is the number of days the function will calculate routes. For each day, it calculates the current date, and then it gets a subproblem of the real problem where it excludes requests and stock with dates later than the current date. Afterward, it calculates the solution for the day, using the generate_route_solution, and appends it to the routes list. It then calls the update_problem function, which updates the total problem stock after allocating the food to the consumers. Finally, it calls the clean_up_update function, cleaning the empty stock and adding one day since harvested to the left produce.

3.7.1 Fitness

The fitness of a solution of several days is similarly calculated to the fitness of just one day, in listing 3.16

```
  for index i in range(current_solution):
```

```
  add to fitness the fitnes of current_solution[i] multiplied by 1/(i+1)
```

Listing 3.16: Route Planning for Several Days Fitness

In this case, for each day, it calculates the solution's fitness, divides the value by the day, and adds to the total sum. This way, the days closer to the first day have a more significant impact on the solution than the ones that are further away, prioritizing the first days over the later ones.

3.7.2 Solution Tweaks

To tweak a solution within several days, one randomly chooses an element of the solution and then uses with equal probability either the tweak_car_allocation function or the tweak_stops_order. The tweak_food_allocation function is not used since tweaking the food allocation in one day could interfere with the food allocation of the following days. Because of this, when creating RouteSolution solutions for each day, the option to allocate food optimally, considering food waste, is used.

3.8 Algorithms

Several metaheuristics algorithms were implemented to reach better solutions for the problem.

3.8.1 Neighbourhood Generation

The Hill Climbing, Simulated Annealing and Tabu Search algorithms used the same neighbourhood generation method, represented in listing 3.17.

```
function neighbourhood_generation(n_neighbours, solution):
  for _ in range(n_neighbours):
    new_neighbour <- solution
    probability <- random()
    if probability smaller 1/3:
      tweak_food_allocation(new_neighbour)
    elif probability smaller 2/3:
      tweak_stops_order(new_neighbour)
    else:
      tweak_car_allocation(new_neighbour)
    add neighbourhood list new_neighbour
  return neighbourhood
```

Listing 3.17: Neighbourhood Generation

With an equal probability of each tweak function, the neighbourhood was generated by calling tweak_food_allocation, tweak_stops_order and tweak_car_allocation.

3.8.2 Hill Climbing

The Hill Climbing algorithm requires the number of maximum iterations, an initial solution state, and the number of neighbours to create each time. A new neighbourhood is generated from the current_state for each iteration, and the smallest fitness checks the best_neighbour. If the best_neighbour fitness is more significant than the current_state, the current_state is returned. If not, the current_state becomes the best_neighbour, and the algorithm continues. One can check the listing 3.18 for a clearer representation.

```
function hill_climbing(max_iterations, initial_state, n_neighbours):
  current_state <- initial_state
  for _ in range(max_iterations):
    neighbourhood <- neighbourhood_generation of n_neighbours with current_state
    best_neighbour <- minimum fitness of neighbourhood
    if fitness of best_neighbour bigger or equal fitness of current_state:
      return current_state
    current_state <- best_neighbour
  return current_state
```

Listing 3.18: Hill Climbing

3.8.3 Simulated Annealing

The simulated annealing algorithm needs the initial state as a solution, the temperature, the cooling rate, the minimum value for the temperature, and the maximum number of iterations. The acceptance_probability function, in listing 3.19, determines the probability of accepting the new solution. If the new fitness is smaller than the current fitness, the probability is 100%. However, if not, the probability is calculated according to the temperature and the difference between the two fitness results.

```
function acceptance_probability(fitness, new_fitness, temperature):
    if new_fitness smaller fitness:
        return 1
    else:
        return exp((fitness minus new_fitness) divided by temperature)
```

Listing 3.19: Acceptance Probability

For each iteration, a new solution is generated with the neighbourhood_generation function from the current_state, and the acceptance_probability is calculated. If the new solution is accepted, the current_state becomes the new_solution, and if the current_cost is smaller than the best_cost, the best_state becomes the current_state. The temperature also decreases with the cooling rate. The algorithm continues until the temperature reaches the minimum temperature or the number of iterations reaches the maximum value. By then, the algorithm returns the best_state and the best_cost, corresponding to the best solution and best fitness, like in listing 3.20.

```
function simulated_annealing(initial_state, temperature, cooling_rate,
    min_temperature, max_iterations):
  while temperature bigger min_temperature and i smaller max_iterations:
    new_state <- neighbour from neighbourhood_generation of size 1
    new_cost <- fitness of new_state
    ap <- acceptance_probability(current_cost, new_cost, temperature)
    if ap bigger random():
      current_state <- new_state
      current_cost <- new_cost
      if current_cost smaller best_cost:
          best_state <- current_state
          best_cost <- current_cost
    temperature equals temperature multiplied by cooling_rate
    add 1 to i
  return best_state, best_cost
```

Listing 3.20: Simulated Annealing

3.8.4 Tabu Search

The Tabu Search algorithm, represented in listing 3.21 requires an initial solution, the tabu list size, maximum iterations, and the number of neighbours as parameters. For each iteration, the algorithm created a neighbourhood with the number of neighbours specified and the current_solution, which is at first the initial_solution. Afterward, it iterates over the neighbourhood, and when a solution where the fitness is smaller than the current_fitness and the solution is not in the tabu list, the current_solution becomes this solution. After going through the neighbourhood, it compares the current_fitness to the best_fitness, and if the value of the current_fitness is smaller than the best_fitness, it updates the best_solution. It adds the current_solution to the tabu_list, and if the tabu_list goes over the maximum size, it removes the oldest element. After reaching the maximum iterations, it returns the best solution.

```
function tabu_search(initial_solution, tabu_list_size, max_iterations, n_neighbours
    ):
    while iteration smaller max_iterations:
      neighbourhood <- neighbourhood_generation of size n_neighbours from
          current_solution
      for each neighbour in neighbourhood:
        if neighbour fitness smaller current_fitness and neighbour not in tabu_list
            :
          current_solution <- neighbour
          current_fitness <- neighbour fitness
      if current_fitness smaller best_fitness:
        best_solution <- current_solution
        best_fitness <- current_fitness
      add current_solution to tabu_list
      if length of tabu_list bigger tabu_list_size:
        take first element of tabu_list out
      add 1 to iteration
    return best_solution
```

Listing 3.21: Tabu Search

3.8.5 Genetic Algorithm

The genetic algorithm, in listing 3.22 starts by generating a population with the size stipulated and getting the current solution as the best solution from this initial population. Afterward, it generates the offspring for each generation and creates a new population, sorted by the lowest fitness value. If the best solution from the new population is better than the current solution, the variable is updated. After the number of generations defined is concluded, it returns the current_sol.

```
function genetic_algorithm(problem, distances, population_size, generations):
  population <- generate_population(population_size, problem, distances)
  sort population list according to fitness
  current_sol <- first element of population
  current_fitness <- fitness current_sol
  for _ in range(generations):
    offspring <- generate_offspring from population
    population <- generate_new_population from population and offspring
    fit <- fitness of first element of population
    if fit snaller current_fitness:
      current_sol <- first element of population
      current_fitness <- fit
  return current_sol
```

Listing 3.22: Genetic Algorithm

3.8.5.1 Population Generation

The population generation is taken care of by the function generate_population, which creates different instances of solutions to the problem. These solutions are created from zero, with the generate_solution function, to increase the diversity of the population.

3.8.5.2 Crossover

The crossover function takes two possible solutions and generates two children. It calls two strategies, the stops_order_crossover, which mixes the routes between parents, and the cars_allocation_crossover, which mixes the cars instead. It is showcased in listing 3.23

```
equal_dicts <- 0
for each car_route1 in solution1.car_routes:
  for each car_route2 in solution2.car_routes:
    if c1.prod_cons equals c2.prod_cons:
      add 1 to equal_dicts
if equal_dicts equals length of solution1.car_routes and length solution1.
    car_routes equals length solution2.car_routes:
  new_solution1, new_solution2 <- stops_order_crossover between solution1,
      solution2
else:
  new_solution1,new_solution2 <-cars_allocation_crossover between solution1
      solution2
```

Listing 3.23: Crossover

The stops_order_crossover, in listing 3.24, needs the solutions to have the same allocation of producers and consumers, so it makes sense to cross between the routes. Therefore, the function compares the dictionaries prod_cons of both solutions, representing the allocation of producers and consumers. In case they are equal, the function calls the stops_order_crossover. If not, the function calls the cars_allocation_crossover. The order crossover of two routes with the same producers and consumers works like the following.

```
idx1 <- random(1, lim-2)
idx2 <- random(1, lim-2)
auxlist1 <- route2[idx2:] add route2[:idx1] add route2[idx1:idx2]
auxlist2 <- route1[idx2:] add route1[:idx1] add route1[idx1:idx2]
auxlist1 <- [element for element in auxlist1 if element not in route1[idx1:idx2]]
auxlist2 <- [element for element in auxlist2 if element not in route2[idx1:idx2]]
child1 <- auxlist1[-idx1:] add route1[idx1:idx2] add auxlist1[:-idx1]
child2 <- auxlist2[-idx1:] add route2[idx1:idx2] add auxlist2[:-idx1]
```

Listing 3.24: Stops Order Crossover

The function takes two indexes and uses them as an interval. The first child gets part of the elements of route two at first, and then the interval of elements between the indexes of route one and then the rest of the elements of route two. This method guarantees that there are no repeated elements in the route. The second child gets part of the elements of route one at first and in the end, and the interval from route two. If the route children created do not respect the order constraint, the correct_order function is called. Listing 3.25 represents it.

```
producers <- list of elements from route if "-" not in element
while not order_constraint:
 route <- list of elements from route if element not in producers list
 for each producer in producers:
    rand_idx <- random(0, length route)
```

```
  insert in route producer in index rand_idx
  order_constraint <- check order_constraint_sanity_check in route
  if not order_constraint:
    add 1 to tries
  if tries equals 20:
    route <- producers add route
    order_constraint <- True
```

Listing 3.25: Correct Order

The function tries to place the producers in a different order in the route for 20 tries. If by 20 tries, no solution fulfills the order constraint, the route will become the producers appearing first and then the consumers. For the car_allocation_crossover, in listing 3.26, there does not need to be any similarity between the solutions and the solutions across the cars they use for the CarRoutes.

```
idx <- random(0, lim-2)
for index i in range(0,idx+1):
  add cars_1[i] to new_cars_1 list
  add cars_2[i] to new_cars_2 list
for index in range(idx+1, length cars_1):
  add cars_1[i] to new_cars_2 list
for index in range(idx+1, length cars_2):
  add cars_2[i] to new_cars_1 list
if repeated values in new_cars_1:
  new_cars_1 <- correct_car_crossover of new_cars_1
if repeated values in new_cars_2:
  new_cars_2 <- correct_car_crossover of new_cars_2
```

Listing 3.26: Car Allocation Crossover

The function takes one index from the list of vehicles used, and for the first child, the list of vehicles used will be the first elements of the first solution and then the last elements of the second solution according to the random index. For the second child, it is the opposite. If any of the new solutions have repeated elements, the function calls the correct_car_crossover that, for each repeated element, switches it with a car not being used by the solution.

3.8.5.3 Mutation

The mutation of solutions is done through the function mutate and is the same as the neighbourhood_generation function explained previously.

3.8.5.4 Tournament

The tournament function takes a list of solutions with the tournament size and returns the solution with the smallest fitness.

3.8.5.5 Offspring and Neighbourhood Generation

To generate offspring, the generate_offspring function, represented in listing 3.27, is called.

```
function generate_offspring(current_neighbourhood, tournament_size):
  for _ in range(population_size/2):
    tournament_list <- sample(current_neighbourhood, tournament_size)
    parent1 <- tournament(tournament_list)
    parent2 <- choice(current_neighbourhood)
    child1, child2 <- crossover(parent1, parent2)
    if should_mutate():
      mutation(child1)
    if should_mutate():
      mutation(child2)
    add child1 to offspring list
    add child2 to offspring list
  return offspring
```

Listing 3.27: Generate Offspring

The offspring will generate the same number of children as the population size. For every two children, it will choose two parents, one from the tournament and another one randomly, both from the current neighborhood, and use a crossover between them to generate the two children. Then, there is a probability for each child to mutate. This function then returns the list of the offspring. The current population and the offspring are required to generate a new population. This function, represented in listing 3.28, sorts the current_population, and the offspring as the smallest fitness being the first and the best half of each list is added to the new population. Thus, using the steady-state strategy, the next population has the best from both the previous population and the offspring.

```
function generate_new_population(current_population, offspring, population_size):
  sort current_population by fitness
  sort offspring by fitness
  for index i in range(population_size/2):
    add current_population[i] to new_population
    add offspring[i] to new_population
  sort new_population by fitness
  return new_population
```

Listing 3.28: Generate New Population

Chapter 4

Experiments and Results

This chapter goes over the experiments performed to validate the proposed solution, and reveals the results obtained.

4.1 Experiments Explanation

This section highlights the experiments carried out and how they were done. A personal computer ran the experiments of the algorithms and registered the fitness results and the time it took to solve them in seconds. The standard deviation of the fitness results was later calculated and added. Each experiment was repeated three times. The computational power was severely limited because a single personal computer was used for the experiments.

