Route Optimisation for Autonomous Agricultural Spraying Robots

Qian Wan

A thesis submitted for the degree of

Doctor of Philosophy

at Monash University in 2025

School of Mathematics
Monash University
Melbourne, Australia
2025

Copyright notice

Acronyms

ARP Arc Routing Problem. 11, 15

CARP Capacitated Arc Routing Problem. i, ii, 8–10, 13, 15–19, 31, 35–39, 44, 77, 82, 83, 86, 98, 99, 101

CPP Chinese Postman Problem. 11, 15

CVRP Capacitated Vehicle Routing Problem. 14–16

MILP Mixed-Integer Linear Programming. i, 9, 13, 17, 19, 77, 83, 98, 102

OSRP Orchard Spraying Routing Problem. i, ii, 6–10, 13, 17, 19, 20, 22–24, 30, 37, 39, 40, 44, 45, 64, 66, 67, 69, 73, 77, 82–85, 91, 97–100, 102

OSRP$_{RG}$ Orchard Spraying Routing Problem on Regular Grid Graphs. 3, 10, 39, 44–49, 51, 52, 54, 55, 57, 60, 62–66, 69–72, 74, 83, 99

PA Precision Agriculture. 5, 12, 98

RPP Rural Postman Problem. 11, 15, 17, 19

SDCARP Split Delivery Capacitated Arc Routing Problem. i, ii, 8–10, 13, 16–20, 31, 35–39, 44, 77, 82, 89, 98–102

SDVRP Split Delivery Vehicle Routing Problem. 15

TOP Team Orienteering Problem. 11

TSP Traveling Salesman Problem. 11, 14

UAVs Unmanned Aerial Vehicles. 6, 12–14

UGVs Unmanned Ground Vehicles. i, 6, 12, 13, 17

VRP Vehicle Routing Problem. 11, 13–15

Abstract

In agriculture, spraying applications are performed multiple times each year. Efficient path planning is critical to enhancing the effectiveness of these operations. Despite considerable research efforts, high-quality path-planning solutions for practical, real-world applications remain limited. This study is motivated by a real-world problem in which orchard owners requested a routing plan for a fleet of Unmanned Ground Vehicles (UGVs)s. In this thesis, the orchards are modeled as undirected graphs, with the robotic sprayers starting from a depot and servicing edges with positive demand. When a robot depletes its spray load, it returns to the depot for refilling.

In agricultural spraying applications, some existing studies have modelled the problem as a Capacitated Arc Routing Problem (CARP). However, servicing an edge with a single vehicle may be impractical, allowing split deliveries could broaden the range of potential solutions. Therefore, the Split Delivery Capacitated Arc Routing Problem (SDCARP) is more suitable for spraying applications in most cases. Both the CARP and SDCARP are challenging NP-hard optimisation problems, with no highly effective methods available yet. Moreover, few studies specifically address the unique structure of orchard spray routing problems. Some research also overlooks the actual service demands of each edge, a critical factor in split delivery scenarios. Due to the complexity of the problem, most heuristic methods proposed for solving CARP and SDCARP are either overly complex, exhibit weak performance on large-scale problems, or lack adaptability to diverse scenarios.

To address these challenges, this study formulates the Orchard Spraying Routing Problem (OSRP) as a Mixed-Integer Linear Programming (MILP) within the framework of SDCARP. Specifically, this thesis introduces two OSRP models: one for basic needs and another for scenarios with large edge demands. The SDCARP is a variation of the CARP that allows multiple vehicles to service a single demand edge. The solution to CARP can be considered an upper bound for SDCARP.

This research leverages the orchards' grid-like structure to introduce a polynomial-time solvable CARP on regular grid graphs. The regular grid graphs are classified into two categories: single-layer and multi-layer. This study obtains optimal solutions and generates optimal path patterns for both categories of grid graphs by utilising mathematical insights from the OSRP on regular grid graphs. Furthermore, several theorems are presented to illustrate the characteristics of these solution patterns under different grid graph conditions. Counter-examples are also provided to demonstrate the theoretical insights across different scenarios.This research builds on the solution properties of regular grid

graphs and introduces small and large regular grid graph approximation methods to solve the single-block OSRP. The key idea is to approximate irregular grid graphs as regular ones and then apply the solution patterns designed for regular grid graphs, which provide lower and upper bounds respectively. Additionally, it proposes an enumeration-based set coverage algorithm to address the multi-block complex OSRP. Furthermore, both exact and heuristic methods are applied. The exact algorithms incorporate lazy constraints, enhanced lower bound constraints, and symmetry elimination constraints, while a heuristic repair technique is used to solve the OSRP.

This thesis evaluates algorithms using various datasets, including classical CARP datasets, generated datasets based on real-world properties, and actual orchard blocks. The numerical results demonstrate that the exact algorithms are efficient for small-scale problems, while heuristic repair algorithms perform better for medium-sized problems, providing improved upper bounds compared to classical exact algorithms. The small and large approximation methods demonstrate strong performance in handling large and complex single-block instances, while the enumeration-based set coverage algorithm efficiently solves complex, irregular multi-block instances. Additionally, several theorems on regular grid graphs are presented, which can be integrated with other methods in various agricultural applications.

This study establishes a connection between agricultural spraying applications and the SDCARP and CARP models, presenting novel approaches and incorporating mathematical insights that offer a new perspective in the agricultural domain. Furthermore, we provide a complete version of the OSRP model, which includes detailed information such as exact routing plans for vehicles and the amount of delivery on each demand edge. Additionally, the small and large regular grid graph approximation methods, as well as the enumeration-based algorithm, are straightforward and scalable, enabling efficient solutions for large-scale real-world instances. The combined use of exact algorithms and heuristic repair techniques offers practical solutions for agricultural routing applications. To address the lack of test datasets in agricultural spraying applications, this research introduces three real-world orchard graph instances and generates multiple datasets with realistic characteristics. These instances broaden the research scope in agricultural spraying and provide robust models for researchers to test, validate, and enhance existing benchmark datasets for agricultural applications, as well as for evaluating CARP and SDCARP. Overall, this thesis lays the groundwork for more effective and efficient agricultural spraying strategies, bridging theoretical research with practical applications and creating a foundation for future studies in this critical field.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Qian Wan
28 February 2025

Publications during enrolment

- **Chapter 3** is based on joint work with Andreas Ernst, Rodolfo García-Flores, Simon Bowly, and Philip Kilby. The manuscript, titled *"An Orchard Spraying Application of the Split-Delivery Capacitated Arc Routing Problem"*, has been submitted to *Optimization Letters* and is also available on *arXiv* under the same title.

- **Chapter 4** is based on joint work with Andreas Ernst, Rodolfo García-Flores, Simon Bowly, and Philip Kilby. The manuscript, titled *"A Polynomial Time Solvable Capacitated Arc Routing Problem on Regular Grid Graphs"*, has been published in *Networks*.

- **Chapter 5** is based on joint work with Andreas Ernst, Rodolfo García-Flores, Simon Bowly, Philip Kilby, and Andreas Schutt. The manuscript, titled *"Routing Optimization for Automated Spraying Robots in Real-World Agricultural Operations"*, has been submitted to *Computers and Electronics in Agriculture*.

All datasets used in this thesis are available at:
https://github.com/QianWan92/Code/tree/main/All_Data

A PhD is not just the pursuit of answers but the realisation that every answer births new questions. Perhaps that is why it is called a Doctor of Philosophy—because true wisdom lies in knowing that the search for knowledge never truly ends.

Acknowledgements

The completion of this PhD has been a journey of challenges, learning, and growth, made possible by the support and guidance of many individuals.

I would like to express my sincere gratitude to my main supervisor, Professor Andreas Ernst, for his exceptional mentorship. His invaluable guidance has not only shaped my research but has also helped me develop essential skills and effective work strategies. His passion for research and dedication to excellence have been truly inspiring, leaving a lasting impact on my academic and personal journey.

I sincerely thank my co-supervisors, Dr. Rodolfo García-Flores, Dr. Simon Bowly, and Dr. Philip Kilby, for their patience, expertise, and unwavering support. Their insights have been instrumental in refining my ideas and strengthening this work.

I am deeply grateful to the members of my review panel for their invaluable feedback and constructive suggestions, which have significantly contributed to improving the quality and clarity of this research.

I also extend my appreciation to my colleagues for the stimulating discussions, shared coffee breaks, and camaraderie, all of which have made this journey more enjoyable.

To my family, thank you for your unconditional love and encouragement. To my parents and younger brother, your unwavering belief in me has been my greatest motivation.

I am thankful to CSIRO Data61 and Monash University for providing the scholarship that allowed me to fully dedicate myself to my research.

Finally, to everyone who has contributed to this journey, whether through academic discussions, moral support, or moments of respite, thank you. This thesis stands as a testament to the collective support and guidance I have received along the way.

Contents

List of Symbols

The following list describes symbols used in this thesis.

(i, H_j) The linking edge between nodes $N(i, j)$ and $N(i, j+1)$, where $i \in \{1, 2, \ldots, K\}$ and $H_j = j$, $j \in \{1, 2, \ldots, M-1\}$. The group of linking edges refers to the set of all linking edges that share the same index j.

(L_i, j) The demand edge between $N(i, j)$ and $N(i+1, j)$, where $i \in \{1, 2, \cdots, K-1\}$ and $j \in \{1, 2, \cdots, M\}$.

$\delta \in [0, 1]$ The fraction of demand edges to be included.

$\delta^+(k)$ Set of outgoing arcs of node k, defined as $\delta^+(k) = \{j \in \mathcal{N} : (k, j) \in \mathcal{A}\}$.

$\delta^-(k)$ Set of incoming arcs of node k, defined as $\delta^-(k) = \{i \in \mathcal{N} : (i, k) \in \mathcal{A}\}$.

Γ The number of times the heuristic will run without improvement before abandoning further attempts to execute the repair heuristic.

$\mathbf{x}$ Set of solutions for x variables, defined as $\mathbf{x} = \{x_{ij}^r : (i, j) \in \mathcal{A}, r \in \mathcal{R}\}$.

$\mathbf{y}$ Set of solutions for y variables, defined as $\mathbf{y} = \{y_{ij}^r : (i, j) \in \mathcal{E}, r \in \mathcal{R}\}$.

$\mathcal{A}$ Set of all existing directed arcs in a graph, denoted as $\mathcal{A} \subset \{(i, j) : i, j \in \mathcal{N},\ i \neq j\}$.

$\mathcal{D}$ The set of all demand edges with positive demand, where $i < j$.

$\mathcal{E}$ Set of undirected edges, defined as $\mathcal{E} = \{(i, j) \in \mathcal{A} : i < j\}$. $|\mathcal{E}|$ represents the number of edges.

$\mathcal{E}^+$ Set of demand edges with positive demand, where $\mathcal{E}^+ \subset \mathcal{E}$. $|\mathcal{E}^+|$ represents the number of demand edges.

$\mathcal{F}(\mathcal{S})$ Subset of nodes that includes at least one of the end nodes of demand edges in $\mathcal{S}$.

$\mathcal{G}$ An undirected connected graph, defined as $\mathcal{G} = (\mathcal{N}, \mathcal{E})$.

$\mathcal{N}$ Set of all nodes. $|\mathcal{N}|$ represents the number of nodes.

$\mathcal{P}$ The set of all enumerated possible demand edge combinations considered as candidate paths.

$\mathcal{RG}$ An undirected $K \times M$ regular grid graph, denoted by $\mathcal{RG} = \text{Reg}(K, M)$.

$\mathcal{R}$ Set of indices for the robotic sprayers' tours, defined as $\mathcal{R} = \{1, 2, \ldots, R\}$. For simplicity, robotic sprayers are referred to as robots. R denotes the minimum number of robots required to spray the orchard, typically calculated as $|\mathcal{R}| = R = \left\lceil \frac{\sum_{(i,j)\in\mathcal{E}} D_{ij}}{P} \right\rceil$. (For a regular grid graph, $|\mathcal{R}| = R = \lceil \frac{ML}{P} \rceil$).

$\mathcal{R}_{new}$ Set of indices for the robotic sprayer's tours that spray more than one demand edge in the large edge demand model, defined as $\mathcal{R}_{new} = \{1, 2, \ldots, \min(|\mathcal{E}^+| - 1, |\mathcal{R}|)\}$.

$\mathcal{S}$ Subset of nodes that includes the depot, where $\mathcal{S} \subset \mathcal{N}$.

$\mathcal{T}$ Subset of nodes, defined as $\mathcal{T} = \mathcal{N} \setminus \mathcal{S}$.

A_r The cumulative cost of linking edges starting from the depot, where $A_r = \sum_{i=1}^{f_r - 1} C_i$, and $f_r = \sum_{j=1}^{r} m_j$ when the solution is adjacent.

C_r^{robot} The cost incurred by robot r, which is equal to $2\sum_{i=1}^{f_r-1} C_i + \left\lceil \frac{m_r}{2} \right\rceil \times 2C_d$.

C_p The cost associated with path p, including the travel cost from the depot to the start of the path and back to the depot.

$C_{(i,H_j)}$ The cost of linking edge (i, H_j). In regular grid graphs, the cost of each linking edge in every group is identical. Therefore, $C_{(i,H_j)} = C_j$ for regular grid graphs.

$C_{(L_i,j)}$ The cost of demand edge (L_i, j). In regular grid graphs, the cost of each demand edge is the same. To simplify notation, we represent the cost of any demand edge (L_i, j) as C_d. Therefore, $C_{(L_i,j)} = C_d$ for regular grid graphs.

C_{ij} Non-negative cost of arc $(i, j) \in \mathcal{A}$, where $C_{ij} = C_{ji}$.

C_{total} The overall cost of an $OSRP_{\mathcal{RG}}$ solution, defined as $C_{total} = \sum_{r\in\mathcal{R}} C_r^{robot}$.

$D_{(L_i,j)}$ The demand of demand edge (L_i, j). In regular grid graphs, the demand of each demand edge is the same. To simplify notation, we represent the demand of any demand edge (L_i, j) as one unit. Therefore, $D_{(L_i,j)} = 1$ for regular grid graphs.

D_{ij} Non-negative spray demand for the arc $(i, j) \in \mathcal{E}$.

E_p The set of undirected edges visited at least once in path p.

f_r The index of the demand edge that is furthest away from the depot, which is sprayed by robot r.

f_r^l The index of the demand edge that is furthest from the depot at the l-th layer, which is sprayed by robot r. Defined as $f_r^l = \max I_r^l$, and $f_r = \max_{l\in\{1,2,\ldots,L\}} f_r^l$.

I_r^l The subset of $\{1, 2, \ldots, M\}$ denoting the indices of the demand edges that the r-th robot sprays at the l-th layer.

L The total number of layers in a regular grid graph, defined as $L = K - 1$. Each layer is indexed by i, where $i \in \{1, 2, \ldots, K-1\}$. Specifically, L_i represents the i-th layer, given by $L_i = i$. When $L = 1$, it is referred to as a regular single-layer graph. When $\mathcal{RG}$ has more than one layer, i.e., $L \geq 2$, it is referred to as a regular multi-layer graph

m_r The number of demand edges sprayed by robot r.

$N(i, j)$ Nodes on a grid, identifiable by vertical and horizontal coordinates $i \in \{1, 2, \cdots, K\}$, $j \in \{1, 2, \cdots, M\}$.

n_r The index of the demand edge that is nearest to the depot, which is sprayed by robot r.

n_r^l The index of the demand edge that is nearest to the depot at the l-th layer, which is sprayed by robot r. Defined as $n_r^l = \min I_r^l$, and $n_r = \min_{l\in\{1,2,\ldots,L\}} n_r^l$.

P The capacity of the robotic sprayer. In regular grid graphs, this capacity is defined as the maximum number of demand edges that each robot can spray. In this study, P is an integer.

S The overall solution for the OSRP$_{RG}$, defined as $S = (S_1, S_2, \ldots, S_{\mathcal{R}})$.

S_r The set of indices of demand edges serviced by robot r for the OSRP$_{RG}$, defined as $S_r = \{n_r, \ldots, f_r\}$.

SP_{ij} The shortest distance (or cost) from node i to node j ($\forall\, i, j \in \mathcal{N}$) calculated using Dijkstra's Algorithm.

$SPath_{ij}$ The shortest path from node i to node j ($\forall\, i, j \in \mathcal{N}$) calculated using Dijkstra's Algorithm.

T The least common multiple of P and $2L$.

t_{ij} Non-negative integers representing the total number of times that robots have passed edge (i, j), where $(i, j) \in \mathcal{A}$. Defined as $t_{ij} = \sum_{r=1}^{|\mathcal{R}|} x_{ij}^r$.

x_{ij}^r Binary variable, equal to 1 if the arc $(i, j) \in \mathcal{A}$ is traversed by $r \in \mathcal{R}$, and 0 otherwise.

y_{ij}^r Amount of chemicals sprayed on edge $(i, j) \in \mathcal{E}$ by robot $r \in \mathcal{R}$, where $y_{ij}^r = 0$ when $D_{ij} = 0$.

z_p Binary decision variable indicating whether path p is selected (1 if selected, 0 otherwise).

z_{ij} The number of single-edge robots (i.e., robots that spray only a single edge) assigned to edge $(i, j) \in \mathcal{E}$. If no such robot is assigned, then $z_{ij} = 0$, where $z_{ij} \in \mathbb{Z}_+$.

Chapter 1

Introduction

1.1 Motivation

Precision Agriculture (PA), also known as precision farming or smart farming, is a farm management approach focused on reducing costs, boosting yield quantity and quality, and minimising environmental impact [67]. Its primary goal is to achieve low-input, high-efficiency, and sustainable agricultural production. PA has become essential to the global economy in meeting the rising demands of a growing population [54]. Automated robots and machinery are integral to PA, significantly improving efficiency and addressing labour shortages, thereby offering substantial advantages to agricultural areas [52]. PA has facilitated the widespread adoption of automation and robotics in agricultural activities such as planting [17, 35, 70], inspection [51, 61, 71], spraying [30, 59, 76], and harvesting [8, 79, 81].

Spraying applications are vital tasks in agriculture, performed multiple times per year. Recently, autonomous robotic sprayers have been developed to apply pesticides, herbicides, and fungicides, offering a safer and more sustainable approach to spraying in agricultural fields. As agricultural areas grow, the time available for spraying remains unchanged, making it increasingly challenging for human workers to complete spraying tasks in larger fields within a limited time frame. As a result, robotic sprayers are becoming the preferred technology for performing these operations efficiently and safely.

An efficient routing plan plays a critical role in the agricultural spraying activities. Optimising the paths of robotic sprayers allows farms to reduce operational time and costs, ultimately increasing profitability [82]. Effective route optimisation also minimises energy consumption and environmental impact by reducing fuel usage and emissions, especially for larger unmanned robotic sprayers. Despite these advantages, there is still a lack of high-quality, accessible path planning solutions in the market that meet the

specific demands of modern precision agriculture [57].

This research is motivated by a practical challenge faced by orchard owners who sought a *routing plan* for a fleet of automated Unmanned Ground Vehicles (UGVs). These owners require a routing plan that coordinates the movement of robotic sprayers across expansive orchards to optimise distance coverage. In this scenario, when a robot's spray load is exhausted, it can return to a designated depot to refill its spray tank, recharge its battery, or refuel. In this study, the vehicles operate continuously, eliminating the need for refuelling stops. The primary focus is on developing a routing plan to minimise the total travel distance of a fleet of identical vehicles while spraying agrochemicals on orchard demand edges, subject to demand and capacity constraints.

1.2 The Orchard Spraying Routing Problem (OSRP)

The present research addresses a practical Orchard Spraying Routing Problem (OSRP). Orchards come in various shapes, often featuring multiple rows of differing lengths, non-spray areas, and occasionally hard-to-access zones. In this thesis, an orchard is represented as an undirected graph $\mathcal{G}$, where tractor tracks are depicted as *edges*. Those edges requiring spraying are termed *demand edges* or *required edges*, while edges without demand are *non-demand edges*. Edges within the orchard are classified as headland edges, interior edges, and island edges [65]. In practice, only the interior edges (illustrated as dashed lines in Fig. 1.1) require spraying. Any intersection between two edges on the graph is represented as a *node*.

In many orchards, barriers such as planted trees and islands restrict vehicle passage and are referred to as *obstacle areas*. Fig. 1.1 provides an example of a real-world orchard graph based on Plessen [65], illustrating two primary types of orchard graphs: uninterrupted (without obstacles) and interrupted graphs. In the graph, dots represent nodes, while solid and dashed lines indicate non-demand edges (headland and island edges) and demand edges (interior edges), respectively.

Various vehicle types are used in agricultural spraying, including Unmanned Aerial Vehicles (UAVs) and UGVs, as well as cooperative systems involving both. These systems are employed under different requirements and agricultural settings. In brief, UAVs offer flexibility and precision in navigating complex terrains, while UGVs provide greater capacity and endurance, making them particularly suitable for structured or flat agricultural environments. These characteristics align well with the needs of orchard spraying, which is the focus of this study. Specifically, this research concentrates on UGVs, which are especially effective for servicing demand edges in large agricultural areas due to their higher payload capacity and operational endurance.

In the OSRP, each robot performs spraying tasks according to its capacity and the orchard's edge demands, ensuring that every demand edge is sprayed by at least one robot. When robots exhaust their supply of agrochemicals, they return to the depot for refilling. The refilling point is typically located at the depot or managed by mobile nursing carts that supply them with agrochemicals for spraying when needed. This research is motivated by a practical consulting issue where the vehicles could operate continuously, rendering pauses for refuelling unnecessary. Thus, unlike cases of energy-constrained routing problems [1], we emphasise the significance of spray refilling at the depot and aim to find an optimal routing plan for our problem. In all of the problems discussed in this thesis, a *fixed depot* is positioned at one of the corners of the orchard graph unless otherwise stated.

The OSRP can be viewed as either a multi-edge robot or a single-edge robot spraying routing problem. Specifically, treating each trip as an individual task for a single robot frames it as a single-edge robot problem. Conversely, viewing each trip as a separate task for multiple robots defines it as a multi-edge robot problem. Mathematically, the multi-edge robot and single-edge robot spraying routing problems are equivalent, as this thesis does not involve scheduling robots. Each robot's trip is treated as an individual task, and indices from 1 to R are used to refer to different trips (robots) throughout the thesis.

In summary, the properties of the OSRP are:

(1) The orchards may have arbitrary shapes, with uneven costs, internal obstacles, and other irregularities.

(2) A transition graph with edges connecting nodes.

(3) The cost array is non-negative. The costs of spraying an edge is equal to its path length.

(4) Edge demands are non-negative and edge demand should be non-zero for those edges that must be sprayed.

(5) The robotic sprayers have identical capacities, denoted by a fixed capacity P. The total demand serviced by each robot must not exceed P.

(6) There is a designated start node, known as the depot, that is also an end node.

(7) The depot also acts as a refill point such that, if the robot runs out of agricultural chemicals to spray, it goes back to the depot to refill.

(8) The demand for one edge may be satisfied by more than one robot (split demand).