The experiences focused on the developed algorithms and their parameters. For the Hill Climbing algorithm, the maximum number of iterations and the neighborhood size were explored. The Simulated Annealing algorithm experiments explored values for the temperature, the cooling rate, the minimum temperature, and the maximum number of iterations. The Tabu Search algorithm experiments tweaked the values of the tabu list size, the maximum number of iterations, and the neighborhood size. Finally, the Genetic algorithm experiences focused on changing the population size, the tournament size, the mutation probability, and the number of generations. Furthermore, both individual solutions of one day and several days (four days) were tested. Different variations of neighbourhood generations were tested. Variant 1 corresponds to the Food Allocation and Stops Order tweak, the variant 2 includes the Car Allocation and Stops Order tweak, and, finally, the variant 3 has the Food Allocation and Car Allocation tweak. The option for optimal food allocation and extra food allocation close to the consumption date were also tested.

Hill Climbing

This section presents the results for the Hill Climbing algorithm. The values experimented for this algorithm are in table 4.1, for the baseline algorithm, the food allocation optimisation option and extra food allocation option close to consumption date. The values in table 4.2 reference the results using Variants of the Tweak Function, Variant1, Variant2 and Variant3. Finally, the values in table 4.3 are the ones for the route planning for several days.

Table 4.1: Values for Hill Climbing

It	No N	Fit	S	T	FitO	SO	TO	FitE	SE	TE
20	10	12.56	0.87	3.50	16.47	0.18	3.40	16.49	0.39	4.02
20	20	11.92	0.88	8.80	15.78	0.43	9.08	15.12	1.28	9.97
20	50	11.9	0.33	24.74	14.88	0.89	23.71	15.11	0.57	26.47
50	10	12.54	1.07	5.77	15.56	0.21	6.30	16.50	0.49	4.43
50	20	10.77	0.44	23.5	14.24	0.29	22.94	15.00	0.75	12.91
50	50	11.32	0.38	64.35	13.54	0.28	60.28	14.88	0.12	67.93
200	10	11.50	0.26	5.85	15.69	1.99	9.09	16.94	1.13	5.42
200	20	10.63	1.02	39.55	10.8	1.63	63.46	13.85	0.59	40.54
200	50	9.07	0.72	143.9	7.98	0.27	239.78	11.45	1.01	175.40
400	70	7.50	0.44	434.81	5.41	0.60	467.56	10.84	0.46	337.30

It - Maximum Iterations; No N - Number of Neighbours; Fit - Fitness (Baseline); S - Standard Deviation (Baseline); T - Time (Baseline); FitO - Fitness (Food Allocation Optimisation); SO - Standard Deviation (Food Allocation Optimisation); TO - Time(Food Allocation Optimisation); FitE - Fitness (Extra Food Allocation); SE - Standard Deviation (Extra Food Allocation); TE - Time (Extra Food Allocation)

From table 4.1, one can see that the fitness values improve over iterations and size of neighbourhood, as expected.

Table 4.2: Hill Climbing Variants

It	No N	Fit1	S1	T1	Fit2	S2	T2	Fit3	S3	T3
20	10	13.53	0.96	7.01	12.55	0.39	2.19	12.90	0.66	2.95
20	20	12.16	0.92	14.65	13.07	0.30	9.12	13.00	0.34	14.52
20	50	12.64	0.37	35.05	12.38	0.26	30.99	12.63	0.28	24.77
50	10	12.54	0.47	11.01	12.67	0.59	1.97	11.89	1.29	2.23
50	20	11.32	0.32	32.37	12.38	0.42	10.85	11.48	0.53	23.63
50	50	10.62	0.55	94.23	11.20	0.04	67.91	11.04	0.69	63.34
200	10	12.51	0.61	13.42	12.16	0.13	3.98	11.55	1.03	4.57
200	20	9.88	0.09	80.81	12.90	1.57	9.98	10.38	0.83	32.18
200	50	7.46	0.47	279.77	11.10	0.60	134.27	8.17	0.55	226.99
400	70	6.74	0.49	429.86	10.49	0.74	171.41	6.90	0.51	351.53

It - Maximum Iterations; No N - Number of Neighbours; Fit1 - Fitness (Variant1); S1 - Standard Deviation (Variant1); T1 - Time (Variant1); Fit2 - Fitness (Variant2); S2 - Standard Deviation (Variant2); T2 - Time(Variant2); Fit3 - Fitness (Variant3); S3 - Standard Deviation (Variant3); T3 - Time (Variant3)

Regarding table 4.2, one can see that the fitness values are lower in Variant 1 and 3, and Variant 2 takes the shortest amount of time.

For table 4.3, as expected, one can see that the fitness values are higher than compared to the baseline algorithm.

Table 4.3: Several Days Hill Climbing

It	No N	Fit	S	T
20	10	18.62	0.9	4.21
20	20	18.07	0.48	19.23
20	50	18.01	0.08	49.53
50	10	18.25	0.56	4.22
50	20	18.31	0.24	21.60
50	50	17.43	0.25	116.48
200	10	18.81	0.36	3.61
200	20	18.22	0.47	17.94
200	50	17.48	0.31	130.13
100	120	16.79	0.21	525.28

It - Maximum Iterations; No N - Number of Neighbours; Fit - Fitness; S - Standard Deviation; T - Time

4.2 Simulated Annealing

This section shows the results obtained from the Simulated Annealing algorithm. The first parameters tested for the algorithm are in table 4.4. However, the time it took to run was too short, so the parameters were adjusted in table 4.5. This table has the baseline algorithm, the food allocation optimisation option and the extra food allocation option close to consumption date. The values in table 4.6 reference the results using Variants of the Tweak Function, Variant1, Variant2 and Variant3 with Simulated Annealing. Table 4.7 references the results from the route planning for several days option.

Table 4.4: Small Simulated Annealing

It	Te	Cool	M T	Fit	S	T
500	100	0,99	0,5	11.23	0.37	11.00
500	100	0,99	1	11.27	0.61	12.13
500	500	0,99	0,5	11.38	0.97	13.21
500	500	0,99	1	11.85	0.35	14.17
1000	100	0,99	0,5	11.54	0.42	16.31
1000	100	0,99	1	11.74	0.42	10.83
1000	500	0,99	0,5	10.82	0.61	15.98
1000	500	0,99	1	10.88	0.10	14.69

It - Maximum Iterations; Te - Temperature; Cool - Cooling Rate; M T - Minimum Temperature; Fit - Fitness; S - Standard Deviation; T - Time

From table 4.4, one can check that the algorithm is quite fast compared to other algorithm tests, thus arising the need to tweak the parameters in order to run for longer.

Regarding table 4.5, it shows that the fitness values don't vary much considering different parameters.

Table 4.5: Values for Simulated Annealing

It	Te	Cool	M T	Fit	S	T	FitO	SO	TO	FitE	SE	TE
1e4	1e3	0,99	1e-14	5.29	0.25	76.64	4.25	0.09	72.60	17.03	0.10	79.88
1e4	1e3	0,99	1e-16	5.48	0.22	85.83	3.98	0.10	81.26	17.36	0.32	89.88
1e4	1e4	0,99	1e-14	5.56	0.18	82.14	3.84	0.03	76.96	17.41	0.41	85.63
1e4	1e4	0,99	1e-16	5.19	0.23	92.26	3.91	0.26	85.65	17.59	0.22	94.83
1e5	1e3	0,99	1e-14	5.66	0.15	77.53	4.21	0.31	73.37	17.49	0.22	80.63
1e5	1e3	0,99	1e-16	5.21	0.27	93.42	4.04	0.16	83.07	17.21	0.56	90.19
1e5	1e4	0,99	1e-14	5.56	0.08	93.64	4.29	0.29	77.43	17.44	0.46	97.24
1e5	1e4	0,99	1e-16	5.29	0.23	100.61	3.92	0.15	85.66	17.39	0.18	118.81
1e18	1e15	0,99	1e-40	-	-	-	-	-	-	16.53	0.10	365.35

It - Maximum Iterations; Te - Temperature; Cool - Cooling Rate; M T - Minimum Temperature; Fit - Fitness (Baseline); S - Standard Deviation (Baseline); T - Time (Baseline); FitO - Fitness (Food Allocation Optimisation); SO - Standard Deviation (Food Allocation Optimisation); TO - Time(Food Allocation Optimisation); FitE - Fitness (Extra Food Allocation); SE - Standard Deviation (Extra Food Allocation); TE - Time (Extra Food Allocation)

Table 4.6: Simulated Annealing Variants

It	T	Cool	M T	Fit1	S1	T1	Fit2	S2	T2	Fit3	S3	T3
1e4	1e3	0,99	1e-14	4.79	0.05	115.83	9.85	0.31	83.62	4.67	0.02	75.29
1e4	1e3	0,99	1e-16	4.25	0.18	124.43	10.42	0.28	98.51	4.35	0.45	83.69
1e4	1e4	0,99	1e-14	4.38	0.12	91.98	9.79	0.75	89.13	4.41	0.08	79.30
1e4	1e4	0,99	1e-16	4.09	0.14	102.61	9.11	0.57	99.79	4.27	0.08	87.78
1e5	1e3	0,99	1e-14	4.48	0.33	117.25	9.48	0.56	102.23	4.51	0.10	75.33
1e5	1e3	0,99	1e-16	4.21	0.11	124.65	8.58	0.65	129.46	4.42	0.27	83.85
1e5	1e4	0,99	1e-14	4.21	0.09	118.46	9.08	0.22	125.89	4.47	0.09	79.65
1e5	1e4	0,99	1e-16	3.98	0.22	93.11	9.03	1.24	126.67	4.29	0.04	87.86

It - Maximum Iterations; Te - Temperature; Cool - Cooling Rate; M T - Minimum Temperature; Fit1 - Fitness (Variation1); S1 - Standard Deviation (Variation1); T1 - Time (Variation1); Fit2 - Fitness (Variation2); S2 - Standard Deviation (Variation2); TO - Time(Variation2); FitE - Fitness (Variation3); SE - Standard Deviation (Variation3); TE - Time (Variation3)

Table 4.7 shows that Variant 1 and Variant 3 achieve similar fitness results, smaller ones compared to Variant 2. Variant 3 is the quickest and Variant 1, on average, takes the longest.

As expected, from table 4.7, the time taken to run several routes is longer than baseline.

4.3 Tabu Search

This section displays the results of the experiments using the Tabu Search Algorithm. The first values used as parameters are in table 4.8 with the baseline option, the food allocation optimisation option and the extra food allocation option close to consumption date. The table 4.9 presents the results using Variant 1, Variant 2 and Variant 3 for the Tweak Function. The results in table 4.10 are from the route planning for several days option.

Table 4.7: Several Days Simulated Annealing

It	Te	Cool	M T	Fit	S	T
1e4	1e3	0,99	1e-14	16.21	0.42	177.49
1e4	1e3	0,99	1e-16	16.02	0.16	209.20
1e4	1e4	0,99	1e-14	16.31	0.13	193.1
1e4	1e4	0,99	1e-16	16.34	0.15	216.73
1e5	1e3	0,99	1e-14	16.42	0.18	193.13
1e5	1e3	0,99	1e-16	16.13	0.33	239.38
1e5	1e4	0,99	1e-14	16.32	0.18	231.43
1e5	1e4	0,99	1e-16	16.03	0.24	215.9

It - Maximum Iterations; Te - Temperature; Cool - Cooling Rate; M T - Minimum Temperature; Fit - Fitness; S - Standard Deviation; T - Time

Table 4.8: Values for Tabu Search

It	No N	T L	Fit	S	T	FitO	SO	TO	FitE	SE	TE
20	20	5	13.06	0.28	9.48	15.93	0.37	8.67	16.26	1.00	19.14
20	20	15	11.84	0.60	9.60	15.55	0.17	11.24	15.64	1.18	19.78
20	40	5	11.23	0.85	19.73	15.84	0.10	24.52	15.35	0.47	40.64
20	40	15	12.61	0.53	19.86	15.34	0.71	18.21	15.54	0.64	40.66
50	20	5	10.87	0.60	23.93	14.43	0.40	21.84	14.96	0.22	50.57
50	20	15	10.98	0.91	24.25	14.40	0.55	22.16	14.92	0.33	51.99
50	40	5	10.67	0.67	50.24	13.91	0.60	46.77	14.86	0.68	105.36
50	40	15	10.45	1.45	50.54	14.12	0.22	46.37	14.33	0.36	105.23
200	20	5	8.03	0.29	95.75	8.55	0.24	87.81	11.42	0.52	201.44
200	20	15	8.03	0.43	98.23	9.39	0.46	89.85	11.90	0.07	205.98
200	40	5	7.33	0.60	204.43	7.93	0.48	183.52	11.28	0.30	424.64
200	40	15	7.91	0.51	208.47	8.34	0.50	206.18	10.92	0.20	409.90
325	40	5	5.91	0.10	468.78	5.50	0.33	290.32	9.56	0.35	359.8

It - Maximum Iterations; No N - Number of Neighbours; T L - Tabu List Size; Fit - Fitness (Baseline); S - Standard Deviation (Baseline); T - Time (Baseline); FitO - Fitness (Food Allocation Optimisation); SO - Standard Deviation (Food Allocation Optimisation); TO - Time(Food Allocation Optimisation); FitE - Fitness (Extra Food Allocation); SE - Standard Deviation (Extra Food Allocation); TE - Time (Extra Food Allocation)

For table 4.8, one can see that the fitness values also improve when increasing the number of iterations and the number of neighbours.