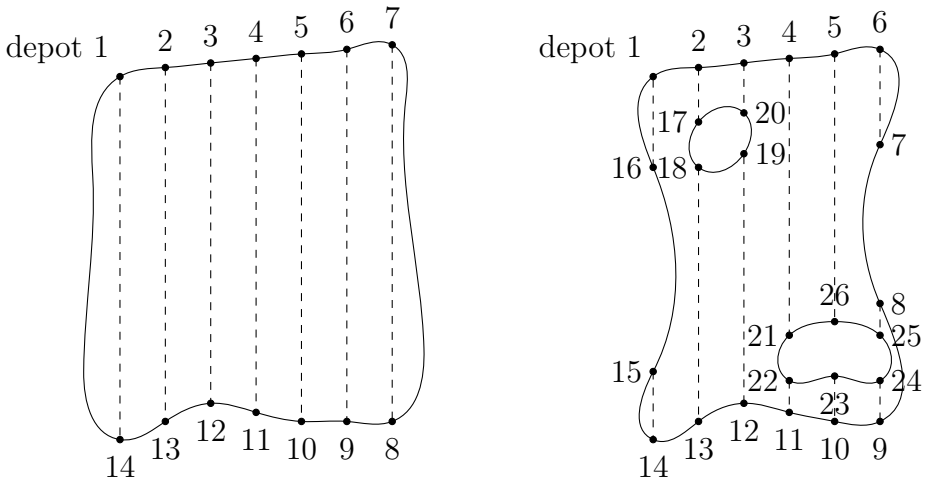


Figure 1.1: An example of an orchard layout based on real-world attributes [65]. Solid and dashed lines represent non-demand edges (headland and island edges) and demand edges (interior edges), respectively, with dots indicating nodes. In the left panel, uninterrupted edges are displayed, while the right panel illustrates interrupted edges caused by obstacle areas that prevent robots from passing through. Node 1 is designated as the *depot*, from which robots depart to cover interior edges (dashed lines). If a robot depletes its agrochemical supply, it returns to the depot to refill.

The objective of the OSRP is to minimise the total cost of servicing all demand edges in an orchard while satisfying capacity and demand constraints. The OSRP can be viewed as an application the Split Delivery Capacitated Arc Routing Problem (SDCARP), which itself is a variation of the Capacitated Arc Routing Problem (CARP).

The CARP is a well-known combinatorial optimisation problem, first introduced by Golden and Wong [29]. The CARP aims to develop the least costly routing plan for a fleet of identical vehicles that must service specific network edges (such as roads or paths) while considering vehicle capacity constraints. The CARP is particularly useful in agricultural spraying applications, where the goal is to minimise the non-spray distance travelled to achieve the lowest possible travel costs. The CARP is defined by several key constraints: all vehicles start from the same depot, service a subset of edges, and return to the depot once their capacity is depleted. Each vehicle's trip is constrained by its capacity, and every demand edge must be serviced exactly once, ensuring that there is no overlap in servicing. The SDCARP, a variation of the CARP, allows multiple vehicles to service each demand edge. This variation is crucial in real-world scenarios where it may be impractical to service an edge with a single vehicle, thus expanding the range of possible solutions. The OSRP focuses on spraying applications in agricultural areas, incorporating real-world constraints while aligning with the core structure of the SDCARP.

This thesis focuses on developing solutions for the OSRP, a class of SDCARP instances

derived from orchard spraying applications. The study formulates the OSRP as a MILP and introduces two mathematical models within the SDCARP framework: one for basic requirements and another designed for scenarios with large edge demands. The SDCARP is an NP-hard optimisation problem, which presents challenges in finding optimal solutions within a reasonable time frame. To address these challenges, this work employs a combination of exact algorithms—integrating lazy constraints, enhanced lower bound constraints, and symmetry elimination constraints—alongside a heuristic repair technique. Experiments on classical CARP instances and generated orchard data validate the effectiveness of the proposed methods in producing reasonable solutions. However, as data size increases, a more straightforward and scalable approach becomes necessary to address the growing complexity of the OSRP.

This study leverages the specific grid-like structure of orchards, which differs from the network structures typically seen in other CARP and SDCARP instances. It introduces a polynomial-time solvable CARP for the specific structures named as regular grid graphs. These regular grid graphs are categorised into two types for different scenarios: single-layer and multi-layer. This thesis utilises mathematical insights specific to the OSRP on regular grid graphs to derive optimal solutions and corresponding path patterns for both categories. Several theorems are presented to describe the characteristics of these optimal solution patterns under varying grid graph conditions, complemented by counterexamples to illustrate the theoretical insights across diverse scenarios. The study builds on the solution properties of regular grid graphs and introduces two methods: the small regular grid graph approximation and the large regular grid graph approximation, to address the OSRP. The key idea is to approximate irregular grid graphs as regular ones and then apply the solution patterns developed for regular grid graphs, which provide lower and upper bounds, respectively. Additionally, an enumeration-based set coverage approach is proposed to solve the OSRP on complex multi-block instances. Numerical results on real-world data demonstrate that the approximation and enumeration-based methods efficiently provide high-quality solutions for the OSRP.

The key contributions of this study are: First, this study establishes a connection between agricultural spraying applications and the SDCARP and CARP models, presenting the OSRP mathematical model. This model includes detailed information, such as exact routing plans for vehicles and the amount of delivery on each demand edge. Additionally, mathematical theorems and lemmas are developed and proven for the OSRP in various scenarios, offering a fresh perspective in the agricultural domain and providing valuable mathematical insights into the OSRP that can be utilised in different methods and applications. Furthermore, the proposed regular grid graph approximation methods are straightforward and scalable, enabling efficient solutions for large-scale, real-world in-

stances. This thesis evaluates algorithms using diverse datasets, including classical CARP datasets, generated datasets based on real-world properties, and data from three actual orchard blocks. This collection of datasets address the lack of test datasets in agricultural spraying applications, broadening the research scope and providing robust models for researchers to test, validate, and enhance existing benchmark datasets. These models can also be used for evaluating CARP and SDCARP instances. Overall, this thesis lays the groundwork for more effective and efficient agricultural spraying strategies, bridging theoretical research with practical applications and creating a solid foundation for future studies in this critical field.

This thesis is organised as follows: Chapter 2 provides an overview of the background and related research on the agricultural spraying routing problem. Chapter 3 introduces the mathematical models for the OSRP and proposes both exact and heuristic solution approaches. Chapter 4 presents a polynomial-time solvable CARP, explains the concept of regular grid graphs, and includes mathematical proofs, theorems, and lemmas for the Orchard Spraying Routing Problem on Regular Grid Graphs (OSRP$_{RG}$). Chapter 5 describes the small and large approximation methods and evaluates their performance on real-world orchard blocks. Chapter 6 explores potential approaches for solving complex multi-block areas and discusses strategies for further improving the proposed solutions. Finally, Chapter 7 concludes the thesis and outlines directions for future research.

Chapter 2

Literature Review

The mathematical complexity and practical relevance of routing problems have attracted the attention of many researchers, especially in the fields of supply chain management, transportation, and logistics. The present study is motivated by a real-world problem in which orchard owners requested a routing plan for a fleet of automated robotic sprayers that is suitable to the specific features of the Australian landscape, such as demand edges in close proximity and numerous obstacles. In agriculture, path planning problems can be broadly divided into two main categories: *node coverage* and *edge coverage*. *Node coverage* focuses on demands at specific locations, or nodes, within a network. A classic example of this is the Traveling Salesman Problem (TSP)[41], where the objective is to efficiently visit each node exactly once. Another example is the Team Orienteering Problem (TOP)[62], where a fixed fleet of vehicles is employed to obtain rewards by visiting selected nodes in a network. In contrast, *edge coverage* addresses demands along the connections, or edges, between nodes. A well-known example is the Chinese Postman Problem (CPP)[25, 33], which aims to cover each edge in the network efficiently, and another example is the Rural Postman Problem (RPP)[26], which seeks to cover only a subset of edges in the most efficient way. Some solution methods for node coverage and edge coverage can be adapted and combined to develop effective solutions for various scenarios, depending on the specific requirements of the applications.

In this chapter, an exploration of the research on the agricultural spraying routing problem is presented in Section 2.1, node coverage is discussed as the Vehicle Routing Problem (VRP) in Section 2.2, and edge coverage is examined as the Arc Routing Problem (ARP) in Section 2.3.

2.1 Routing Plans for Agricultural Spraying Applications

In this section, a discussion of the different types of vehicles used in agriculture is provided in Section 2.1.1, followed by an overview of the path planning algorithm for spraying applications in Section 2.1.2.

2.1.1 Agricultural Spraying Vehicles

In recent decades, the expansion of orchards has become necessary to meet the growing demand for agricultural products driven by population growth. Due to labour shortages and rising labour costs, research on automated robots has emerged as the second most popular topic in PA [53]. Furthermore, certain agrochemicals, such as pesticides, pose risks to human health. Thus, developing effective routing plans for robots in agricultural fields is essential to advancing sustainable and efficient farming practices.

In agricultural spraying applications, various vehicles, including UAVs and UGVs, are employed to improve efficiency and precision. UAVs, widely used in orchard pest control [45], operate by flying over crops and applying sprays from above, making them particularly effective for reaching the upper layers of tall crops, such as fruit trees [83]. In contrast, UGVs work on the ground, applying sprays directly to plants and soil. This enables more precise targeting of the lower parts of plants and improved access under dense canopies where aerial applications may be less effective [55].

UAVs are generally faster, covering large areas quickly due to their aerial operation, though they carry smaller spray volumes and require more frequent refills, which can impact overall efficiency. On the other hand, UGVs, though slower due to navigation challenges on the ground, typically have larger spray tanks that allow for longer, uninterrupted operation. UAVs are particularly suited to rough, steep, or irregular terrain, offering greater versatility across varied landscapes and independence from the orchard's structural layout. In contrast, UGVs perform best on flat or gently sloping terrain and in structured crop layouts like row crops, though they may face challenges in irregular or uneven environments.

In terms of resource use, UAVs tend to use less water and pesticides, owing to their controlled and precise application, which reduces waste and environmental impact. UGVs, while often using larger volumes, can deliver sprays directly to the plant base, reducing unintended drift. Despite their benefits, UAVs are limited by payload and battery life, making them less suited to large fields or dense orchards requiring heavy applications. UGVs, while terrain-dependent and less manoeuvrable in complex layouts, are advanta-

geous for large fields where greater volumes of pesticides or fertilisers are necessary.

In summary, UAVs provide flexibility and precision for complex landscapes, while UGVs are advantageous for sustained operations with larger capacities, especially in structured or flat agricultural settings, aligning well with our OSRP requirements in the orchard.

2.1.2 Path Planning for Spraying Vehicles

Xu et al. [83] introduced an orchard pesticide spraying routing problem involving coordinated multiple vehicles, where drones are used for pesticide spraying while ground vehicles support them by replenishing their power and supplies in a timely manner. The problem is formulated as an integer programming model, and an enhanced Simulated Annealing-Lin Kernighan Helsgaun Algorithm is proposed to address the NP-hard problem. Their approach demonstrates superior performance compared to the general Variable Neighbourhood Search Algorithm [72], the Improved Ant Colony Algorithm [48], and the Improved Genetic Algorithm [16]. Additionally, Gan et al. [27] presented a Dynamic Parameter-A* (DP-A*) algorithm, an enhancement of the traditional A* algorithm, which enabled UGVs to continuously refine their path while repeatedly performing the same task. DP-A* outperformed the standard A* algorithm across various evaluation criteria, including convergence speed, memory efficiency, path optimisation, and dynamic learning capability.

Most spraying studies present a CARP model tailored to specific scenarios, often excluding the consideration of split deliveries. However, split delivery is essential in real-world situations where servicing an edge with a single vehicle may be impractical, allowing split deliveries could broaden the range of potential solutions. In the present study, we formulate the OSRP as a MILP within the framework of SDCARP. The general SDCARP is NP-hard, making it challenging to obtain optimal solutions. To tackle this complexity, we employ both exact and heuristic approaches. Moreover, the orchard layout typically resembles a grid graph composed of blocks and rows. Leveraging this grid-like structure, we propose simple and scalable methods that exploit the properties of regular grid graphs, enabling the problem to be solved in polynomial time.

Section 2.2 discusses research on the VRP, which serves as a solid starting point, as some orchard spraying problems, such as those involving fruit trees, can be viewed as a VRP.

2.2 Vertex Routing Problem (VRP)

The general VRPs pose significant computational challenges and are classified as NP-hard, as demonstrated by Lenstra and Kan [43]. There are many variations [20, 39, 50, 74]. A well-known variant is the TSP, as described by Grötschel et al. [32], where only one vehicle is available at the depot and no additional operational constraints are imposed [66]. Another variation is the Capacitated Vehicle Routing Problem (CVRP), which seeks to find routes with minimal cost for multiple identical vehicles with limited capacity, where the demand of multiple *nodes* (i.e., customers) must be serviced exactly once by exactly one vehicle [13, 39]. Dror and Trudeau [23] introduced a relaxed version of VRP, allowing deliveries to a demand point to be split among any number of vehicles, highlighting the potential cost savings achievable through such split deliveries.

Utamima and Djunaidy [75] review the literature on routing problems, offering extensive insights into the application of VRP for agricultural machinery. Their study highlights current challenges and identifies research gaps in farm field logistics. Specific applications in field logistics include coverage path planning, such as aerial coverage optimisation [77] and crop harvesting [69].

Barrientos et al. [7] focused on deploying small aerial robots with limited capacity, offering practical solutions for applications such as multi-edge robot task planning, path planning, and UAVs control. Their methodology consists of two main steps: first, dividing the fields into several sub-areas, each assigned to specific robots; and second, solving a coverage routing planning problem based on a wavefront planner [40]. This approach has been applied successfully across various fields. Valente et al. [77] introduced a meta-heuristic algorithm, Harmony Search (HS), to tackle the coverage path planning problem for vineyard parcels. While this approach required more computational time than the method proposed by Barrientos et al. [7], it achieved better route optimisation results.

Sethanan and Neungmatcha [69] presented a multi-objective approach to simultaneously minimise distance and maximise total sugarcane yield for mechanical harvester route planning. Sethanan and Neungmatcha [69] utilised a variant of Particle Swarm optimisation (PSO) to address this problem. A novel particle encoding/decoding scheme was developed to integrate path planning with accessibility and split harvesting constraints, demonstrating improved performance over traditional particle swarm optimisation methods. Moreover, Gracia et al. [31] addressed a biomass collection problem modelled as a VRP. Volte et al. [78] presented examples of the Differential Harvest Problem, which is an extension of VRPs with specific constraints that make it harder to solve. Their problem stems from harvesting grapes of different quality, and is solved in a reasonable time using column generation in the general purpose software *VRPSolver*™. They applied a hybrid

approach combining genetic algorithms and local search methods to solve the real-world problem.

Dror and Trudeau [23] demonstrated that allowing split demands has the potential to significantly reduce both the total distance travelled and the number of vehicles required, resulting in notable cost savings. Archetti and Speranza [6] conducted a comprehensive survey of VRPs, specifically focusing on the Split Delivery Vehicle Routing Problem (SDVRP), and reviewed its key characteristics along with both exact and heuristic solution methods. Their work offers a broad overview of various SDVRP variants and the associated literature. While VRPs primarily emphasise "node coverage," ARPs are more relevant in scenarios where routing between specific edges is crucial.

2.3 Arc Routing Problem (ARP)

In the ARP, customers are not located at nodes but along the arcs of the road network, as described in [21]. The ARP has numerous real-life applications, such as garbage collection [3], postal delivery [25, 26], and snow removal [15], among other uses. ARPs can be categorised into two classes: the CPP and the Rural Postman Problem (RPP) [25, 26]. The CPP seeks to find the minimum route for a postman starting from a post office, delivering mail to all streets, and returning to the post office. This problem can be viewed as finding the route with the minimum travel distance that visits all edges in an undirected graph. Although certain cases of the CPP can be solved in polynomial time [24], there are various variants, such as the mixed CPP (where the graph has both undirected and directed edges), which are NP-hard [63]. The RPP seeks the shortest round trip that covers a specified subset of edges within a network [60]. Like many combinatorial problems, the RPP is NP-hard [42]. However, neither the CPP nor the RPP accounts for vehicle capacity or edge demands.

The CARP was formally introduced by Golden and Wong [29] and holds a similar role in arc routing as the CVRP does in node routing. CARP aims to devise the most cost-effective routing plan for a fleet of identical vehicles assigned to service demand edges within a network. This problem has been widely applied and extended in agricultural path planning scenarios. Several key constraints define the CARP: all vehicles begin at the same depot, handle the subset of demand edges, and return to the depot upon depleting their capacity. Each vehicle's trip is limited by its capacity. Additionally, each demand edge must be serviced precisely once, preventing multiple vehicles from servicing the same edge. The general CARP poses significant challenges in finding an optimal solution, as it is known to be an NP-hard problem. Golden and Wong [29] demonstrated that even achieving a 0.5-approximate solution for CARP is also NP-hard.

The primary exact algorithms for CARP rely on cutting plane and column generation techniques [9]. Additionally, some studies explore transformations of CARP into CVRP [46]. Transformation techniques have emphasised the relationship between node coverage and edge coverage problems. Recently, Pecin and Uchoa [64] introduced a branch-and-cut-and-price algorithm that combines various methods from the CARP literature with elements adapted from recent successful node routing algorithms. Due to the complexity of these problems, finding optimal solutions for large instances is computationally challenging, and heuristic algorithms are commonly used in practice to obtain solutions.

Various heuristic methods have been employed to address the CARP, such as tabu search [14] and memetic algorithms [73]. Chen et al. [19] introduced a hybrid metaheuristic approach that combines randomized tabu thresholding with an infeasible descent procedure to the memetic framework, yielding improved results for standard CARP instances. Wøhlk and Laporte [80] developed a fast heuristic called FastCARP, which partitions the graph into districts and constructs routes for each tour, making it suitable for solving large-scale CARP problems, such as waste collection problems. López-Santana et al. [47] propose a heuristic algorithm based on scatter search, simulated annealing and iterated local search to solve a variant of the CARP with refill points known as CARP-RP. CARP-RP assumes two types of vehicles: the service vehicles that traverse rows, and the refilling vehicles that can meet the service vehicle at any point in the graph for refilling. Amaya et al. [2] introduce a version of CARP-RP when the refill vehicle meets the service vehicle several times before returning to the depot, i.e., with multiple loads, or CARP-RP-ML. In Amaya et al. [2]'s problem, road markings have to be painted or repainted every year, and it was solved with an exact cutting plane method and a route first-cluster second heuristic procedure. Zhang et al. [85] propose a methodology to optimise a harvester path planning problem for quadrilateral fields, considering that the stubble crushing caused by the harvester during the first season of ratoon rice harvesting will directly affect the grain yield of the second ratoon season. The mechanised harvesting process in the first season is modelled as a CARP and solved using a genetic algorithm. Recently, Boyacı et al. [12] introduced a fast upper and lower bounds algorithm that employs heuristic methods and local search to construct a giant tour to obtain upper bounds, as well as a cutting-plane algorithm to obtain lower bounds, to solve a real-life Mixed CARP with Intermediate Facilities and a Deadline (MCARPIFD) with more than 10,000 roads.

The SDCARP [56] is a relaxed variation of the classical CARP. The main difference between CARP and SDCARP is that, in SDCARP, the demand for a single edge can be serviced by multiple vehicles. This flexibility is essential in real-world contexts where restricting service to a single vehicle per demand edge may be impractical and overly

limiting, thereby expanding the range of potential solutions. The general SDCARP is classified as NP-hard, as it reduces to the NP-hard RPP when assuming unlimited capacity for the robot.

In the study of the SDCARP, Mullaseril et al. [56] described a feed distribution split-delivery capacitated rural postman problem and introduced an SDCARP incorporating time windows on arcs. Initially, their research aimed to develop effective solutions for the corresponding routing problem where split deliveries were not allowed. Later, they applied heuristic techniques to generate and improve split-delivery solutions, achieving noteworthy enhancements in efficiency. This progress is elaborated in Dror et al. [22]. Furthermore, Muyldermans and Pang [58] discussed the SDCARP as a variant of CARP, highlighting its relevance and the associated research. Muyldermans and Pang [58] noted that in many practical situations, it is often impractical to service a demand edge using a single vehicle. Belenguer et al. [10] introduced an aggregated formulation and incorporated valid inequalities, implementing a cutting plane algorithm to offer precise lower bounds. Recently, Lai et al. [38] introduced a mathematical formulation for addressing the SDCARP with Time Windows (SDCARPTW) through graph transformation techniques. They further presented a tabu search algorithm that leverages the forest structure of solutions, effectively reducing the overall cost. Nonetheless, Lai et al. [38] acknowledged the potential for the development of more robust algorithms to tackle this problem.

Both the general CARP and SDCARP are classified as NP-hard problems, meaning they cannot be solved in polynomial time unless $P = NP$. In the SDCARP model, most existing studies fail to address the actual service demands on each edge, which is particularly important for split delivery edges. Due to the complexity of the problem, most heuristic methods proposed for solving CARP and SDCARP are either overly complex, exhibit weak performance for large-scale problems, or are not easily adaptable to different scenarios. To address this gap, in Chapter 3, we formulate the OSRP as a MILP within the framework of the SDCARP. We integrate the structural and mathematical concepts of the OSRP to develop a complete SDCARP model. Furthermore, the integration of exact algorithms with heuristic repair techniques provides practical and efficient solutions for addressing agricultural routing challenges.

To better align routing models with the demands of real-world agricultural path planning, it is crucial to incorporate both accurate service demand representation and practical constraints derived from actual orchard data. An extensive review of path planning applications for UGVs by Chakraborty et al. [18] highlighted that path planning in agricultural environments is one of the most critical areas in the development of agricultural robotic systems. The authors emphasised that future research should prioritise optimisation and validation through real-world testing, as well as the development and sharing of agricul-

tural field datasets to facilitate practical deployment. Santos et al. [68] further classified agricultural path planning into point-to-point and field coverage categories and noted that fewer than half of the reviewed studies had been tested in real-world scenarios. This observation reinforces the ongoing need for practical validation to complement theoretical development.

In terms of scalability, many agricultural path planning studies are tailored to specific scenarios. For example, Xu et al. [83] evaluated a collaborative routing strategy using a real-world orchard layout containing up to 723 fruit-tree locations. Other studies test algorithms on randomly generated grids, such as the DP-A* algorithm for path planning and contactless delivery proposed by Gan et al. [27]. Classical problems such as CARP and SDCARP have typically been benchmarked on instances ranging from tens to several hundred edges, using datasets such as *val* and *egl*, as discussed in Section 3.4. In the present study, three real-world single-block orchards are used for testing, with the largest containing up to 360 nodes and 180 edges. For the multi-block case study, the largest instance features an irregular layout with up to 404 nodes and 261 edges.

However, many theoretical works do not report problem sizes explicitly or rely on synthetic datasets with simplified structures. This gap further highlights the challenge of translating theoretical advances into real-world agricultural contexts, where field structures are irregular, and operational constraints, such as vehicle capacity, time windows, or terrain irregularity, are far more pronounced. Addressing this disconnect remains a vital direction for future research in agricultural robotics and optimisation.

To overcome the shortage of test datasets in agricultural spraying applications, this research introduces three real-world orchard graph instances and generates multiple datasets with realistic characteristics. These instances broaden the research scope in agricultural spraying, providing robust models for researchers to test, validate, and refine existing benchmark datasets, as well as to evaluate CARP and SDCARP.

Few studies consider graph structure when solving CARP or SDCARP, despite the fact that road networks in urban environments and orchards are typically specialised types of graphs (e.g., planar, grid-like, or regular). Also there is far more work on heuristics than exact methods for SDCARPs. In Chapter 4, we begin by examining the orchard layout, introducing regular grid graphs and presenting theorems on optimal solution patterns. We develop a polynomial-time solution for the CARP on regular grid graphs, offering practical applications in agricultural spraying that are straightforward, scalable, and easily adaptable to other agricultural scenarios.

Chapter 3

The Split Delivery Capacitated Arc Routing Problem in Agricultural Spraying Applications

In this chapter, we model the OSRP as an SDCARP, formulating it as a MILP. Two mathematical models are proposed: a *basic spray model* and another adapted for scenarios with *large demand edges*.

The general SDCARP is classified as NP-hard, as it reduces to the NP-hard RPP when assuming unlimited robot capacity. This chapter introduces exact solution techniques that incorporate lazy constraints, additional constraints, symmetry elimination, and a heuristic repair method. More specifically, the lazy constraints, additional constraints, and symmetry elimination techniques provide lower bounds, while the heuristic repair method offers upper bounds and accelerates the solution-finding process. We evaluated the algorithm on data sets derived from a real-world application and on classical instances from CARP literature. This research contributes to the advancement of agricultural robots for autonomous spraying tasks and provides valuable insights into mobile routing challenges.