As previously noted, in table 4.9, the variants that achieve the best results are 1 and 3. The one that takes the shortest amount of time is 3 and the longest is 1.

Regarding table 4.10, one can see that, as expected, it takes longer to run a solution for several days than a single solution with the baseline approach.

Table 4.9: Tabu Search Variants

It	No N	T L	Fit1	StD1	Time1	Fit2	StD2	Time2	Fit3	StD3	Time3
20	20	5	12.57	0.78	13.73	12.36	1.22	14.57	11.89	0.42	13.83
20	20	15	11.53	1.86	16.94	12.06	0.66	14.41	12.47	0.31	15.20
20	40	5	12.00	1.34	31.22	12.34	0.67	30.15	11.99	0.72	23.48
20	40	15	12.30	1.31	33.74	12.58	0.39	30.45	12.00	0.75	19.88
50	20	5	11.76	0.93	39.59	11.41	0.73	38.09	10.58	1.27	23.83
50	20	15	11.82	1.57	39.23	12.41	0.35	37.93	11.59	0.20	24.57
50	40	5	11.07	1.31	78.66	11.35	0.48	68.65	10.92	0.73	50.78
50	40	15	11.46	1.82	76.31	10.44	0.52	51.83	10.94	0.52	50.74
200	20	5	8.57	0.47	160.07	9.99	0.99	98.15	7.91	0.13	95.40
200	20	15	7.87	0.32	151.98	9.63	0.38	99.47	8.03	0.32	98.06
200	40	5	7.39	0.21	316.59	10.25	0.11	208.90	6.93	0.29	201.09
200	40	15	7.27	0.20	295.78	8.74	0.87	259.28	7.21	0.14	289.92
325	40	5	5.59	0.14	416.95	8.76	0.83	376.83	5.78	0.15	322.44

It - Maximum Iterations; No N - Number of Neighbours; T L - Tabu List Size; Fit1 - Fitness (Variation1); S1 - Standard Deviation (Variation1); T1 - Time (Variation1); Fit2 - Fitness (Variation2); S2 - Standard Deviation (Variation2); T2 - Time(Variation2); Fit3 - Fitness (Variation3); S3 - Standard Deviation (Variation3); T3 - Time (Variation3)

Table 4.10: Several Days Tabu Search

It	No N	T L	Fit	S	T
20	20	5	17.97	0.41	26.59
20	20	15	18.11	0.20	29.06
20	40	5	17.87	0.36	52.77
20	40	15	17.93	0.68	52.52
50	20	5	17.55	0.17	65.98
50	20	15	17.65	0.15	66.26
50	40	5	17.32	0.31	117.51
50	40	15	17.35	0.38	112.37
200	20	5	16.65	0.40	235.65
200	20	15	16.53	0.13	241.20
200	40	5	16.01	0.34	472.58
200	40	15	16.26	0.05	512.03

It - Maximum Iterations; No N - Number of Neighbours; T L - Tabu List Size; Fit - Fitness; S - Standard Deviation; T - Time

4.4 Genetic Algorithm

This section reveals the results by running the Genetic Algorithm. The first values to experiment this algorithm are in table 4.11, along with the food allocation optimisation option and the extra food allocation close to consumption date option. The values in table 4.12 represent the results from the route planning for several days option.

Table 4.11: Values for Genetic Algorithm

Gen	P S	T S	M	Fit	S	T	FitO	SO	TO	FitE	SE	TE
20	20	5	0.2	11.43	0.51	14.31	15.90	0.07	12.77	15.00	0.52	14.49
20	20	5	0.5	11.64	0.41	15.84	15.55	0.21	15.76	15.52	0.40	19.30
20	20	10	0.2	11.31	0.17	13.14	15.65	0.44	12.83	14.76	0.22	17.69
20	20	10	0.5	11.49	0.57	16.16	15.70	0.10	15.15	16.00	0.56	21.30
20	50	5	0.2	11.00	0.14	38	15.15	0.20	32.22	14.83	0.36	41.81
20	50	5	0.5	11.2	0.35	51.07	15.14	0.30	37.76	14.85	0.64	49.74
20	50	10	0.2	11.33	0.46	42.27	15.23	0.44	33.22	14.47	0.70	42.18
20	50	10	0.5	10.65	0.45	45.43	15.28	0.06	39.25	14.58	0.23	59.05
50	20	5	0.2	11.59	0.35	34.25	15.51	0.40	28.38	14.68	0.66	44.54
50	20	5	0.5	11.61	0.78	40.37	15.95	0.39	34.76	15.18	0.51	58.12
50	20	10	0.2	11.68	0.56	33.41	15.73	0.56	29.69	14.80	1.29	48.73
50	20	10	0.5	11.73	0.48	39.22	15.54	0.04	35.15	15.39	0.20	61.05
50	50	5	0.2	10.97	0.32	84.98	15.71	0.35	77.24	14.23	0.39	118.13
50	50	5	0.5	11.23	0.10	110.16	15.23	0.34	91.22	14.57	0.17	128.68
50	50	10	0.2	11.21	0.34	82.68	15.26	0.30	76.88	14.62	0.29	108.81
50	50	10	0.5	10.65	0.39	101.02	15.41	0.11	113.02	14.32	0.52	130.99
200	20	5	0.2	11.48	0.33	142.82	15.73	0.50	110.61	15.13	0.26	161.41
200	20	5	0.5	11.15	0.71	191.03	15.58	0.31	155.36	15.11	0.66	183.98
200	20	10	0.2	11.33	1.07	145.68	15.70	0.56	131.6	14.53	0.54	154.90
200	20	10	0.5	11.70	0.44	156.46	15.51	0.49	137.98	15.53	0.07	187.67
200	50	5	0.2	11.10	0.30	378.32	15.24	0.10	338.71	14.57	0.65	397.11
200	50	5	0.5	11.01	0.63	457.28	15.26	0.56	405.05	14.73	0.23	472.77
200	50	10	0.2	11.23	0.20	390.72	15.57	0.36	359.63	14.81	0.24	410.16
200	50	10	0.5	11.16	0.33	470.86	15.27	0.17	404.40	14.42	0.20	517.53

Gen - Generations; P S - Population Size; T S - Tournament Size; M - Mutation Rate; Fit - Fitness (Baseline); S - Standard Deviation (Baseline); T - Time (Baseline); FitO - Fitness (Food Allocation Optimisation); SO - Standard Deviation (Food Allocation Optimisation); TO - Time(Food Allocation Optimisation); FitE - Fitness (Extra Food Allocation); SE - Standard Deviation (Extra Food Allocation); TE - Time (Extra Food Allocation)

From table 4.11, one can see that the fitness values don't differentiate much from the different set of parameters, unlike the time taken to run the algorithm.

In table 4.12, comparing the same set of parameters between this approach and the baseline, one can see that calculating the route for several days takes way more time than the baseline approach.

Table 4.12: Several Days Genetic Algorithm

Gen	P S	T S	M	Fit	S	T
20	20	5	0.2	18.00	0.19	49.57
20	20	5	0.5	17.91	0.20	56.15
20	20	10	0.2	17.88	0.21	44.32
20	20	10	0.5	18.07	0.25	63.44
20	50	5	0.2	17.65	0.12	137.07
20	50	5	0.5	17.62	0.27	168.92
20	50	10	0.2	17.30	0.50	145.14
20	50	10	0.5	17.60	0.26	153.33
50	20	5	0.2	17.89	0.11	120.02
50	20	5	0.5	17.84	0.24	141.04
50	20	10	0.2	17.85	0.13	118.31
50	20	10	0.5	18.00	0.31	143.33
50	50	5	0.2	17.84	0.25	303.97
50	50	5	0.5	17.49	0.03	392.28
50	50	10	0.2	17.62	0.24	341.32
50	50	10	0.5	17.18	0.31	407.26
60	60	10	0.2	17.40	0.16	518.91

Gen - Generations; P S - Population Size; T S - Tournament Size; M - Mutation Rate; Fit - Fitness; S - Standard Deviation; T - Time

Chapter 5

Discussion

This chapter analyses the results of the algorithms and discusses the reasons for said results. Furthermore, it explores the solution diversity in a critical light and its efficiency as well. Finally, it goes over the challenges faced when developing the proposed solution.

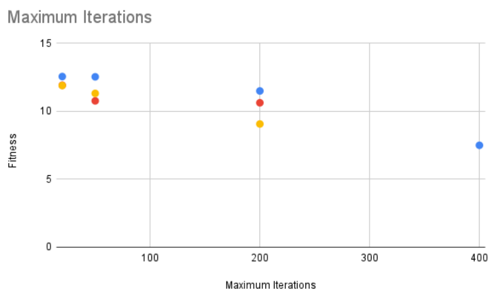
5.1 Results Comparison

This section analyses the results presented in the last chapter. It concludes the most impactful parameters, the quickest algorithm, which algorithm found the best results, and which tweak functions improved the results most by comparing them.

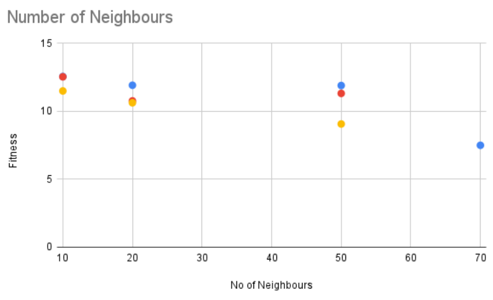
5.1.1 Hill Climbing

For the Hill Climbing algorithm, the parameter that primarily influences the time it takes to run is the Number of Neighbours when this number is low. For example, whether the maximum number of iterations is 20, 50, or 200, it takes less than 6 seconds to run the algorithm with ten neighbours per iteration. However, increasing the neighborhood size, generating more possible solutions, and increasing the selective pressure make it easier to find a better solution than the current one. This way, the algorithm runs through more iterations, and the number of maximum iterations parameter becomes more relevant. For example, with 20 iterations and No of Neighbors as 10, the time is 3.50; with 200 iterations, it is 5.85, a slight increase. However, comparing the time it takes with 50 neighbours and 20 iterations, 24.74, and 50 neighbours and 200 iterations, 143.9, the difference is way more prominent.

Regarding fitness, both parameters are essential to achieve a better result. When the neighborhood's size increases, the competition between neighbours increases, and the likelihood of finding a better solution increases. When the algorithm runs for more iterations, more possible solutions are generated, enabling better solutions. This way, when increasing both parameters, the fitness value decreases accordingly. One can look at figure 5.1 to visualize this effect. The standard deviation is generally smaller when the number of neighbours is more considerable. This might be explained by the bigger selective pressure, which can lead to a more systematic search and a more stable convergence pattern, reducing the variability and resulting in a smaller standard deviation.



(a) Maximum Iterations



(b) Number of Neighbours

Figure 5.1: Hill Climbing Parameters and Fitness

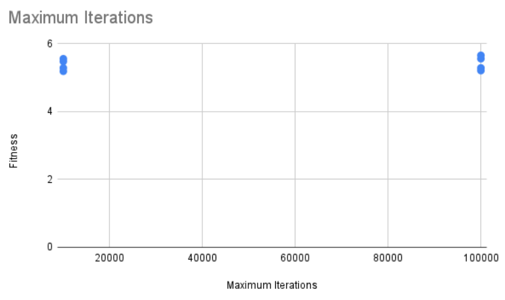
5.1.2 Simulated Annealing

The first tests for Simulated Annealing used relatively small parameters, and since Simulated Annealing does not create a neighbourhood in each iteration, the time it took to run this set of parameters was considerably shorter compared to the other algorithms. Given this initial feedback, the values for the parameters were changed so that the algorithm could run for more extended periods. To achieve this, the maximum number of iterations was increased, such as the temperature, and the minimum value for the temperature was decreased. Looking at the values, one can then confirm that the time it takes to run the algorithm increases for more significant maximum iteration values, bigger temperatures, and smaller minimum temperatures. However, it varies little compared to other algorithms due to the choice of parameters.

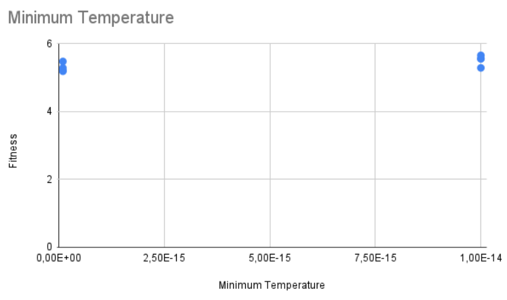
The fitness results are similar for the second set of parameters. The fitness has converged since it has been running for quite some iterations, and the results are reasonable compared to other algorithms. To see a more significant difference in fitness and time, choosing parameters further away from each other would be beneficial. Either way, having a more considerable value for maximum iterations and temperature and a smaller value for minimum temperature allows the algorithm to run for longer, creating more solutions and increasing the chances of finding a better solution. Figure 5.2 visually represents how parameters influence fitness. For the standard deviation, the results are overall similar and generally minor. This might also be a consequence of the algorithm convergence.