This chapter is organised as follows: Section 3.1 introduces the notation and formulates the OSRP as an SDCARP for the basic scenario. Section 3.2 examines the characteristics of OSRP and introduces a model tailored for large demand edges. The methodology for problem-solving is described in Section 3.3. Section 3.4 validates the effectiveness of our algorithms, and finally, Section 3.5 concludes with a summary of results and potential directions for future research.

3.1 Notation and OSRP Basic Formulation

In this chapter, a basic spray model of the OSRP is introduced, building upon the SD-CARP model. Section 3.1.1 outlines the variables used in this chapter, while the complete list of symbols for this thesis can be found in the List of Symbols.

3.1.1 Notation

The following list outlines several symbols that will be utilised in this chapter.

List of Symbols:

$\mathcal{N} = \{1, 2, ..., n\}$ set of all nodes. $|\mathcal{N}|$ represents the number of nodes.

$\mathcal{A} \subset \{(i, j) : i, j \in \mathcal{N},\ i \neq j\}$ set of all existing directed arcs in a graph.

$\mathcal{E} = \{(i, j) \in \mathcal{A} : i < j\}$ set of undirected edges (set of arcs when the direction is not a consideration). $|\mathcal{E}|$ represents the number of edges.

$\mathcal{E}^+$ set of demand edges with positive demand, where $\mathcal{E}^+ \subset \mathcal{E}$. $|\mathcal{E}^+|$ represents the number of demand edges.

$\mathcal{G} = (\mathcal{N}, \mathcal{E})$ undirected connected graph.

P capacity of the robot sprayer.

$\delta^-(k) = \{i \in \mathcal{N} : (i, k) \in \mathcal{A}\}$ set of incoming arcs of node k.

$\delta^+(k) = \{j \in \mathcal{N} : (k, j) \in \mathcal{A}\}$ set of outgoing arcs of node k.

$\mathcal{R} = \{1, 2, ..., R\}$ set of the index of the *robotic sprayers tour*. We refer to robotic sprayers as *robots* for simplicity. R is the minimum number of robots required to spray the orchard (typically $|\mathcal{R}| = R = \left\lceil \frac{\sum_{(i,j)\in\mathcal{E}} D_{ij}}{P} \right\rceil$).

$\mathcal{R}_{new} = \{1, 2, ...,\mathbf{min}(|\mathcal{E}^+| - 1, |\mathcal{R}|)\}$ set of indices for the robotic sprayer's tours that spray more than one demand edge in the large edge demand model.

$\mathcal{S}$ subsets of nodes that includes the depot, where $\mathcal{S} \subset \mathcal{N}$.

$\mathcal{T}$ subsets of nodes, defined as $\mathcal{T} = \mathcal{N} \setminus \mathcal{S}$.

$\mathcal{F}(\mathcal{S})$: subset of nodes that includes at least one of the end nodes of demand edges in $\mathcal{S}$.

C_{ij} non-negative cost of arc $(i, j) \in \mathcal{A}$, where $C_{ij} = C_{ji}$.

D_{ij} non-negative demand for the edge $(i, j) \in \mathcal{E}$.

$SPath_{ij}$ the shortest path from node i to node j ($\forall\ i, j \in \mathcal{N}$) calculated using *Dijkstra's Algorithm*.

SP_{ij} the shortest distance (or cost) from node i to node j ($\forall\ i, j \in \mathcal{N}$) calculated using *Dijkstra's Algorithm*.

Γ the number of times the heuristic will run without improvement before abandoning further attempts to execute the repair heuristic.

Decision Variables:

x_{ij}^r binary variable, equal to 1 if the arc $(i, j) \in \mathcal{A}$ is traversed by $r \in \mathcal{R}$, and 0 otherwise.

$\mathbf{x}$ set of solutions for x variables, defined as $\mathbf{x} = \{x_{ij}^r : (i, j) \in \mathcal{A}, r \in \mathcal{R}\}$.

y_{ij}^r amount of chemicals sprayed on edge $(i, j) \in \mathcal{E}$ by robot $r \in \mathcal{R}$, where $y_{ij}^r = 0$ when $D_{ij} = 0$.

$\mathbf{y}$ set of solutions for y variables, defined as $\mathbf{y} = \{y_{ij}^r : (i, j) \in \mathcal{E}, r \in \mathcal{R}\}$.

$z_{ij} \in \mathbb{Z}_+$ the number of single-edge robots (i.e., robots that spray only a single edge) assigned to edge $(i, j) \in \mathcal{E}$. If no such robot is assigned, then $z_{ij} = 0$.

$t_{ij} = \sum_{r=1}^{|\mathcal{R}|} x_{ij}^r$ non-negative integers representing the total number of times that robots have passed edge (i, j), where $(i, j) \in \mathcal{A}$.

3.1.2 OSRP Basic Formulation

The mathematical formulation of the basic model can be stated as follows:

$$\min \quad \sum_{r\in\mathcal{R}} \sum_{(i,j)\in\mathcal{A}} C_{ij} x_{ij}^r \tag{3.1a}$$

s.t.

$$\sum_{(i,j)\in\mathcal{E}} y_{ij}^r \leq P \qquad \forall\, r \in \mathcal{R} \tag{3.1b}$$

$$\sum_{r\in\mathcal{R}} y_{ij}^r = D_{ij} \qquad \forall\, (i,j) \in \mathcal{E} \tag{3.1c}$$

$$y_{ij}^r \leq P(x_{ij}^r + x_{ji}^r) \qquad \forall\, (i,j) \in \mathcal{E},\ r \in \mathcal{R} \tag{3.1d}$$

$$\sum_{i\in\delta^-(k)} x_{ik}^r = \sum_{j\in\delta^+(k)} x_{kj}^r \qquad \forall\, k \in \mathcal{N},\ r \in \mathcal{R} \tag{3.1e}$$

$$\sum_{i\in\mathcal{S},j\in\mathcal{T}} P x_{ij}^r \geq \sum_{i,j\in\mathcal{T}} y_{ij}^r \qquad \begin{array}{l}\forall\, \mathcal{S} \subset \mathcal{N},\ 1 \in \mathcal{S},\\ \mathcal{T} = \mathcal{N} \setminus \mathcal{S},\ r \in \mathcal{R}\end{array} \tag{3.1f}$$

$$x_{ij}^r \in \{0,1\} \qquad \forall\, (i,j) \in \mathcal{A},\ r \in \mathcal{R} \tag{3.1g}$$

$$y_{ij}^r \geq 0 \qquad \forall\, (i,j) \in \mathcal{E},\ r \in \mathcal{R} \tag{3.1h}$$

The objective in (3.1a) minimizes the overall travel distance while ensuring that the robot capacity constraints in (3.1b) are satisfied and the demand outlined in (3.1c) is fulfilled. Constraints (3.1c) ensure that the amount of chemicals sprayed by a robot on each edge does not exceed the robots capacity. Constraints (3.1d) ensure that the edge (i,j) can be sprayed by a robot only if robot r traverses the edge (i,j). Robot r is allowed to serve the edge (i,j) in both directions.. The network flow constraints described in (3.1e) require that the robot paths form complete cycles. Constraints (3.1f) ensure there are no disconnected sub-cycles by mandating that any robot delivering spray across a graph cut between node sets $\mathcal{S}$ and $\mathcal{T}$ must satisfy connectivity requirements. Finally, the routing variables (x_{ij}^r) and spray delivery variables (y_{ij}^r) are defined as binary and non-negative, respectively, in constraints (3.1g) and (3.1h).

3.2 Properties of the OSRP

The following outlines the definitions and relevant properties of the OSRP.

Definition 3.2.1. The *support* of a robot in a solution is the set of edges where the robot satisfies the demand $S_r(y) = \{(i,j) \in \mathcal{E},\ y_{ij}^r > 0\}$. For any solution $\mathbf{x}, \mathbf{y}$ we can group

robots into categories based on the size of the support:

$$R_0(y) = \{r : S_r(y) = \emptyset\} = \text{unused robots}$$
$$R_1(y) = \{r : |S_r(y)| = 1\} = \text{robots spraying a single edge}$$
$$R_{2+}(y) = \{r : |S_r(y)| \geq 2\} = \text{robots spraying multiple edges}$$

Theorem 3.2.1. For the OSRP, there is an optimal solution denoted by $\mathbf{x}$, $\mathbf{y}$ *such that* $|R_{2+}(y)| < m = |\mathcal{E}^+|$. *In this solution, the cardinality of the set* $R_{2+}(y)$ *is less than* m, *where* m *is equal to the number of undirected demand edges. The remaining robots either remain unused or spray a single edge each.*

Proof. We prove it by contradiction. Assume there exists an optimal solution $\mathbf{x}, \mathbf{y}$ with $|R_{2+}(y)| \geq m$. Create an auxiliary graph where the nodes of the auxiliary graph are the edges of the original graph, and there exists a link in the auxiliary graph if the edges in the original graph were sprayed by the same robot. Then there are at least m robots covering multiple edges, meaning the auxiliary graph has at least m nodes and m edges. Thus there exists at least one cycle in the auxiliary graph. Fig. 3.1 give an example to illustrate the proof idea.

Step 1: There are greater than or equal to m edges in the auxiliary graph, identify a cycle C.

Step 2: Balancing spray around cycle C, update the solution by an amount sufficient to ensure one node in the auxiliary graph is not sprayed. Note that for a cycle, every node visited by the cycle has exactly two adjacent edges. The divergence of the nodes equals zero. In other words, if we increase the amount of spray passes over an edge respect to the passes assigned to robot r in the original graph, the amount of spray passes over the same edge from another robot will decrease. Specifically, if there exists a demand edge (i, j) with a spray demand greater than or equal to P, increase the spray on that edge to P, such that $y^r_{ij} = P$. Otherwise, increase the spray to match the nodes actual demand for one robot. Afterwards, reduce the remaining spray for other nodes to 0 and adjust the sprays of other nodes accordingly to satisfy the constraints of the robots capacity and the nodes required demand.

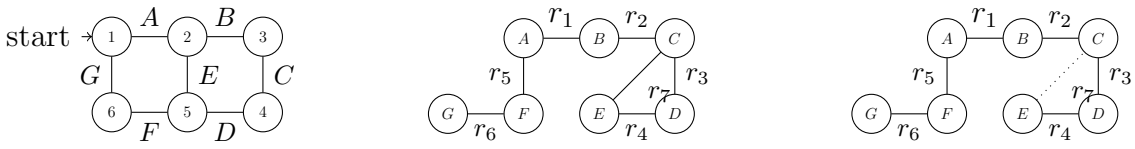
Step 3: Remove edges in the auxiliary graph with zero sprays in the updated solution.

Step 4: The number of cycles C (summed across all robots) has decreased by at least 1. If there are greater than or equal to m edges in the auxiliary graph, go to Step 1.

Step 5: Updating the auxiliary graph find $\overline{y}$ (updated solution of set y) that satisfies $|R_{2+}(\overline{y})| < |R_{2+}(y)|$.

We get a new solution $(x, \overline{y})$ with fewer robots spraying multiple edges. In conclusion,

there exists an optimal solution with less than m robots spraying multiple edges. □



(a) Original graph (b) Auxiliary graph (c) Updated graph

Figure 3.1: An example to illustrate the proof idea of Theorem 3.2.1: A to G represent demand edges, and 1 to 6 are nodes. $\{r_1, r_2, \ldots, r_7\}$ denote multi-edge robot tours, where r_1 sprays edges $\{A, B\}$, r_2 sprays $\{B, C\}$, and so on (as can be easily observed from Fig. 3.1b). Transfer the edges of the original graph to the nodes to construct an auxiliary graph. More specifically, the nodes of the auxiliary graph are the edges of the original graph, and there exists a link in the auxiliary graph if the original two edges share the same node. We decrease the number of robots by breaking the cycles in the auxiliary graph. In this example, we break the cycle in Fig. 3.1c, and the number of robots decreases from 7 to 6.