5.1.3 Tabu Search

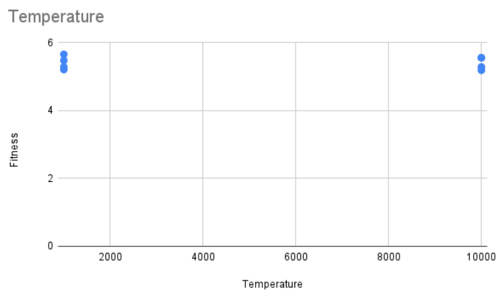
Regarding the Tabu Search algorithm, one parameter that revealed little significance was the Tabu List Size, having similar Fitness and Time values. For future work, other values for Tabu List Size should be tested. Regarding the maximum number of iterations and the number of neighbours, given the algorithm's nature, the neighborhood's size is less relevant than in comparison with Hill



(a) Maximum Iterations



(b) Minimum Temperature



(c) Temperature

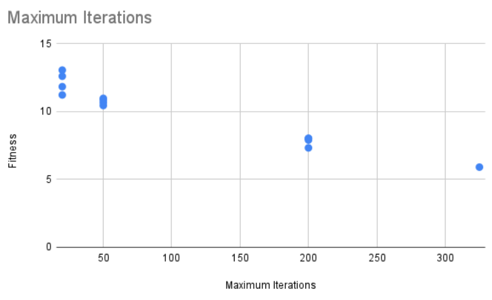
Figure 5.2: Simulated Annealing Parameters and Fitness

Climbing since it runs through all iterations independently of finding a better solution. Thus, increasing the maximum number of iterations and the size of the neighbourhood will improve the fitness results, with the bigger neighborhood size increasing selective pressure and the number of iterations generating more solutions. Similarly, increasing these parameters will take longer to run the algorithm. When doubling the number of neighbours from 20 to 40, the time is also approximately doubled. Analysing the values of maximum iterations, the values for the time are also approximately proportional. One can consult figure 5.3 to visualize this.

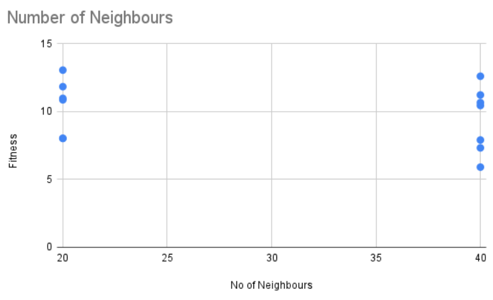
Considering the standard deviation, it is generally more prominent when the tabu list size is bigger. This might be explained by the fact that when the list size is bigger, the diversification of solutions is encouraged by avoiding previously visited solutions. This effect may introduce more randomness or stochasticity in the search process, and the behavior becomes less deterministic, leading to a broader range of solutions explored and a more significant standard deviation.

5.1.4 Genetic Algorithm

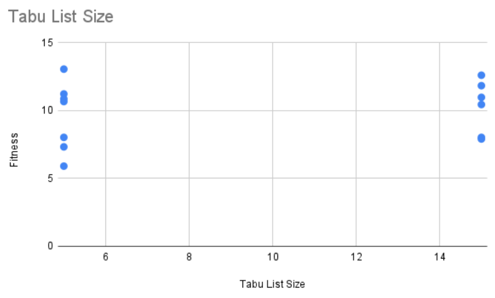
When analysing the Genetic Algorithm table, four parameters were tested: the number of generations, the population size, the tournament size, and mutation probability. These parameters strongly influenced the time to run the algorithm but not the fitness. When increasing the mutation probability by tweaking more solutions, the time it takes to run the algorithm is slightly longer. By increasing the mutation rate, one should also expect the diversity of the solutions to increase.



(a) Maximum Iterations



(b) Number of Neighbours



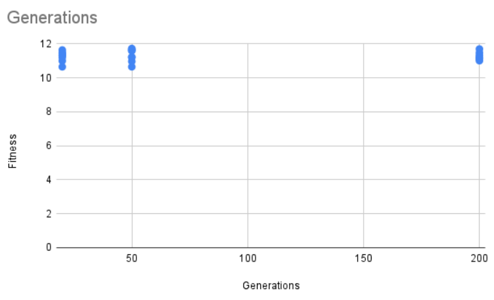
(c) Tabu List Size

Figure 5.3: Tabu Search Parameters and Fitness

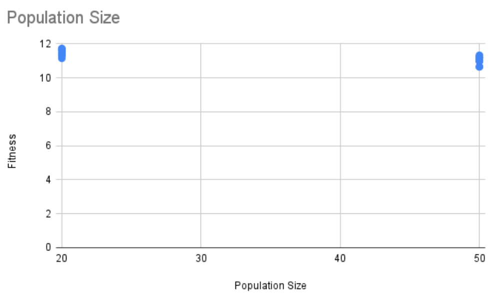
Regarding the tournament size, by increasing this parameter, more solutions compete to be the parent of offspring. Thus, more time was taken to run the algorithm, and more selective pressure was expected. By increasing the population size, more solutions are created, technically increasing the diversity of the population. Furthermore, the number of offspring created increases for each generation and the time to run the algorithm. Finally, by running the algorithm through more generations, more solutions are created, and more time is necessary to run the algorithm. Even after taking into account these considerations, the parameter tweaking did not translate into significantly better results for fitness. Figure 5.4 is a visual representation of how parameters influence fitness.

Regarding the standard deviation, this value decreases when the population size increases. This might happen because when the population size increases, it provides more individuals to undergo selection, crossover, and mutation operations in each generation. This increased number of potential parents allows the algorithm to explore more offspring solutions. The algorithm might exploit reasonable solutions more efficiently because of the increased chance of a better solution. This improved exploitation capability reduces the variability in the population over generations and might contribute to a smaller standard deviation.

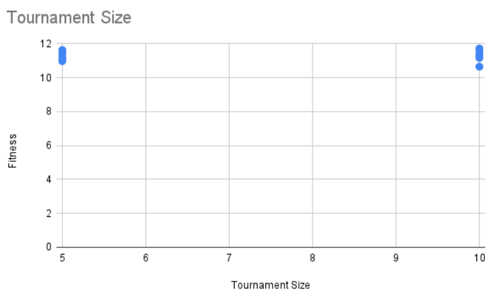
For future work, one should either try to improve the crossover method or try for more generations to improve the fitness results. As discussed later in the Tweaks and Diversity section, the tweak function that provided the best results was the food allocation one. However, for Genetic Algorithm, there isn't any crossover function that tackles the food allocation and that might be why the results are worse than the other algorithms. One could also change the mutation rate during the algorithm, starting with a higher mutation probability, favouring exploration, and slowly decreasing it, favoring solution exploitation. Finally, one could mutate the whole population to escape the local optimum when fitness does not improve for generations. One should also consider the algorithm's complexity and the data structure. This combination makes this approach computationally expensive and is most likely not the best solution.



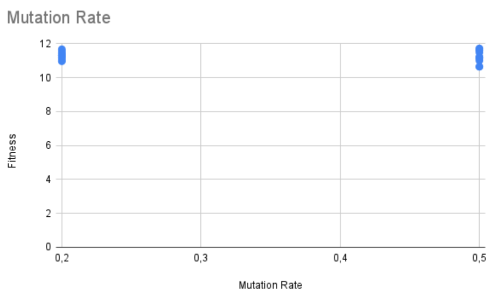
(a) Generations



(b) Population Size



(c) Tournament Size



(d) Mutation Rate

Figure 5.4: Genetic Algorithm Parameters and Fitness

5.1.5 Baseline Algorithms

Considering the fitness results and the time it took to reach them, the algorithm that reached the best results in a shorter time was Simulated Annealing. Regarding time, the Simulated Annealing is the quickest since it does not have to generate neighbourhoods or population. It only generates one solution for each iteration. The second best algorithm is Tabu Search, with fitness results similar to Simulated Annealing with 350 maximum iterations and 40 neighbours, but it takes more than four times longer. With a similar time (between 430-470 seconds), Hill Climbing comes next, and finally, Genetic Algorithm. This algorithm takes quite a long time, and the results are worse than the other algorithms.

In conclusion, the fastest and most efficient algorithm for the simplest version of the problem is Simulated Annealing, followed by Tabu Search, Hill Climbing, and finally, Genetic Algorithm. A recap can be visualized in figure 5.5.

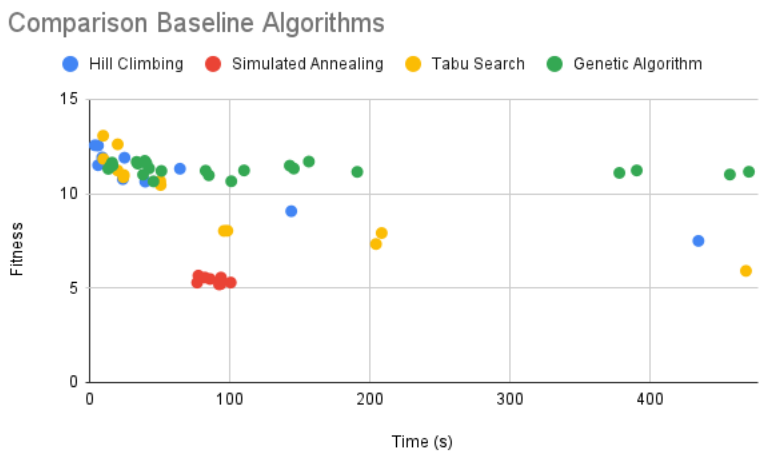


Figure 5.5: Comparison of Baseline Algorithms

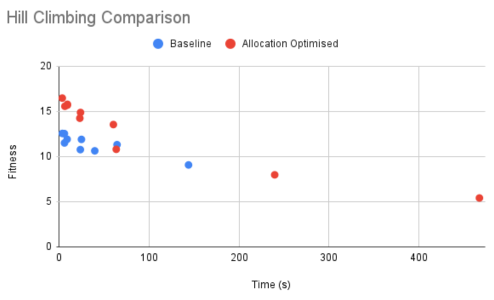
5.1.6 Food Allocation Optimisation

The Food Allocation Optimisation feature was also tested in all algorithms to see if it could improve the results. For both Hill Climbing and Simulated Annealing, the fitness results improved significantly, going from 7.50 to 5.41 (27.87%) and 5.29 to 3.92 (25.90%) in the last iteration, respectively. For Tabu Search, the fitness also improved slightly from 5.91 to 5.50 (6.94%). However, Tabu Search and Hill Climbing started with worse solutions than their simpler counterparts, so these two algorithms evolved more with this feature. Regarding the Genetic Algorithm, the results were worse than its simple version (15.27 to 11.16 in the last iteration - 36.83% increase). Its results in this modality also did not fluctuate much. A visual comparison between this feature and the baseline for all algorithms is in figure 5.6.

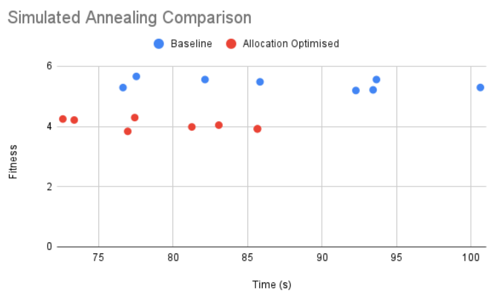
Regarding time, applying this feature for Simulated Annealing, Tabu Search, and Genetic Algorithm made the algorithms run faster. In the case of the Simulated Annealing, it was 14.85% faster, Tabu Search was 38.07% faster, and Genetic Algorithm was 14.11% faster. In the case of Hill Climbing, it took a little longer, 7.53 %, which can be explained by the fact that it generated better solutions, making the algorithm run for more iterations. Comparing the algorithms between each other, the order stays the same as the simple version, having Simulated Annealing with the best results in a shorter amount of time, Tabu Search having similar results as Hill Climbing but taking way less time, and Genetic Algorithm last. With these results, one can conclude that this feature was a success.

5.1.7 Extra Allocation for Food Close to Consumption Date

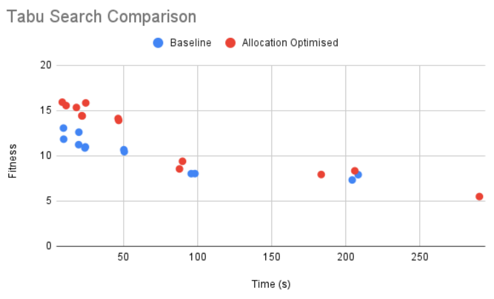
This modality not only allocates the food requested by the consumers but also allocates food close to the consumption date to specific consumers who can receive food in these conditions. Therefore, the fitness results will be bigger than the basic version of the problem since more food and stops are considered in the fitness function.



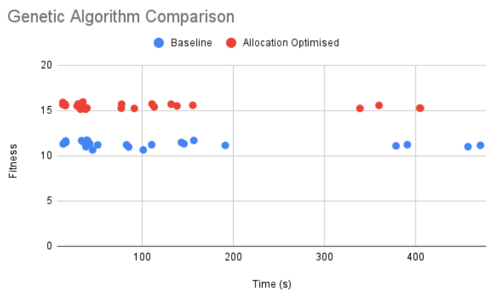
(a) Hill Climbing



(b) Simulated Annealing



(c) Tabu Search



(d) Genetic Algorithm

Figure 5.6: Food Allocation Optimised compared to Baseline

Comparing the different algorithms, both Hill Climbing and Tabu Search have similar times (337.30 and 369.8, respectively), and Tabu Search has a slightly better result than Hill Climbing (9.56 over 10.84). In the case of Simulated Annealing, it took 365.25 seconds to reach a result of 16.53, taking a similar time as Hill Climbing and Tabu Search but a significantly worse result. For the Genetic Algorithm, with only 187.67 seconds, it reached a fitness of 15.53, so it reached better results quicker than Simulated Annealing. A visual representation of this comparison can be checked in figure 5.7. In this case, contrary to the other cases, Tabu Search is slightly better than Hill Climbing, the Genetic Algorithm, and Simulated Annealing.

5.1.8 Route Planning for Several Days Algorithms

Regarding the route planning for several days algorithms, since one solution is a list of RouteSolution objects instead of a single RouteSolution, the algorithms will naturally take longer to run compared to the simple version. Using similar sets of parameters from the baseline, the time of each algorithm also varied more compared to the baselines.

Simulated Annealing only required 215.9 seconds to reach 16.03 of fitness. Both Tabu Search and Genetic Algorithm required more time to reach their final result. Tabu Search reached 16.26 at 512.03 seconds which was a better result than the Genetic Algorithm, which with 518.91 seconds, only achieved 17.40. Regarding Hill Climbing, it reached 16.79 of fitness for 525.28 seconds, with a worse result than Tabu Search and a better result than Genetic Algorithm with similar

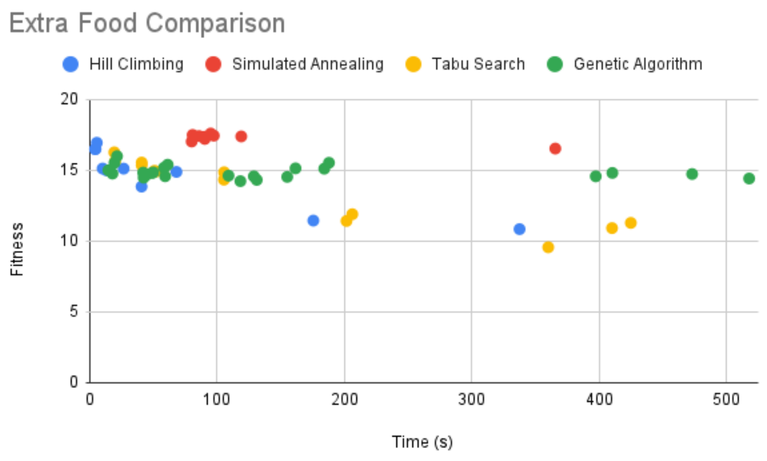


Figure 5.7: Comparison of Extra Food Allocation Close to Consumption Date

times. Therefore, the algorithms' performance order is Simulated Annealing, Tabu Search, Hill Climbing, and Genetic Algorithm, similar to the baseline. In figure 5.8, one can observe how the different algorithms performed in this modality.

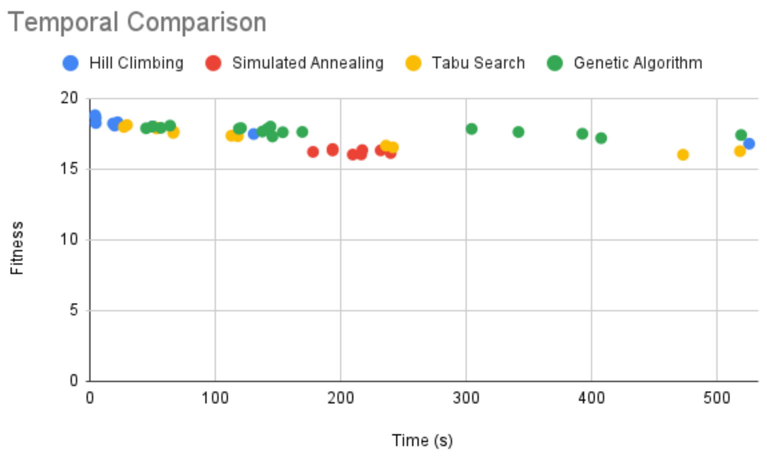


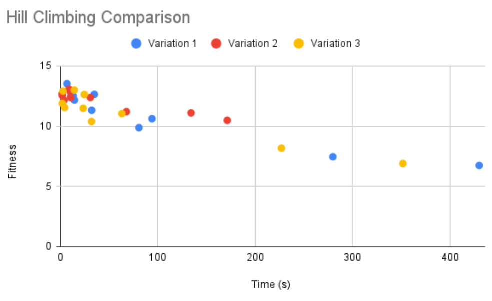
Figure 5.8: Comparison of Temporal Allocation

5.1.9 Tweaks and Diversity

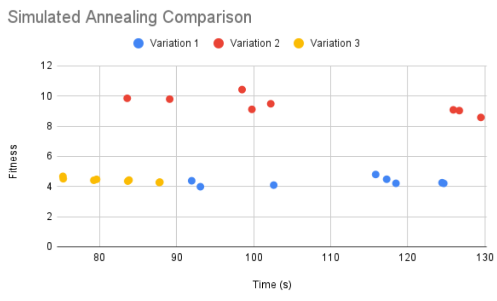
In order to check which tweak function produces better results, Hill Climbing, Simulated Annealing, and Tabu Search were tested with variants of the neighbourhood generation. The first variant includes the food allocation tweak and the stops order tweak, and the second variant includes the car allocation and stops order tweak. The third variant includes the car allocation tweak and the food allocation tweak. Analysing the results, one can see that the variants that produce the best fitness results are Variant 1, with the food allocation tweak and stops order tweak, and Variant 3, with the food allocation tweak and car allocation tweak. These variants have similar fitness results, and one can then conclude that the tweak function that improves the results the most is the food allocation one. The variant with the worst results is Variant 2, with both the car allocation and stops order tweak. Since the values for Variant 1 and 3 are similar, one can assume that the stops order and the car allocation tweaks produce similar results in terms of fitness. A comparison between these variants in the different algorithms can be visualized in figure 5.9

Regarding time, the variant that takes the longest is the first variant, and the variant that takes the shortest amount of time is the third. These results were taken from the Tabu Search and Simulated Annealing's tables since these algorithms go through all the iterations indicated in maximum iterations. For Hill Climbing, since Variant 2 produces worse solutions, the algorithm fails to create improved solutions in the neighbourhood and consequently takes a shorter time to run, stopping earlier. With these values, one can then conclude that the car allocation tweak function is faster than the stops order tweak function.

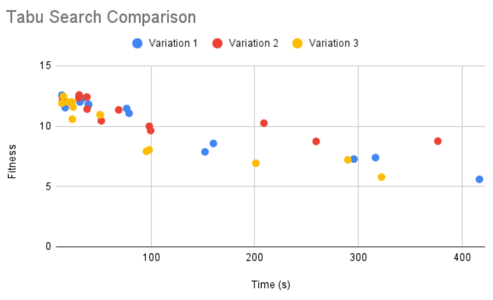
These variants were not tested in the Genetic Algorithm since the core of it is in the crossover of the parents, using different functions other than the tweak ones. The tweak ones are used to mutate the solutions, and they are not called at all times.



(a) Hill Climbing



(b) Simulated Annealing



(c) Tabu Search

Figure 5.9: Tweak Variations Comparison

5.2 Solution Diversity

In order to reach good results, several approaches were applied to increase the diversity of the solutions. First, there are two ways to build a route, either by putting the producer and its consumers first or all the producers and then the consumers. Additionally, when creating a route, there is a 0.5 possibility of optimising the route. If there is more than one stop in a consumer by different producers, it condenses the last one with all the producers. When allocating a consumer to a producer, it is done randomly (when the option for optimal food allocation is false). Furthermore, there are three possible ways to tweak a solution: changing the food allocation, the order of the route, and the cars assigned to the solution. When calling the food allocation tweak and adding the food delivery to the route, it randomly adds the producer in the first five stops and the consumer in a random position in case the route did not stop in those places before.

For the genetic algorithm, when generating a child, there are two possible crossover methods, one tackling the car allocation and one the route order. It has a given probability of suffering mutation with the three tweak functions. However, ideally, three possible crossover methods should have been implemented, adding one tackling the food allocation. These approaches allow to either build solutions that can be widely different from one another or efficiently tweak the solution slightly with the hopes of improving fitness.

5.3 Solution Efficiency

When building a solution, it is crucial to reflect on its efficiency, especially to achieve good results in a reasonable amount of time.

Firstly, Python is not the most efficient choice memory-wise, and it was required to deep copy some data structures defined so that the functions would alter the correct ones. This operation can be time-consuming for complex data structures, as is the case. In addition, initially, there were three ways to build a route, one completely random. However, after testing with the generated dataset, that option had to be removed because it took too long to find a randomized route that fulfilled the order constraint.

Furthermore, the process of optimally allocating food can be tricky since each consumer needs to go to each request, and each producer and each element in its stock has to find the number of days closer to the consumption date. Randomly allocating food is quicker since the first producer that can supply the request is the one that is allocated. The tweak food allocation function can also take a while since it has to find two car routes randomly, pick a food delivery object from the first one, and randomly find a producer to supply that food delivery, and the cycle keeps going until that requirement is fulfilled. The points mentioned can be improved for future work.

5.4 Challenges

Throughout the development of the solution, several challenges were faced. Due to the nature of the problem, a lot of information has to be taken into consideration to blend the economic factor, the environmental factor, and the food waste factor. Given this, the data structure to represent it needs to hold a considerable amount of data, and it entails a more difficult job of applying metaheuristic concepts. For example, to change the solution slightly and to hold diversity, three methods that tackle different parts of the solution, the order of the routes, the food allocation between producers and consumers, and the car allocation, had to be implemented. Furthermore, achieving a viable method for crossover between two solutions proved challenging, taking the order crossover for routes and a similar idea for the car allocation, even though it behaves differently. Finally, having an intricate data structure also meant looking out for tiny technicalities when mixing and changing each RouteSolution.

As seen in the State of the Art section, another attempt at incorporating food waste and environmental impact in a route solution algorithm was not found. Hence, it proved to be challenging to evaluate objectively the results reached. Additionally, finding a dataset that incorporated all the information needed for the problem was impossible due to the aforementioned reason, and the dataset used for the testing had to be generated.

Chapter 6

Conclusions

This chapter summarizes the work developed, recapping the steps taken and if the project achieved the initial goal. The project's contributions are also discussed, and the final section introduces the future work suggested.

6.1 Summary

The first step towards developing the dissertation was figuring out its goals. Afterward, an investigation towards the State of the Art on Route Optimisation applied to Food Waste and the Environmental Impact was carried out using the systematic literature review method. Subsequently, the proposed solution was developed by implementing the simple version, then adding features, such as a volume constraint, allocating food close to consumption date to specific consumers, route planning for several days, and optimising food allocation and routes. Four algorithms were also developed to reach better solutions, Hill Climbing, Simulated Annealing, Tabu Search, and Genetic Algorithm.

After the development stage, several experiments were carried out. These focused on recording each algorithm's fitness results and time to run different sets of parameters, calculating the standard deviation of the fitness results, testing the base option, the food allocation optimisation option, allocating food close to the consumption date, route calculation for several days, and the diversity of the tweak functions. The results were then discussed, taking into account the different algorithms and their behaviour, which algorithm worked best, which parameters contributed the most to the results, which tweak function helped the most achieve better solutions, the overall solution diversity and efficiency, and, finally, the challenges faced when developing the solution.

6.2 Contributions

Considering the initial objectives, the dissertation shows that the study was done on route optimisation algorithms and how it explored ways to minimize the environmental impact. Furthermore, it addressed the need to utilize food waste by allocating food close to the consumption date to specific consumers. Finally, the proposed solution calculates optimised routes incorporating food waste and environmental variables. Thus, all in all, the main goal of the dissertation was completed. The proposed solution considers the economic, food waste, and environmental impact factors and proposes routes based on them. Furthermore, it considers the vehicle's capacity, can allocate food optimally right away, can allocate food close to the consumption date to specific consumers able to receive it, and can plan routes for several days.

Going back to the research question "How to improve route optimisation considering sustainability and food waste minimisation?", the work developed is a proposed answer. As specified previously, no other attempt was found to combine food waste and environmental impact in route optimisation. The results reached in this dissertation can be a first ground in considering food waste when planning route optimisation of produce. From here, this idea can be improved and implemented, decreasing food waste during the transportation of goods and the environmental impact of this necessary activity. Furthermore, this solution also raises awareness of the pressing matters of food waste and the negative impact on the environment from transportation. Hopefully, it motivates others to act on these pressing challenges.

6.3 Future Work

Regarding future work, several considerations can be implemented to enrich the food waste and environmental impact factors. One can consider how loaded the vehicle is when predicting the CO2 emissions and the environmental impact of refrigeration in the vehicles for the environmental factor in the calculations. Additionally, one can consider bad flooring, ramps, and air quality for calculating the food waste factor. Furthermore, one can also make the weights of each factor in the fitness function customizable for the user. Another interesting aspect to add to the solution is considering electric vehicles in the fleet and charging stations. Finally, to further develop the aspect of route optimisation, one can consider as well time windows, which means ensuring deliveries are performed within specific time frames that are acceptable or preferred by the stakeholders and the drivers' time constraints by taking into account the daily working hours and mandatory breaks and rest periods.

In conclusion, the possibilities are varied to construct the solution further, and it is a groundbreaking opportunity to keep investing in applying food waste and environmental impact together with route optimisation.

[backend=biber, style=numeric,]biblatex

References

[1] Food and Agriculture Organization. Food outlook; biannual report on global food markets, November 2017.