Theorem 3.2.1 can be derived from the known results in Belenguer et al. [10], which are equivalent to Theorem 3.2.1. Belenguer et al. [10] introduced two main theorems. Theorem 1 states that there always exists an optimal solution for the OSRP that contains no k-split cycle. Theorem 3.2.2 asserts that there exists an optimal OSRP solution such that, for any subset of required edges, the number of splits in these edges is less than the number of trips servicing them. Both theorems focus on comparing splits (defined on each edge) and robots. However, our problem is about specifying the number of robots that spray multiple edges, an objective which is clearly different from Belenguer et al. [10].

Based on the above theorem, we can further reduce the size of the problem by simply using robots that empty their whole capacity on a single row whenever that row has a demand greater than the robots capacity. The next theorem shows that this strategy does not guarantee an optimal solution.

Theorem 3.2.2. For arbitrary robot spray routing problems, the condition $D_{ij} > P$ is *not* sufficient to guarantee that there exists an optimal solution with a robot r such that $\overline{y}_{ij}^{r} = P$.

Proof. We prove this by providing a counter-example. Consider the problem in Fig. 3.2. As shown in the figure, $D_{23} = 12$, all other $D_{ij} = 1$ and all $C_{ij} = 1$. Let $P = 8$; then $D_{23} > P$.

Table 3.1a provides an optimal solution to OSRP formulation 1 at a cost of 7. Table 3.1b demonstrates that when the first robot is restricted to only spray edge $(2, 3)$, the cost increases to 9. Therefore, in any robot sprayer routing problem, if $D_{ij} > P$ for

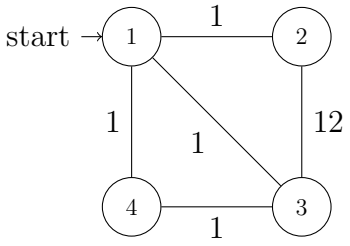


Figure 3.2: Counter-example: An undirected graph and the fixed depot is node 1. A robotic sprayer starts from the depot to spray chemicals to edges (1,2), (1,3), (1,4), (2,3), and (3,4). The number along each edge is the edge corresponding required demands.

Table 3.1: Solution of the counter-example of Fig. 3.2

r_1	$(1,2)$	$(2,3)$	$(3,1)$	
	$y_{12}^{r_1}=0$	$y_{23}^{r_1}=7$	$y_{31}^{r_1}=1$	
r_2	$(1,2)$	$(2,3)$	$(3,4)$	$(4,1)$
	$y_{12}^{r_2}=1$	$y_{23}^{r_2}=5$	$y_{34}^{r_2}=1$	$y_{41}^{r_2}=1$

(a) Optimal solution

r_1	$(1,2)$	$(2,3)$	$(3,1)$			
	$\overline{y}_{12}^{r_1}=0$	$\overline{y}_{23}^{r_1}=8$	$\overline{y}_{31}^{r_1}=0$			
r_2	$(1,2)$	$(2,3)$	$(3,4)$	$(4,1)$	$(1,3)$	$(3,1)$
	$\overline{y}_{12}^{r_2}=1$	$\overline{y}_{23}^{r_2}=4$	$\overline{y}_{34}^{r_2}=1$	$\overline{y}_{41}^{r_2}=1$	$\overline{y}_{13}^{r_2}=1$	$\overline{y}_{31}^{r_2}=0$

(b) Solution with $\overline{y}_{ij}^{r}=P$

some $(i,j) \in \mathcal{E}$, there may not always be an optimal solution where a robot r satisfies $y_{ij}^r = P$. □

Based on Theorem 3.2.1 there is an opportunity to create a simpler formulation when some of the edges demands D_{ij} are much greater than P.

Apply *Dijkstra's algorithm* to calculate the shortest path distance from the start node 1 (depot) to other nodes in the graph. Let SP_{1i} be the shortest path distance from node 1 to node i, $\mathcal{R}_{new} = \{1, 2, ..., \min(|\mathcal{E}| - 1, |\mathcal{R}|)\}$ set of robots that work on multiple edges, and $z_{ij} \in \mathbb{Z}_+$ represents the number of single-edge robots that spray edge $(i,j) \in \mathcal{E}$. If there is no single-edge robot, $z_{ij} = 0$.

The mathematical formulation of the Large Demand Model can be stated as follows:

$$\min \quad \sum_{(i,j)\in\mathcal{E}} z_{ij}(SP_{1i} + C_{ij} + SP_{j1}) + \sum_{r\in\mathcal{R}_{new}} \sum_{(i,j)\in\mathcal{A}} C_{ij} x_{ij}^r \tag{3.2a}$$

s.t.

$$\sum_{(i,j)\in\mathcal{E}} y_{ij}^r \leq P \qquad \forall r \in \mathcal{R}_{new} \tag{3.2b}$$

$$\sum_{r\in\mathcal{R}_{new}} y_{ij}^r + z_{ij}P \geq D_{ij} \qquad \forall (i,j) \in \mathcal{E} \tag{3.2c}$$

$$y_{ij}^r \leq P(x_{ij}^r + x_{ji}^r) \qquad \forall (i,j) \in \mathcal{E}, r \in \mathcal{R}_{new} \tag{3.2d}$$

$$\sum_{i\in\delta^-(k)} x_{ik}^r = \sum_{j\in\delta^+(k)} x_{kj}^r \qquad \forall k \in \mathcal{N}, r \in \mathcal{R}_{new} \tag{3.2e}$$

$$\sum_{i\in\mathcal{S}, j\in\mathcal{T}} P x_{ij}^r \geq \sum_{i,j\in\mathcal{T}} y_{ij}^r \qquad \begin{matrix}\forall \mathcal{S}\subset\mathcal{N}, \mathcal{S}\cup\mathcal{T}=\mathcal{N},\\ 1\in\mathcal{S}, \mathcal{T}=\mathcal{N}\setminus\mathcal{S}, r\in\mathcal{R}_{new}\end{matrix} \tag{3.2f}$$

$$x_{ij}^r \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}, r \in \mathcal{R}_{new} \tag{3.2g}$$

$$y_{ij}^r \geq 0 \qquad \forall (i,j) \in \mathcal{E}, r \in \mathcal{R}_{new} \tag{3.2h}$$

$$z_{ij} \in \{0, 1, \ldots, \left\lfloor \frac{D_{ij}}{P} \right\rfloor\} \qquad \forall (i,j) \in \mathcal{E} \tag{3.2i}$$

$$\sum_{(i,j)\in\mathcal{E}} z_{ij} \geq \left\lceil \frac{\sum_{(i,j)\in\mathcal{E}} D_{ij}}{P} \right\rceil - |\mathcal{R}_{new}| \tag{3.2j}$$

The key idea is splitting the large demand on a single edge amongst vehicles to involve fewer vehicles and decrease the number of decision variables x_{ij}^r and y_{ij}^r when $|\mathcal{R}| > |\mathcal{E}| - 1$. Theorem 3.2.1 indicates that there exits at least one robotic sprayer r that $\overline{y}_{ij}^r = P$ when $|\mathcal{R}| > |\mathcal{E}| - 1$. More specifically, there are at most $|\mathcal{E}| - 1$ robots that spray multiple edges (i.e., multi-edge robots), and at least $|\mathcal{R}| - (|\mathcal{E}| - 1)$ robots that spray a single edge (except in the case of the last robot covering the rest of the demand edge). Formulation (3.2) is analogous to Formulation (3.1), except that the objective function ((3.2a)) considers the

total costs of robots covering single and multiple demand edges, and constraints (3.2c) indicate that each edge can be serviced by robots that spray single or multiple demand edges, so that the total demand is satisfied.

3.3 Methodology

3.3.1 Lazy Constraints

Lazy constraints are useful when the full set of constraints is too large to include in the initial formulation. The main idea is to remove the costly constraints and get a relaxed problem with the remaining constraints, and then reintroduce that constraint if the relaxation solutions violate some of the constraints we left out.

The basic spray formulation described in Section 3.1 is costly for a large-scale problem because the number of constraints (3.1f) grows exponentially as the nodes increase in the graph. An alternative is treating them as lazy constraints and only adding them when violated.

In the basic spray model, we generate the set S of nodes connected to depot 1 by edges with $x_{ij}^r > 0.5$ for each robot r. We then check if the resulting set $\mathcal{S}$, which is a subset of $\mathcal{N}$, violates the constraint. If it does, we reintroduce it; if not, we leave it out.

The constraints could be added at both branch-and-bound nodes with integer and fractional LP solutions. For computational efficiency, we add constraints for fractional solutions early in the search and all infeasible integer solutions. More specifically, we add lazy constraints on a limited number of nodes (not too deep in the branch and bound tree) when the gap between the lower and upper bounds is not too small.

3.3.2 Additional Constraints

We introduce some constraints to enhance precision and expedite the generation of solutions. In subsequent discussions, we will provide additional constraints for the basic spray model and relevant ones for scenarios with large edge demand.

For efficient variable branching and quick utilisation in the solver, we introduce the auxiliary variables $t_{ij} = \sum_{r=1}^{|\mathcal{R}|} x_{ij}^r$, which are non-negative integers representing the total number of times that robots traverse edge (i, j), where $(i, j) \in \mathcal{A}$. These variables correspond to constraints (3.1g) summed over all $r \in \mathcal{R}$.

Then we have:

$$\sum_{j\in\delta^+(i)} t_{ij} \geq \left\lceil \frac{|\delta^+(i)|}{2} \right\rceil \quad \forall\, i \in \mathcal{N},\ \delta^+(i) \text{ is odd} \tag{3.3}$$

$$\sum_{j\in\delta^+(i)} t_{ij} + \sum_{\substack{i<j \\ j\in\delta^+(i)}} z_{ij} + \sum_{\substack{i>j \\ j\in\delta^-(i)}} z_{ji} \geq \left\lceil \frac{|\delta^+(i)|}{2} \right\rceil \quad \forall\, i \in \mathcal{N},\ \delta^+(i) \text{ is odd} \tag{3.4}$$

The inequalities specified in inequality (3.3) (referred to as inequalities (3.4) in the context of the large edge demand model (3.2)) ensures that when i has an odd degree, the *demand edges* incident to node i must be traversed by at least an integer greater than half of $|\delta^+(i)|$. This helps avoid some fractional solutions for t_{ij}. It is worth noting that (i, j) are *demand edges* in inequalities (3.3) and inequalities (3.4).

Furthermore, we also introduced some rounding cuts during the branch and bound process, which accelerated the optimal solution-finding process.

$$\sum_{i\in\mathcal{S},j\in\mathcal{T}} t_{ij} \geq \left\lceil \sum_{(i,j)\in\mathcal{F}(\mathcal{T})} \frac{D_{ij}}{P} \right\rceil \tag{3.5}$$

$$\sum_{i\in\mathcal{S},j\in\mathcal{T}} t_{ij} + \sum_{(i,j)\in\mathcal{F}(\mathcal{T})} z_{ij} \geq \left\lceil \sum_{(i,j)\in\mathcal{F}(\mathcal{T})} \frac{D_{ij}}{P} \right\rceil \tag{3.6}$$

Inequalities (3.5) (referred to as inequalities (3.6) in the context of the large edge demand model (3.2)) ensures that the number of robots from $\mathcal{S}$ to $\mathcal{T}$ must be greater than or equal to the number of robots required on the bridge and in subset $\mathcal{T}$. $\mathcal{F}(\mathcal{T})$ is a subset of nodes that includes at least one of the end nodes of demand edges in $\mathcal{T}$.