[2] ed. FAO. *Moving Forward on Food Loss and Waste Reduction*. Food and Agriculture Organization of the United Nations, Rome, 2019.

[3] United Nations. *World Population Prospects: The 2017 Revision. Key Findings and Advance Tables*. 2017.

[4] United Nations. *Transforming Our World: The 2030 Agenda for Sustainable Development*. United Nations, New York, NY, USA, 2015.

[5] International Energy Agency. *Energy Technology Perspectives 2020: Special Report on Clean Energy Innovation*. International Energy Agency, 2020.

[6] International Energy Agency. Global co2 emissions from fuel combustion. https://www.iea.org/data-and-statistics?country=WORLD&fuel=CO2%20emissions&indicator=Total%20CO2%20emissions%20from%20fuel%20combustion, 2020.

[7] World Health Organization. Ambient air pollution: Health impacts. https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health, 2018. Accessed on February 22, 2023.

[8] International Energy Agency. *Global Energy & CO2 Status Report 2020*. International Energy Agency, 2020. Accessed on February 22, 2023.

[9] Eureka Network. Clusters for sustainability 2022. https://www.eurekanetwork.org/open-calls/clusters-sustainability-2022, 2022. Accessed on 2023-02-03.

[10] Matilde Rodrigues, Brigida Monica Faria, Alexandra Oliveira, and Luis Paulo Reis. SUS2022-071 RETAILL- REtail using Technology based on Artificial InteLLigence, Cluster Eureka ITEA 2022. 2022.

[11] Luís Spínola, Daniel Castro Silva, and Luís Paulo Reis. A Highly Customizable Information Visualization Framework. In *International Conference on Computational Science*, pages 110–116. Springer, 2022.

[12] Henrique Lopes-Cardoso, Tomás Freitas Osório, Luís Vilar Barbosa, Gil Rocha, Luís Paulo Reis, João Pedro Machado, and Ana Maria Oliveira. Robust complaint processing in portuguese. *Information*, 12(12):525, 2021. Publisher: MDPI.

[13] Tatiana Sousa Pinto, Brigida Monica Faria, Luis Paulo Reis, Henrique Lopes Cardoso, and Tiago Santos. Compliance study of hazard analysis and critical control point system. *International Conference Big Data Analytics, Data Mining and Computational Intelligence 2019*, pages 111–118, 2019. Accepted: 2023-06-12T10:38:28Z ISBN: 9789898533920 Publisher: IADIS Publications.

[14] Gustavo Magalhães, Brígida Faria, Luís Reis, and Henrique Lopes Cardoso. TEXT MINING APPLICATIONS TO FACILITATE ECONOMIC AND FOOD SAFETY LAW ENFORCEMENT. pages 199–203, July 2019.

[15] Gustavo Magalhães, Brígida Mónica Faria, Luís Paulo Reis, Henrique Lopes Cardoso, Cristina Caldeira, and Ana Oliveira. Automating Complaints Processing in the Food and Economic Sector: A Classification Approach. In Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, Sandra Costanzo, Irena Orovic, and Fernando Moreira, editors, *Trends and Innovations in Information Systems and Technologies*, Advances in Intelligent Systems and Computing, pages 445–456, Cham, 2020. Springer International Publishing.

[16] Joao Filgueiras, Luís Barbosa, Gil Rocha, Henrique Lopes Cardoso, Luís Paulo Reis, Joao Pedro Machado, and Ana Maria Oliveira. Complaint analysis and classification for economic and food safety. In *Proceedings of the Second Workshop on Economics and Natural Language Processing*, pages 51–60, 2019.

[17] 1000EcoFarms. Find and buy natural farm-produced food near you. https://www.1000ecofarms.com//en, October 2021. Accessed on 2022-10-05.

[18] Barn2Door. BARN2DOOR. https://www.barn2door.com, 2022. Accessed on 2023-02-04.

[19] CSAware. CSA Software | CSAware. https://www.csaware.com/, 2022. Accessed on 2023-02-04.

[20] Eat From Farms. Home - Eat From Farms. https://www.eatfromfarms.com/. Accessed on 2023-02-04.

[21] Harvie. Harvie. http://www.harvie.farm/, 2023. Accessed on 2023-02-04.

[22] Copernicus. Farm Sustainability Tool (FaST) - Space Data for Sustainable Farming | Copernicus. https://www.copernicus.eu/en/use-cases/farm-sustainability-tool-fast-space-data-sustainable-farming. Accessed on 2023-02-04.

[23] Community Supported Agriculture. https://communitysupportedagriculture.org.uk/. https://communitysupportedagriculture.org.uk/, February 2023. Accessed on 2023-02-04.

[24] Agri Marketplace. Agri Marketplace - Fairtrade made easy. https://agrimp.com, 2023. Accessed on 2023-02-04.

[25] Routific. Delivery Route Planning & Route Optimization Software. https://routific.com/. Accessed on 2023-02-04.

[26] OptimoRoute. OptimoRoute | Delivery Route Planning & Field Service Scheduling. https://optimoroute.com/, 2023. Accessed on 2023-02-04.

[27] RoadWarrior. Fleet App Delivery Management for Teams. https://roadwarrior.app/teams/, 2023. Accessed on 2023-02-04.

[28] AntsRoute. AntsRoute | Last mile route optimisation software. https://antsroute.com/en/. Accessed on 2023-02-04.

[29] Verizon Connect. Fleet Management Software and Solutions | Verizon Connect. https://www.verizonconnect.com/, 2023. Accessed on 2023-02-04.

[30] Onfleet. Onfleet - Delightful delivery management software. https://onfleet.com, 2023. Accessed on 2023-02-04.

[31] Route4Me Route Planner. Route planning and optimization apps for businesses. https://www.route4me.com, 2023. Accessed on 2023-02-04.

[32] Too Good To Go. Save Food - Help The Planet. https://toogoodtogo.com/en-us. Accessed on 2023-02-04.

[33] Goodafter. Supermercado Contra o Desperdício. https://goodafter.com/pt/. Accessed on 2023-02-04.

[34] OLIO. Become a zero food waste business with OLIO. https://olioex.com/businesses/, 2022. Accessed on 2023-02-04.

[35] Flashfood. Flashfood | Save money and reduce food waste. https://www.flashfood.com/How-it-works, 2022. Accessed on 2023-02-04.

[36] Food Rescue US. Home. https://foodrescue.us/, 2023. Accessed on 2023-02-04.

[37] @YourKarmaApp. Karma App - Save food with a tap. https://karma.life/, 2021. Accessed on 2023-02-04.

[38] FoodCloud. FoodCloud: Food waste hurts our planet. https://food.cloud/. Accessed on 2023-02-04.

[39] NoWaste. Home. https://www.nowasteapp.com, 2017. Accessed on 2023-02-04.

[40] nosh. nosh - Revolutionizing food consumption and reducing food waste at home. https://nosh.tech/, 2021. Accessed on 2023-02-04.

[41] Kitche. Food Waste App | Kitchen App | Kitche App | Food Waste, Kitche. https://kitche.co/, March 2019. Accessed on 2023-02-04.

[42] Justin Ehlert. Fridgely - Food Expiration Date Tracker. https://fridgelyapp.com. Accessed on 2023-02-04.

[43] ScienceDirect. Life Cycle Assessment - an overview | ScienceDirect Topics. https://www.sciencedirect.com/topics/earth-and-planetary-sciences/life-cycle-assessment, 2017. Accessed on 2023-02-04.

[44] SimaPro. About SimaPro. https://simapro.com/about/, 2023. Accessed on 2023-02-04.

[45] Sphera. Product Sustainability Software & Data. https://sphera.com/product-sustainability-software/, 2023. Accessed on 2023-02-04.

[46] openLCA. openLCA.org | openLCA is a free, professional Life Cycle Assessment (LCA) and footprint software with a broad range of features and many available databases, created by GreenDelta since 2006. https://www.openlca.org/, 2022. Accessed on 2023-02-04.

[47] Ecochain. Solutions. https://ecochain.com/solutions/, 2022. Accessed on 2023-02-04.

[48] iPoint. Product Compliance and Sustainability Software. https://www.ipoint-systems.com/software/, 2023. Accessed on 2023-02-05.

[49] Miguel Milheiro Pinto Ferreira. Disruption Management of ASAE's Inspection Routes. 2021.

[50] Telmo Barros, Alexandra Oliveira, Henrique Lopes Cardoso, Luís Paulo Reis, Cristina Caldeira, and João Pedro Machado. Economic and Food Safety: Optimized Inspection Routes Generation. In *Agents and Artificial Intelligence: 12th International Conference, ICAART 2020, Valletta, Malta, February 22–24, 2020, Revised Selected Papers 12*, pages 482–503. Springer, 2021.

[51] Henrique Lopes Cardoso, Luıs Paulo Reis, Cristina Caldeira, and Ana Oliveira. Online Geocoding of Millions of Economic Operators. *Trends and Innovations in Information Systems and Technologies: Volume 1*, 1159:426, 2020. Publisher: Springer Nature.

[52] Telmo Barros, Alexandra Oliveira, Henrique Lopes Cardoso, Luís Paulo Reis, Ana Cristina Caldeira, and João Pedro Machado. Generation and Optimization of Inspection Routes for Economic and Food Safety. In *ICAART (2)*, pages 268–278, 2020.

[53] Telmo Barros, Tiago Santos, Alexandra Oliveira, Henrique Lopes Cardoso, Luís Paulo Reis, Cristina Caldeira, and João Pedro Machado. Interactive inspection routes application for economic and food safety. In *Trends and Innovations in Information Systems and Technologies: Volume 1 8*, pages 640–649. Springer, 2020.

[54] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, September 2016.

[55] G. Clarke and J. W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568–581, August 1964. Publisher: INFORMS.

[56] Canhong Lin, K. L. Choy, G. T. S. Ho, S. H. Chung, and H. Y. Lam. Survey of Green Vehicle Routing Problem: Past and future trends. *Expert Systems with Applications*, 41(4, Part 1):1118–1138, March 2014.

[57] G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6(1):80–91, October 1959. Publisher: INFORMS.

[58] Christophe Lecluyse, Kenneth Sörensen, and Herbert Peremans. A network-consistent time-dependent travel time layer for routing optimization problems. *European Journal of Operational Research*, 226(3):395–413, May 2013.

[59] A. L. Kok, E. W. Hans, and J. M. J. Schutten. Vehicle routing under time-dependent travel times: The impact of congestion avoidance. *Computers & Operations Research*, 39(5):910–918, May 2012.

[60] Sophie N. Parragh, Karl F. Doerner, and Richard F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, April 2008.

[61] Sophie N. Parragh, Karl F. Doerner, and Richard F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2):81–117, June 2008.

[62] Frank A. Tillman. The Multiple Terminal Delivery Problem with Probabilistic Demands. *Transportation Science*, 3(3):192–204, August 1969. Publisher: INFORMS.

[63] Jacques Renaud, Gilbert Laporte, and Fayez F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23(3):229–235, March 1996.

[64] Michel Gendreau, Gilbert Laporte, and René Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, January 1996.

[65] CDT Watson-Gandy and PJ Dohrn. Depot location with van salesmen — A practical approach. *Omega*, 1(3):321–329, June 1973.

[66] Roberto Baldacci, Aristide Mingozzi, and Roberto Wolfler Calvo. An Exact Method for the Capacitated Location-Routing Problem. *Operations Research*, October 2011. Publisher: INFORMS.

[67] Harilaos N. Psaraftis. A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem. *Transportation Science*, May 1980. Publisher: INFORMS.

[68] Gianpaolo Ghiani, Francesca Guerriero, Gilbert Laporte, and Roberto Musmanno. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151(1):1–11, November 2003.

[69] E. J. Beltrami and L. D. Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65–94, 1974. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230040106.

[70] Jesús Alegre, Manuel Laguna, and Joaquín Pacheco. Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. *European Journal of Operational Research*, 179(3):736–746, June 2007.

[71] M. Dror, M. Ball, and B. Golden. Computational comparison of algorithms for inventory routing. *Annals of Operations Research*, 4:3–23, 1985.

[72] Claudia Archetti, Luca Bertazzi, Gilbert Laporte, and Maria Grazia Speranza. A Branch-and-Cut Algorithm for a Vendor-Managed Inventory-Routing Problem. *Transportation Science*, August 2007. Publisher: INFORMS.

[73] Roberto Baldacci, Maria Battarra, and Daniele Vigo. Valid inequalities for the fleet size and mix vehicle routing problem with fixed costs. *Networks*, 54(4):178–189, 2009. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.20331.

[74] Shuguang Liu, Weilai Huang, and Huiming Ma. An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review*, 45(3):434–445, May 2009.

[75] Gianpaolo Ghiani and Gennaro Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, 122(1):11–17, April 2000.

[76] E.D. Chajakis and M. Guignard. Scheduling deliveries in vehicles with multiple compartments. *Journal of Global Optimization*, 26(1):43–78, 2003.

[77] A. El fallahi, C. Prins, and R. Wolfler Calvo. A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers and Operations Research*, 35(5):1725–1741, 2008.

[78] I-Ming Chao and Tian-Shy Liou. A New Tabu Search Heuristic for the Site-Dependent Vehicle Routing Problem. In Bruce Golden, S. Raghavan, and Edward Wasil, editors, *The Next Wave in Computing, Optimization, and Decision Technologies*, Operations Research/-Computer Science Interfaces Series, pages 107–119, Boston, MA, 2005. Springer US.

[79] Moshe Dror and Pierre Trudeau. Savings by Split Delivery Routing. *Transportation Science*, May 1989. Publisher: INFORMS.

[80] Claudia Archetti, Martin W. P. Savelsbergh, and M. Grazia Speranza. Worst-Case Analysis for Split Delivery Vehicle Routing Problems. *Transportation Science*, May 2006. Publisher: INFORMS.

[81] Yongshuang Zheng and Baoding Liu. Fuzzy vehicle routing model with credibility measure and its hybrid intelligent algorithm. *Applied Mathematics and Computation*, 176(2):673–683, May 2006.

[82] Jiafu Tang, Zhendong Pan, Richard Y. K. Fung, and Henry Lau. Vehicle routing problem with fuzzy time windows. *Fuzzy Sets and Systems*, 160(5):683–695, March 2009.

[83] D Sariklis and S Powell. A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, 51(5):564–573, May 2000. Publisher: Taylor & Francis _eprint: https://doi.org/10.1057/palgrave.jors.2600924.

[84] José Brandão. A tabu search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 157(3):552–564, September 2004.

[85] Shaul P. Ladany and Avraham Mehrez. Optimal routing of a single vehicle with loading and unloading constraints. *Transportation Planning and Technology*, 8(4):301–306, January 1984. Publisher: Routledge _eprint: https://doi.org/10.1080/03081068408717261.

[86] Emmanouil E. Zachariadis, Christos D. Tarantilis, and Chris T. Kiranoudis. The Pallet-Packing Vehicle Routing Problem. *Transportation Science*, 46(3):341–358, August 2012. Publisher: INFORMS.

[87] Guido Perboli, Roberto Tadei, and Daniele Vigo. The Two-Echelon Capacitated Vehicle Routing Problem: Models and Math-Based Heuristics. *Transportation Science*, June 2011. Publisher: INFORMS.

[88] Brian Kallehauge, Jesper Larsen, Oli B.G. Madsen, and Marius M. Solomon. Vehicle Routing Problem with Time Windows. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, *Column Generation*, pages 67–98. Springer US, Boston, MA, 2005.

[89] Miguel Andres Figliozzi. An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. *Transportation Research Part C: Emerging Technologies*, 18(5):668–679, October 2010.

[90] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, June 1992.

[91] Gilbert Laporte, Michel Gendreau, Jean-Yves Potvin, and Frédéric Semet. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7(4):285–300, September 2000.

[92] J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53(5):512–522, 2002.

[93] P. Singh, Z. Elmi, Y.-Y. Lau, M. Borowska-Stefańska, S. Wiśniewski, and M.A. Dulebenets. Blockchain and AI technology convergence: Applications in transportation systems. *Vehicular Communications*, 38, 2022.

[94] A.W. Sadek. Artificial intelligence applications in transportation. *Transportation Research Circular, Artificial Intelligence in Transportation*, pages 1–6, 2007.

[95] Aymen Aloui, Nadia Hamani, Ridha Derrouiche, and Laurent Delahoche. Systematic literature review on collaborative sustainable transportation: overview, analysis and perspectives. *Transportation Research Interdisciplinary Perspectives*, 9:100291, March 2021.

[96] T. Bektaş, J.F. Ehmke, H.N. Psaraftis, and J. Puchinger. The role of operational research in green freight transportation. *European Journal of Operational Research*, 274(3):807–823, 2019.

[97] E. Demir, T. Bektaş, and G. Laporte. A review of recent research on green road freight transportation. *European Journal of Operational Research*, 237(3):775–793, 2014.

[98] H. Rakha, K. Ahn, and A. Trani. Comparison of MOBILE5a, MOBILE6, VT-MICRO, and CMEM models for estimating hot-stabilized light-duty gasoline vehicle emissions. *Canadian Journal of Civil Engineering*, 30(6):1010–1021, 2003.

[99] Sevgi Erdoğan and Elise Miller-Hooks. A Green Vehicle Routing Problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100–114, January 2012.

[100] Tolga Bektaş and Gilbert Laporte. The Pollution-Routing Problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250, September 2011.

[101] J Bauer, T Bektaş, and T G Crainic. Minimizing greenhouse gas emissions in intermodal freight transport: an application to rail service design. *Journal of the Operational Research Society*, 61(3):530–542, March 2010. Publisher: Taylor & Francis _eprint: https://doi.org/10.1057/jors.2009.102.

[102] Rommert Dekker, Moritz Fleischmann, Karl Inderfurth, and Luk N. van Wassenhove. *Reverse Logistics: Quantitative Models for Closed-Loop Supply Chains*. Springer Science & Business Media, January 2004. Google-Books-ID: 6i4b1F0zX90C.

[103] Patrick Beullens, Dirk Van Oudheusden, and Luk N. Van Wassenhove. Collection and Vehicle Routing Issues in Reverse Logistics. In Rommert Dekker, Moritz Fleischmann, Karl Inderfurth, and Luk N. Van Wassenhove, editors, *Reverse Logistics: Quantitative Models for Closed-Loop Supply Chains*, pages 95–134. Springer, Berlin, Heidelberg, 2004.

[104] Reza Moghdani, Khodakaram Salimifard, Emrah Demir, and Abdelkader Benyettou. The green vehicle routing problem: A systematic literature review. *Journal of Cleaner Production*, 279:123691, January 2021.

[105] L. Matijević. METAHEURISTIC APPROACHES FOR THE GREEN VEHICLE ROUTING PROBLEM. *Yugoslav Journal of Operations Research*, 33(2):153–198, 2023.

[106] N. Elgharably, S. Easa, A. Nassef, and A. El Damatty. Stochastic Multi-Objective Vehicle Routing Model in Green Environment With Customer Satisfaction. *IEEE Transactions on Intelligent Transportation Systems*, 24(1):1337–1355, 2023.

[107] A. Oumachtaq, L. Ouzizi, and M. Douimi. Green Vehicle Routing Problem (GVRP): State-of-the-Art. *Lecture Notes in Mechanical Engineering*, pages 406–425, 2023. ISBN: 9783031236143.

[108] Y. Xiao, Y. Zhang, I. Kaku, R. Kang, and X. Pan. Electric vehicle routing problem: A systematic review and a new comprehensive model with nonlinear energy recharging and consumption. *Renewable and Sustainable Energy Reviews*, 151, 2021.

[109] Xiaorong Zuo, Yiyong Xiao, Meng You, Ikou Kaku, and Yuchun Xu. A new formulation of the electric vehicle routing problem with time windows considering concave nonlinear charging function. *Journal of Cleaner Production*, 236:117687, November 2019.

[110] Hamed Soleimani, Yusof Chaharlang, and Hadi Ghaderi. Collection and distribution of returned-remanufactured products in a vehicle routing problem with pickup and delivery considering sustainable and green criteria. *Journal of Cleaner Production*, 172:960–970, January 2018.

[111] Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. Exact Algorithms for Electric Vehicle-Routing Problems with Time Windows. *Operations Research*, 64(6):1388–1405, December 2016. Publisher: INFORMS.

[112] Aala Kalananda Vamsi Krishna Reddy and Komanapalli Venkata Lakshmi Narayana. Metaheuristics optimization in electric vehicles -an extensive review. *Renewable and Sustainable Energy Reviews*, 160:112285, May 2022.

[113] M. Asghari and S.M.J. Mirzapour Al-e hashem. Green vehicle routing problem: A state-of-the-art review. *International Journal of Production Economics*, 231, 2021.

[114] Samuel Pelletier, Ola Jabali, and Gilbert Laporte. 50th Anniversary Invited Article—Goods Distribution with Electric Vehicles: Review and Research Perspectives. *Transportation Science*, February 2016. Publisher: INFORMS.

[115] Chong Ye, Wenjie He, and Hanqi Chen. Electric vehicle routing models and solution algorithms in logistics distribution: A systematic review. *Environmental Science and Pollution Research*, 29(38):57067–57090, August 2022.

[116] Julia Kleineidam. Fields of Action for Designing Measures to Avoid Food Losses in Logistics Networks. *Sustainability*, 12(15):6093, January 2020. Number: 15 Publisher: Multidisciplinary Digital Publishing Institute.

[117] A. English. *The State of Food and Agriculture*. Food and Agriculture Organization of the United Nations, Rome, Italy, 2019.

[118] Maria Curie-Skłodowska University, Lublin, Poland and Aleksandra Kowalska. The issue of food losses and waste and its determinants. *Logforum*, 13(1), March 2017.

[119] Vanessa S. M. Magalhães, Luís Miguel D. F. Ferreira, and Cristóvão Silva. Causes and mitigation strategies of food loss and waste: A systematic literature review and framework development. *Sustainable Production and Consumption*, 28:1580–1599, October 2021.

[120] Marie Mourad. Recycling, recovering and preventing "food waste": competing solutions for food systems sustainability in the United States and France. *Journal of Cleaner Production*, 126:461–477, July 2016.

[121] Effie Papargyropoulou, Rodrigo Lozano, Julia K. Steinberger, Nigel Wright, and Zaini bin Ujang. The food waste hierarchy as a framework for the management of food surplus and food waste. *Journal of Cleaner Production*, 76:106–115, August 2014.

[122] Anais Lemaire and Sabine Limbourg. How can food loss and waste management achieve sustainable development goals? *Journal of Cleaner Production*, 234:1221–1234, October 2019.

[123] Claudio Beretta, Franziska Stoessel, Urs Baier, and Stefanie Hellweg. Quantifying food losses and the potential for reduction in Switzerland. *Waste Management*, 33(3):764–773, March 2013.

[124] Carlos Mena, Leon A. Terry, Adrian Williams, and Lisa Ellram. Causes of waste across multi-tier supply networks: Cases in the UK food sector. *International Journal of Production Economics*, 152:144–158, June 2014.

[125] Food and Agriculture Organization. *The State of Food Security and Nutrition in the World.* Building Resilience for Food and Food Security. FAO, Rome, Italy, 2017.

[126] Ceren Hiç, Prajal Pradhan, Diego Rybski, and Jürgen P. Kropp. Food Surplus and Its Climate Burdens. *Environmental Science & Technology*, 50(8):4269–4277, April 2016.

[127] Samir Gokarn and Thyagaraj S. Kuthambalayan. Analysis of challenges inhibiting the reduction of waste in food supply chain. *Journal of Cleaner Production*, 168:595–604, December 2017.

[128] M. Soysal, J. M. Bloemhof-Ruwaard, M. P. M. Meuwissen, and J. G. A. J. van der Vorst. A Review on Quantitative Models for Sustainable Food Logistics Management. *International Journal on Food System Dynamics*, 3(2):136–155, December 2012. Number: 2.

[129] Tómas Hafliðason, Guðrún Ólafsdóttir, Sigurður Bogason, and Gunnar Stefánsson. Criteria for temperature alerts in cod supply chains. *International Journal of Physical Distribution & Logistics Management*, 42(4):355–371, January 2012. Publisher: Emerald Group Publishing Limited.

[130] K. van Donselaar, T. van Woensel, R. Broekmeulen, and J. Fransoo. Inventory control of perishables in supermarkets. *International Journal of Production Economics*, 104(2):462–472, December 2006.

[131] Milena Lipińska, Marzena Tomaszewska, and Danuta Kołożyn-Krajewska. Identifying Factors Associated with Food Losses during Transportation: Potentials for Social Purposes. *Sustainability*, 11(7):2046, January 2019. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.

[132] Dong Li, Xiaojun Wang, Hing Kai Chan, and Riccardo Manzini. Sustainable food supply chain management. *International Journal of Production Economics*, 152:1–8, June 2014.

[133] E. Valli, R. Manzini, R. Accorsi, M. Bortolini, M. Gamberi, A. Bendini, G. Lercker, and T. Gallina Toschi. Quality at destination: Simulating shipment of three bottled edible oils from Italy to Taiwan. *Rivista Italiana delle Sostanze Grasse*, 90(3):163–169, 2013.

[134] Benjamin K. Sovacool, Morgan Bazilian, Steve Griffiths, Jinsoo Kim, Aoife Foley, and David Rooney. Decarbonizing the food and beverages industry: A critical and systematic review of developments, sociotechnical systems and policy options. *Renewable and Sustainable Energy Reviews*, 143:110856, June 2021.

[135] Sandeep Jagtap, Shahin Rahimifard, and Linh N. K. Duong. Real-time data collection to improve energy efficiency: A case study of food manufacturer. *Journal of Food Processing and Preservation*, 46(8):e14338, 2022. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jfpp.14338.

[136] Ayten Aylin Alsaffar. Sustainable diets: The interaction between food industry, nutrition, health and the environment. *Food Science and Technology International*, 22(2):102–111, March 2016. Publisher: SAGE Publications Ltd STM.

[137] Franklin Carrero-Martínez, Emi Kameyama, and Paula Tarnapol Whitacre, editors. *Reducing Impacts of Food Loss and Waste: Proceedings of a Workshop*. National Academies Press, Washington, D.C., May 2019.

[138] S. J. James and C. James. The food cold-chain and climate change. *Food Research International*, 43(7):1944–1956, August 2010.

[139] Thor Benson. Coronavirus vs. Climate Change - IEEE Spectrum. https://spectrum.ieee.org/covid19-pandemic-reduce-greenhouse-gas-emissions. Accessed on 2023-02-01.