3.3.3 Symmetry Elimination Constraints

There are many equivalent ways to encode the same solution in our formulation simply by renumbering the robots. We can eliminate this symmetry with constraints (3.7). These constraints order the robots from less costly to more costly to eliminate symmetric solutions. The set $\mathcal{R}$ should be replaced with $\mathcal{R}_{new}$ in the large edge demand model (3.2).

$$\sum_{(i,j)\in\mathcal{A}} C_{ij} x_{ij}^r \geq \sum_{(i,j)\in\mathcal{A}} C_{ij} x_{ij}^{r-1} \quad \forall r \in \mathcal{R} : r > 1 \tag{3.7}$$

Traversing a route in the reverse order does not change the solution and thus creates another form of symmetry. To eliminate this symmetry, $\delta^-(1)$ is the set of nodes for arcs

Algorithm 3.1 Greedy Routing Algorithm

Input: For a robot $r \in \mathcal{R}$, $y = \{y^r_{ij} : (i,j) \in \mathcal{E}\}$, $SPath_{ij}$, SP_{ij}
Output: $Tour$

$Spray \leftarrow \{(i,j) \,|\, y^r_{ij} > 0\}$, $Tour \leftarrow [1]$ ▷ Node 1 is the depot
while $Spray \neq \emptyset$ **do**
  if $(Tour[\text{end}], s)||(s, Tour[\text{end}]) \in Spray$ **then**
    $Tour \leftarrow [Tour, s]$ ▷ Append node s to the tour
    $Spray \leftarrow Spray \setminus \{(Tour[\text{end}], s), (s, Tour[\text{end}])\}$
  else
    $MinCost(t) \leftarrow \min \left\{ SP_{Tour[\text{end}],t} \,\middle|\, (t,s)||(s,t) \in Spray \right\}$
    $Tour \leftarrow [Tour, SPath_{Tour[\text{end}]\,t}, s]$ ▷ Append shortest path and node s to the tour
    $Spray \leftarrow Spray \setminus \{(t,s),(s,t)\}$
for $i \in 1 : (\text{length}(Tour) - 1)$ **do**
  if $Tour[i] = Tour[i+1]$ **then**
    $Tour \leftarrow Tour \setminus Tour[i]$
if $Tour[\text{end}] \neq 1$ **then**
  $Tour \leftarrow [Tour, SPath_{Tour[\text{end}]\,1}]$
return $Tour$

coming into start node 1. The constraints (3.8) orient the route by not allowing the return arc at the depot to connect to a lower-indexed node than the outgoing arc. However, this doesn't help if the first and last edges visited by the robot are the same.

$$\sum_{i=1}^{k} x^r_{1i} \geq x^r_{k1} \quad \forall k \in \delta^-(1), \;\; \forall r \in \mathcal{R}\,. \tag{3.8}$$

3.3.4 Heuristic Repair Method

The main idea of the heuristic repair method is to construct a feasible solution from an infeasible one and use the generated feasible solution as new incumbent during the branch-and-bound process. The main components are described in the following paragraphs.

Preprocessing step: Compute the shortest path and corresponding shortest *distance* (cost) from node i to node j ($\forall\, i,j \in \mathcal{N}$) by *Dijkstra's Algorithm*, represented by $SPath_{ij}$ and SP_{ij}, respectively.

An infeasible solution with feasible y values is utilised to construct a connected path that covers all edges (i,j) where $y^r_{ij} > 0$. This construction is achieved using the *Greedy Routing Algorithm* described in Algorithm 3.1. Since y^r_{ij} values can be fractional, they are inherently feasible. It is worth noting that in chaining paths from one spray edge to the next, we may be traversing an arc in the same direction twice and that we can trivially improve the solution by removing such duplicates. Then generate x^r_{ij} according to the connected path. Finally, sort x^r_{ij} and y^r_{ij} according to the symmetry elimination constraints in (3.7) and (3.8).

The primary idea outlined in Algorithm 3.1 revolves around leveraging the y-values

to formulate a feasible solution. For example, if robot 1 possesses positive values for y^1_{13}, y^1_{34}, and y^1_{56}, indicating that the robot sprayers should cover demand edges $(1,3)$, $(3,4)$, and $(5,6)$, a potential tour could be constructed as follows: $Tour = \{1 \to 3 \to 4 \to 5 \to 6 \to 3 \to 4 \to 3 \to 1\}$. However, there exists an enhanced tour with reduced cost that circumvents the duplication of the arc $(3,4)$ and ensures that each arc (i,j) in $\mathcal{A}$ is utilised only once by each robot.

To improve the solution, we can employ the following methods to handle reductions:

(1) If the path contains $\ldots, i, j, \ldots, i, j$, reverse the subsequence between the two occurrences of j. This results in a walk with i, j, i.

(2) If the walk contains $\ldots, i, j, i \ldots$ and another occurrence of (i,j) or (j,i), then delete the superfluous i, j, i part of the path.

Repeatedly applying these actions whenever the same arc appears twice will eventually lead to a directed circuit, where each directed arc appears at most once. For example, we can obtain the tour $\{1 \to 3 \to 4 \to 3 \to 6 \to 5 \to 4 \to 3 \to 1\}$ through the first step and $\{1 \to 3 \to 6 \to 5 \to 4 \to 3 \to 1\}$ through the second step. This revision removes the redundancy caused by the duplicated edge $(3,4)$.

To enhance the efficiency of the heuristic techniques, we introduce a parameter Γ representing the number of times the heuristic will run without improvement before abandoning further attempts to execute the repair heuristic. In the following chapter, we set Γ to 10,000.

3.4 Numerical Results

In this section, we evaluate the effectiveness of our algorithms in solving OSRP by examining three main classes of instances. As existing benchmark instances differ structurally from our real-world applications, we generate the first and second classes of data sets based on the attributes of real-world examples described by Plessen [65].

The first class of instances consists of nine different networks, some with random obstacles and others without. Specifically, instances Ins.A, Ins.B, and Ins.C depict small-scale orchards without obstacles, whereas instances Ins.D, Ins.E, and Ins.F feature one random obstacle each. Instances Ins.G, Ins.H, and Ins.I represent larger scenarios with two obstacles.

The second set of instances comprises five large edge demands networks derived from both our generated dataset and classical instance sets val and egl, denoted as LD(A),

LD(E), LD(H), LD(val7A), and LD(egl-s4-A). These five instances are used to assess the efficacy of the large demand model across various scenarios.

Additionally, we evaluate our algorithms performance using benchmark instances derived from well-known classical CARP studies. The data for *val* was created by Benavent et al. [11], while *egl* was created by Li and Eglese [44]. The *val* instance set in classical CARP comprises *medium-sized* undirected graphs, consisting of 10 distinct graphs with varying vehicle capacities, where all edges are necessary. On the other hand, the *egl* group contains *large-scale* instances. Subsequently, we compare our algorithms results with the state-of-the-art outcomes for the SDCARP problem using these instances, as detailed in the *val* file in Table 3.4 and the *egl* file in Table 3.5. The *val* and *egl* datasets have a data structure that deviates from our actual orchard graphs, which is uncommon in our typical scenarios. Nonetheless, its valuable to evaluate our model and solution method using these datasets to compare them with existing literature.

The algorithm described in this chapter was implemented in Julia 1.10, utilizing Gurobi 10.0.1 as the MIP solver, and tested on machines equipped with 2.70 GHz Intel Xeon-Gold-6150 processors. We used JUMP, a modelling language, and a collection of supporting packages for mathematical optimisation in Julia [49]. We restricted the total workload to 7200 units, each unit roughly equivalent to one second, in Gurobi by configuring the *WorkLimit* parameter accordingly. Initially, we ran with a work limit of 3600 units and parameter settings chosen to focus on finding good solutions. A subsequent run used parameter settings that focused on improving the dual bound to prove optimality. These work limits are deterministic, meaning that under identical hardware, parameter, and attribute settings, the optimisation process halts at the same point every time.

3.4.1 Computational Results for Different Techniques

In this subsection, we analyze the algorithms and techniques discussed in Section 3.3, and elaborate on the comparison of results presented in Table 3.2. In this table, LC refers to the use of Lazy Constraints, while Cons indicates Additional Constraints, and Sym and Heu represent Symmetry Eliminations and Heuristic Repair techniques, respectively. $|\mathcal{N}|$ represents the number of nodes, and $|\mathcal{E}|$ represents the number of edges. The ratio $\sum_{(i,j)\in\mathcal{E}} D_{ij}/(R\,P)$ indicates how capacity-constrained the problem is with the selected number of robots. In the present chapter, R is the minimum number of robots required for our problem.

The results are derived from five rounds, each initiated with a distinct random seed number. Although different seed numbers typically yield different solution paths, the outcomes are frequently similar. However, in some cases, significant differences in solu-

tions may arise. The UP column showcases the *best Upper Bound* (objective solution), while the LB and Cut columns represent the acquired *best Lower Bound* and the average number of cuts in the five rounds, respectively. The parameter $|Acc|$ indicates the average count of accepted heuristic repair solutions across the five runs. All algorithms underwent testing within a 7200-work limit (nearly 2 hours). Notably, when the time work limit was doubled for these instances, the solution improved slightly compared to that achieved within the 7200-work limit, but the additional time required was too long. This implies that extending the work limit was not applied in our test.

Table 3.2: The comparative results of various techniques on instances from Ins.A to Ins.I are presented. This set was designed based on real-world orchard data [65]. The results with stars highlight the optimal solution.

		Ins.A	Ins.B	Ins.C	Ins.D	Ins.E	Ins.F	Ins.G	Ins.H	Ins.I
$\|\mathcal{N}\|$		20	30	40	56	66	76	96	106	116
$\|\mathcal{E}\|$		28	43	58	82	97	112	142	157	172
P		20	20	20	20	20	20	20	20	20
R		3	4	5	6	7	9	9	11	12
$\frac{sum(D)}{RP}$		69.33%	81.38%	89.00%	92.08%	96.43%	89.22%	98.33%	91.05%	95.58%
LC	UB	56.78*	82.63*	110.36*	149.93	187.63	252.08			
	LB	56.78	78.63	110.36	144.96	182.49	224.26	231.20	260.98	298.90
	Cut	645	3507	4095	21568	21155	28787	37004	40710	27035
LC Cons	UB	56.78*	82.63*	110.36*	149.93	187.35	232.92			
	LB	56.78	82.63	110.36	145.19	183.49	224.62	236.21	269.79	308.45
	Cut	632	2828	1827	17779	23397	31951	37955	27069	17096
LC Cons Sym	UB	56.78*	82.63*	110.36*	150.45	187.35	268.07			
	LB	56.78	82.63	110.36	145.14	183.49	224.73	237.37	270.94	311.35
	Cut	416	2192	1468	16111	23702	43650	33744	15306	16836
LC Cons Sym Heu	UB	56.78*	82.63*	110.36*	149.93	187.35	234.69	308.35	350.83	406.88
	LB	56.78	82.63	110.36	145.09	183.49	224.62	236.04	270.04	310.62
	Cut	412	2130	1394	12304	18841	34019	33059	24376	28010
	$\|Acc\|$	2	3	3	7	10	13	8	8	9
LC Cons Heu	UB	56.78*	82.63*	110.36*	149.93	187.35	234.47	300.82	369.28	435.73
	LB	56.78	82.63	110.36	145.18	183.49	224.09	234.33	270.32	307.30
	Cut	710	2613	1806	16076	20873	32336	44610	34138	32116
	$\|Acc\|$	3	3	3	5	7	7	12	5	8

Table 3.2 compares various methods introduced in Section 3.4. It begins with Lazy Constraint (LC) as a foundational technique and then integrates Cons on LC to assess the impact of Additional Constraints (Cons). This process is continued with Symmetric Eliminations (Sym) and Heuristic Repair Techniques (Heu). Given the effective performance of symmetric eliminations on large instances Ins.G, Ins.H, and Ins.I, we compare solutions with and without symmetric eliminations to evaluate their effectiveness.

Table 3.2 illustrates that for small-scale instances Ins.A, Ins.B, and Ins.C, all algorithms achieve optimal solutions with a significant reduction in the number of cuts when

additional constraints, symmetric eliminations, and heuristic techniques are employed. This improvement becomes particularly notable after integrating symmetric eliminations, leading to enhanced efficiency. For medium-sized instances Ins.D, Ins.E, and Ins.F, incorporating additional constraints results in better lower bounds, and applying symmetric eliminations and heuristic techniques decreases the number of cuts in Ins.D and Ins.E.