[140] A. Mittal, C.C. Krejci, and T.J. Craven. Logistics best practices for regional food systems: A review. *Sustainability (Switzerland)*, 10(1), 2018.

[141] Steve Martinez, Michael Hand, Michelle Da Pra, Susan Pollack, Katherine Ralston, Travis Smith, Stephen Vogel, Shellye Clark, Luanne Lohr, Sarah Low, and Constance Newman. Local Food Systems: Concepts, Impacts, and Issues.

[142] Steven M. Schnell. Food miles, local eating, and community supported agriculture: putting local food in its place. *Agriculture and Human Values*, 30(4):615–628, December 2013.

[143] Corinna Feldmann and Ulrich Hamm. Consumers' perceptions and preferences for local food: A review. *Food Quality and Preference*, 40:152–164, March 2015.

[144] Sarah A. Low, Aaron Adalja, Elizabeth Beaulieu, Nigel Key, Stephen Martinez, Alex Melton, Agnes Perez, Katherine Ralston, Hayden Stewart, Shellye Suttles, Stephen Vogel, and Becca B. R. Jablonski. Trends in U.S. Local and Regional Food Systems: A Report to Congress. http://www.ers.usda.gov/publications/pub-details/?pubid=42807. Accessed on 2023-02-01.

[145] Shermain D. Hardesty. The Growing Role of Local Food Markets. *American Journal of Agricultural Economics*, 90(5):1289–1295, 2008. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8276.2008.01219.x.

[146] Lindsey Day-Farnsworth and Michelle Miller. Networking Across the Supply Chain: Transportation Innovations in Local and Regional Food Systems. Technical report, University of Wisconsin, June 2014.

[147] Claudia Paciarotti and Francesco Torregiani. The logistics of the short food supply chain: A literature review. *Sustainable Production and Consumption*, 26:428–442, April 2021.

[148] Biancamaria Torquati, Chiara Taglioni, and Alessio Cavicchi. Evaluating the CO2 Emission of the Milk Supply Chain in Italy: An Exploratory Study. *Sustainability*, 7(6):7245–7260, June 2015. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[149] Garry W Auld, Dawn Thilmany, and Katie Jones. Factors Affecting Small Colorado Producers' Local Food Sales. *Journal of Hunger & Environmental Nutrition*, 4(2):129–146, May 2009. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/19320240902915284.

[150] A. Jones. An environmental assessment of food supply chains: A case study on dessert apples. *Environmental Management*, 30(4):560–576, 2002.

[151] Rich Pirog, Timothy Van Pelt, Kamyar Enshayan, and Ellen Cook. Food, fuel, and freeways: An iowa perspective on how far food travels, fuel usage, and greenhouse gas emissions. 2001.

[152] Ari Paloviita. Consumers' Sustainability Perceptions of the Supply Chain of Locally Produced Food. *Sustainability*, 2(6):1492–1509, June 2010. Number: 6 Publisher: Molecular Diversity Preservation International.

[153] Michal Kulak, Thomas Nemecek, Emmanuel Frossard, Véronique Chable, and Gérard Gaillard. Life cycle assessment of bread from several alternative food networks in Europe. *Journal of Cleaner Production*, 90:104–113, March 2015.

[154] A Malak-Rawlikowska, E Majewski, A Was, SO Borgen, P Csillag, M Donati, R Freeman, V Hoang, JL Lecoeur, MC Mancini, et al. Measuring the economic, environmental, and social sustainability of short food supply chains. sustainability, 11 (15), 4004, 2019.

[155] Dataset Search. https://datasetsearch.research.google.com/. Accessed on 2023-06-07.

[156] Home - VRP-REP: the vehicle routing problem repository. http://www.vrp-rep.org/. Accessed on 2023-06-07.

[157] Aldy Gunawan, Graham Kendall, Barry McCollum, Hsin-Vonn Seow, and Lai Soon Lee. Vehicle routing: Review of benchmark datasets. *Journal of the Operational Research Society*, 72(8):1794–1807, August 2021.

[158] Openrouteservice. https://openrouteservice.org/. Accessed on 2023-06-07.

[159] Google Maps Platform APIs by Platform. https://developers.google.com/maps/apis-by-platform. Accessed on 2023-06-07.

[160] Bing Maps API | Free Trial, Explore Features, Talk to Specialists. https://www.microsoft.com/en-us/maps/choose-your-bing-maps-api, February 2022. Accessed on 2023-06-17.

[161] API Docs. https://docs.mapbox.com/api/overview/. Accessed on 2023-06-17.

[162] Life cycle assessment - an overview | ScienceDirect topics. https://www.sciencedirect.com/topics/earth-and-planetary-sciences/life-cycle-assessment.

[163] İmdat Kara, Bahar Y. Kara, and M. Kadri Yetis. Energy Minimizing Vehicle Routing Problem. In Andreas Dress, Yinfeng Xu, and Binhai Zhu, editors, *Combinatorial Optimization and Applications*, Lecture Notes in Computer Science, pages 62–71, Berlin, Heidelberg, 2007. Springer.

[164] Yong-Ju Kwon, Young-Jae Choi, and Dong-Ho Lee. Heterogeneous fixed fleet vehicle routing considering carbon emission. *Transportation Research Part D: Transport and Environment*, 23:81–89, August 2013.

[165] Heiko W. Kopfer, Jörn Schönberger, and Herbert Kopfer. Reducing greenhouse gas emissions of a heterogeneous vehicle fleet. *Flexible Services and Manufacturing Journal*, 26(1):221–248, June 2014.

[166] Hamid Allaoui, Yuhan Guo, and Joseph Sarkis. Decision support for collaboration planning in sustainable supply chains. *Journal of Cleaner Production*, 229:761–774, August 2019.

[167] Thomas Chabot, Florence Bouchard, Ariane Legault-Michaud, Jacques Renaud, and Leandro C. Coelho. Service level, cost and environmental optimization of collaborative transportation. *Transportation Research Part E: Logistics and Transportation Review*, 110:1–14, February 2018.

[168] Rafik Makhloufi, Diego Cattaruzza, Frédéric Meunier, Nabil Absi, and Dominique Feillet. Simulation of Mutualized Urban Logistics Systems with Real-time Management. *Transportation Research Procedia*, 6:365–376, January 2015.

[169] Jairo R. Montoya-Torres, Andrés Muñoz-Villamizar, and Carlos A. Vega-Mejía. On the impact of collaborative strategies for goods delivery in city logistics. *Production Planning & Control*, 27(6):443–455, April 2016. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/09537287.2016.1147092.

[170] Abdelhamid Moutaoukil, Gilles Neubert, and Ridha Derrouiche. Urban Freight Distribution: The impact of delivery time on sustainability. *IFAC-PapersOnLine*, 48(3):2368–2373, January 2015.

[171] Andrés Muñoz-Villamizar, Carlos L. Quintero-Araújo, Jairo R. Montoya-Torres, and Javier Faulin. Short- and mid-term evaluation of the use of electric vehicles in urban freight transport collaborative networks: a case study. *International Journal of Logistics Research and Applications*, 22(3):229–252, May 2019. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/13675567.2018.1513467.

[172] Andrés Muñoz-Villamizar, Javier Santos, Jairo R. Montoya-Torres, and Josué C. Velázquez-Martínez. Measuring environmental performance of urban freight transport systems: A case study. *Sustainable Cities and Society*, 52:101844, January 2020.

[173] Sachin Nataraj, Daniele Ferone, Carlos Quintero-Araujo, Angel A. Juan, and Paola Festa. Consolidation centers in city logistics: A cooperative approach based on the location routing problem. *International Journal of Industrial Engineering Computations*, pages 393–404, 2019.

[174] Hanan Ouhader and Malika El Kyal. Combining Facility Location and Routing Decisions in Sustainable Urban Freight Distribution under Horizontal Collaboration: How Can Shippers Be Benefited? *Mathematical Problems in Engineering*, 2017:e8687515, July 2017. Publisher: Hindawi.

[175] Elena Pérez-Bernabeu, Angel A. Juan, Javier Faulin, and Barry B. Barrios. Horizontal cooperation in road transportation: a case illustrating savings in distances and greenhouse gas emissions. *International Transactions in Operational Research*, 22(3):585–606, 2015. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.12130.

[176] Carlos L. Quintero-Araujo, Aljoscha Gruler, Angel A. Juan, and Javier Faulin. Using horizontal cooperation concepts in integrated routing and facility-location decisions. *International Transactions in Operational Research*, 26(2):551–576, 2019. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.12479.

[177] Mehmet Soysal, Jacqueline M. Bloemhof-Ruwaard, Rene Haijema, and Jack G. A. J. van der Vorst. Modeling a green inventory routing problem for perishable products with horizontal collaboration. *Computers & Operations Research*, 89:168–182, January 2018.

[178] Helena M. Stellingwerf, Gilbert Laporte, Frans C. A. M. Cruijssen, Argyris Kanellopoulos, and Jacqueline M. Bloemhof. Quantifying the environmental and economic benefits of cooperation: A case study in temperature-controlled food logistics. *Transportation Research Part D: Transport and Environment*, 65:178–193, December 2018.

[179] H. M. Stellingwerf, A. Kanellopoulos, F. C. A. M. Cruijssen, and J. M. Bloemhof. Fair gain allocation in eco-efficient vendor-managed inventory cooperation. *Journal of Cleaner Production*, 231:746–755, September 2019.

[180] Yong Wang, Shuanglu Zhang, Kevin Assogba, Jianxin Fan, Maozeng Xu, and Yinhai Wang. Economic and environmental evaluations in the two-echelon collaborative multiple centers vehicle routing optimization. *Journal of Cleaner Production*, 197:443–461, October 2018.

[181] Emrah Demir, Tolga Bektaş, and Gilbert Laporte. The bi-objective Pollution-Routing Problem. *European Journal of Operational Research*, 232(3):464–478, February 2014.

[182] Mohammad Asghari and S Mohammad J MirzapourAl-e hashem. New advances in vehicle routing problems: A literature review to explore the future. *Green Transportation and New Advances in Vehicle Routing Problems*, pages 1–42, 2020.

[183] F. Alkaabneh, A. Diabat, and H.O. Gao. Benders decomposition for the inventory vehicle routing problem with perishable products and environmental costs. *Computers and Operations Research*, 113, 2020.

[184] P. Karakostas, A. Sifaleras, and M.C. Georgiadis. Adaptive variable neighborhood search solution methods for the fleet size and mix pollution location-inventory-routing problem. *Expert Systems with Applications*, 153, 2020.

[185] Antonio Giallanza and Gabriella Li Puma. Fuzzy green vehicle routing problem for designing a three echelons supply chain. *Journal of Cleaner Production*, 259:120774, June 2020.

[186] Luciano Costa, Thibaut Lust, Raphael Kramer, and Anand Subramanian. A two-phase Pareto local search heuristic for the bi-objective pollution-routing problem. *Networks*, 72(3):311–336, 2018. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.21827.

[187] Juho Andelmin and Enrico Bartolini. An Exact Algorithm for the Green Vehicle Routing Problem. *Transportation Science*, 51(4):1288–1303, November 2017. Publisher: INFORMS.

[188] Chun Cheng, Peng Yang, Mingyao Qi, and Louis-Martin Rousseau. Modeling a green inventory routing problem with a heterogeneous fleet. *Transportation Research Part E: Logistics and Transportation Review*, 97:97–112, January 2017.

[189] J. Guo and C. Liu. Time-dependent vehicle routing of free pickup and delivery service in flight ticket sales companies based on carbon emissions. *Journal of Advanced Transportation*, 2017, 2017.

[190] M. Rahimi, A. Baboli, and Y. Rekik. Multi-objective inventory routing problem: A stochastic model to consider profit, service level and green criteria. *Transportation Research Part E: Logistics and Transportation Review*, 101:59–83, 2017.

[191] Hongqi Li, Junli Yuan, Tan Lv, and Xinyu Chang. The two-echelon time-constrained vehicle routing problem in linehaul-delivery systems considering carbon dioxide emissions. *Transportation Research Part D: Transport and Environment*, 49:231–245, December 2016.

[192] Alejandro Montoya, Christelle Guéret, Jorge E. Mendoza, and Juan G. Villegas. A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, 70:113–128, September 2016.

[193] R. Kramer, N. Maculan, A. Subramanian, and T. Vidal. A speed and departure time optimization algorithm for the pollution-routing problem. *European Journal of Operational Research*, 247(3):782–787, 2015.

[194] R. Kramer, A. Subramanian, T. Vidal, and L.D.A.F. Cabral. A matheuristic approach for the Pollution-Routing Problem. *European Journal of Operational Research*, 243(2):523–539, 2015.

[195] Ángel Felipe, M. Teresa Ortuño, Giovanni Righini, and Gregorio Tirado. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, 71:111–128, November 2014.

[196] Michael Schneider, Andreas Stenger, and Dominik Goeke. The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations. *Transportation Science*, March 2014. Publisher: INFORMS.

[197] Khodakaram Salimifard and Ramin Raeesi. A green routing problem: optimising CO2 emissions and costs from a bi-fuel vehicle fleet. *International Journal of Advanced Operations Management*, 6(1):27–57, January 2014. Publisher: Inderscience Publishers.

[198] Jiani Qian and Richard Eglese. Finding least fuel emission paths in a network with time-varying speeds. *Networks*, 63(1):96–106, 2014. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.21524.