Moving to larger-scale problems, specifically Ins.G, Ins.H, and Ins.I, where lazy constraints and additional techniques fail to provide feasible solutions within the initial 3600-unit working limit, symmetric eliminations and heuristic repair techniques demonstrate their effectiveness by finding feasible solutions. When comparing heuristic techniques with and without symmetric eliminations in the last two block rows, we observe that heuristic techniques combined with symmetric eliminations generally provide better solutions than those without, except for Ins.G. Additionally, employing symmetric eliminations with heuristic techniques results in a higher number of acceptable heuristic repair solutions, except for Ins.G, in larger-scale instances, thereby speeding up the process of finding a high-quality solution in some sense.

In summary, additional constraints effectively enhance lower bounds, while symmetric eliminations typically reduce the number of cuts, thereby potentially improving efficiency. Heuristic repair techniques contribute to obtaining feasible, high-quality solutions with improved upper bounds, and by offering better upper bounds, they help expedite the solution-finding process. The advantages of heuristic repair techniques are especially significant when addressing large-scale problems.

3.4.2 Computational Results for Large Edge Demands Instances

To demonstrate the effectiveness of the large demand model (3.2) we create some addition instances with much larger demand. We conduct experiments based on five instances with different network structures: Ins.A, Ins.E, Ins.H, val7A, and egl-s4-A. We intentionally reduce the robots capacity and raise the demand to develop LD(A), LD(E), and LD(H), along with LD(val7A) and LD(egl-s4-A), which maintain the same network structure as their predecessors. These tests incorporate all techniques (LC, Cons, Sym, Heu) and were solved five times, each initialised with a unique seed number to gather diverse results. Different seed numbers yield varying solution paths and outcomes. STEDVP measures the upper bounds standard deviation, indicating solution stability across seed variations.

The large edge demand model typically uses fewer robots, reducing the number of variables and potentially boosting efficiency. As detailed in Table 3.3, the large demand model surpasses the basic model in upper bound quality for large edge instances LD(val7A) and LD(egl-s4-A) with lower variability, while also producing better lower bounds in all cases

Table 3.3: Comparative analysis of basic and large demand models across network structures

	Basic				Large Edge Demand			
Instance	R	UB	LB	STEDVP	R_{new}	UB	LB	STDEVP
LD(A)	28	331.83	330.92	0.06	8	331.83	330.92	0.00
LD(E)	90	1429.39	1356.66	24.36	31	1445.83	1268.11	9.20
LD(H)	134	2652.05	1863.59	10.66	51	2679.98	2199.07	23.57
LD(val7A)	224	5076.00	4247.00	93.51	66	4806.00	4330.00	11.74
LD(egl-s4-A)	559	428787.00	55900.00	11112.66	190	317156.00	191668.00	5004.52

except LD(E). This discrepancy in upper bound results arises because the optimal solution is derived from five different random seeds. The basic model occasionally achieves superior upper and lower bounds for certain seed numbers, resulting in a better solution. The large edge demand model starts with a significantly higher lower bound (roughly double that of the basic model) for the initial LP relaxation, and actively explores nodes during the branch and bound process, demonstrating the efficacy of the large edge demand model. For example, the initial lower bound for LD(E) is 1051.11, compared to 553.08 for the basic model. The large demand model completes the root node in 121 seconds with a 16.4% gap, while the basic model takes 1061 seconds with an 11.4% gap, showing the large demand models superior performance in solving problems with large edge demands.

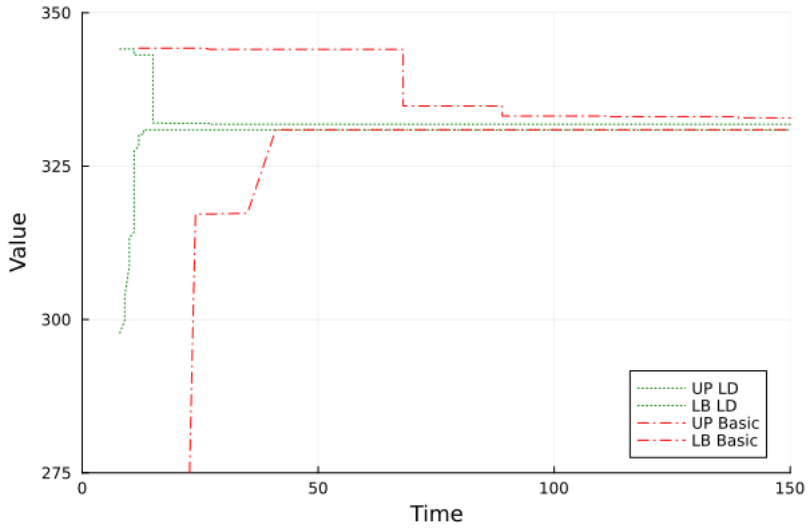


Figure 3.3: Comparison results for LD(A): The graph features green dotted and red dash-dotted lines representing the upper and lower bounds for the large edge demands and the basic model, respectively. It demonstrates that the large edge demands model (green dotted lines) converges quickly and achieves a narrow gap within 150 seconds.

Fig. 3.3 illustrates the performance of both the basic and large demand models on LD(A) during the solution exploration process. The x axis represents time (s), and the y axis represents the values of the lower and upper bounds. The green dotted line represents the upper and lower bounds for the large edge demand model, while the red dash-dotted lines depict the upper and lower bounds for the basic model. As shown in Table 3.3, both models produce the same solution within the work limit setting for Ins.A. However, the large demand model starts with superior lower bounds, and the upper and lower bounds converge rapidly, ultimately yielding a better solution within a short time frame. This emphasizes the effectiveness of the large edge demand model.

3.4.3 Computational Results for Benchmark Instances

The *val* benchmark dataset contains 34 instances designed by Benavent et al. [11] to find the lower bounds of the CARP. These instances have been generated by varying the capacity of the vehicles for each graph (denoted val1a, val1b,...,val10d). The val benchmark has been proven to be optimal, and its optimal solution can be seen as an upper bound of the SDCARP problem. We present the lower bounds for the SDCARP problem, as few studies have reported lower bounds in SDCARP literature using the val dataset. Only [10] have reported part of lower bounds.

The results for the val benchmark are detailed in Table 3.4. The BKR column displays the *Best Known Results* from previous CARP literature, summarized by Chen et al. [19], which proved the optimality of the solution for CARP. Additionally, Lai et al. [37] implemented a forest-based tabu algorithm for the SDCARP problem, conducting tests on the val dataset, which are shown in the FTS column. The columns Heur and Heur-LB feature the solutions obtained by applying heuristic repair techniques and the corresponding lower bounds of this study, respectively. The STDEVP column reports the standard deviation of the heuristic solutions over five trials using randomly selected seeds, within a work limit of 7200 units.

The computational results from the val file, as shown in Table 3.4, indicate that the solution method introduced in this chapter not only achieves better results for instances such as val4d and val6c but also confirms optimality for the majority of the cases. This highlights the efficacy of the current method compared to recent studies on these benchmarks. Notably, many recent studies utilizing the val benchmarks for extended CARP studies do not provide detailed solutions with lower bounds or information about optimality for SDCARP. Arakaki and Usberti [5] introduced an efficiency-based path-scanning heuristic for CARP, which excelled in average time across algorithms but showed poorer deviation for the set of val instances. Similarly, [37] delivered top results in various rounds

Table 3.4: Evaluation of the solution method of this study compared with state-of-the-art algorithms on 34 undirected graphs from the val set. The BKR column displays the optimal results from previous CARP literature, while FTS represents a forest-based tabu algorithm [37] designed for the SDCARP. The columns Heur and Heur-LB feature the solutions obtained by applying heuristic repair techniques and the corresponding lower bounds of this study, respectively. Better solutions are highlighted in grey cells, with a star denoting optimality.

File	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{E}\rvert$	R	P	$\frac{\sum_{(i,j)\in\mathcal{E}} D_{ij}}{RP}$	FTS	BKR	Heur	Heur-LB	STDEVP
val1a	24	39	2	200	89.50%	173	173	173*	173	0.00
val1b	24	39	3	120	99.44%	173	173	173*	173	0.00
val1c	24	39	8	45	99.44%	235	245	235*	235	0.00
val2a	24	34	2	180	86.11%	227	227	227*	227	0.00
val2b	24	34	3	120	86.11%	259	259	259*	259	0.00
val2c	24	34	8	40	96.88%	455	457	455*	455	0.00
val3a	24	35	2	80	85.63%	81	81	81*	81	0.00
val3b	24	35	3	50	91.33%	87	87	87*	87	0.00
val3c	24	35	7	20	97.86%	138	138	138*	138	0.00
val4a	41	69	3	225	92.89%	400	400	400*	400	0.00
val4b	41	69	4	170	92.21%	412	412	412*	412	0.00
val4c	41	69	5	130	96.46%	428	428	428*	428	1.20
val4d	41	69	9	75	92.89%	530	528	528	520	10.81
val5a	34	65	3	220	93.03%	423	423	423*	423	0.00
val5b	34	65	4	165	93.03%	446	446	446*	446	0.80
val5c	34	65	5	130	94.46%	474	474	474	467	2.00
val5d	34	65	9	75	90.96%	577	575	*583*	571	2.04
val6a	31	50	3	170	88.43%	223	223	223*	223	0.00
val6b	31	50	4	120	93.96%	233	233	233*	223	0.00
val6c	31	50	10	50	90.20%	317	317	315	307	1.60
val7a	40	66	3	200	93.17%	279	279	279*	279	0.00
val7b	40	66	4	150	93.17%	283	283	283*	283	0.00
val7c	40	66	9	65	95.56%	334	334	334	327	1.96
val8a	30	63	3	200	94.33%	386	386	386*	386	0.00
val8b	30	63	4	150	94.33%	395	395	395*	395	0.00
val8c	30	63	9	65	96.75%	521	521	*523*	509	0.00
val9a	50	92	3	235	92.77%	323	323	323*	323	0.00
val9b	50	92	4	175	93.43%	326	326	326*	326	0.00
val9c	50	92	5	140	93.43%	332	332	332*	332	0.96
val9d	50	92	10	70	93.43%	391	389	*405*	376	9.57
val10a	50	97	3	250	93.87%	428	428	428*	428	0.00
val10b	50	97	4	190	92.63%	436	436	436*	436	0.98
val10c	50	97	5	150	93.87%	446	446	446*	446	0.89
val10d	50	97	10	75	93.87%	528	525	*536*	521	7.50

but lacked lower bounds or details on optimality for their SDCARP study. Moreover, [10] established lower bounds on several instances within the val dataset. The solution method introduced in this chapter effectively matches these lower bounds, achieving and confirming optimality for most of the instances reported by Belenguer et al. [10]. Although our approach slightly falls short of val9d (378 in their study versus 376 in ours), our method consistently achieves or exceeds the lower bounds in most scenarios with minimal standard deviation, highlighting its robustness and effectiveness in addressing these complex problems.

The val instances are typically considered medium-sized, while the egl instances are regarded as large-scale. Table 3.5 evaluates the solution method of this chapter alongside state-of-the-art algorithms on large-scale instances for SDCARP. The egl instances consist of 77 nodes with various numbers of demand edges, where the indicator $\sum_{(i,j)\in\mathcal{E}} D_{ij}/(R\,P)$ stands at 100%. The column Heur-Gap calculates the discrepancy between our solution and the lowest value (min(FTS, CARP-BKR)) among the CARP best-known results (CARP-BKR) and the state-of-the-art FTS as presented in the study by Lai et al. [37]. The column LB1 reports the best lower bound achieved by our method over five trials with different seed numbers, while LB1-Gap shows the gap between our heuristic solution (Heur) and LB1. The columns LB2 and LB2-Gap display the lower bound and its gap, respectively, when our objective is set to the minimum value between FTS and CARP-BKR, discarding any solutions worse than this minimum value using the parameter *Cutoff* in Gurobi. The STDEVP column reports the standard deviation of our solutions (Heur).

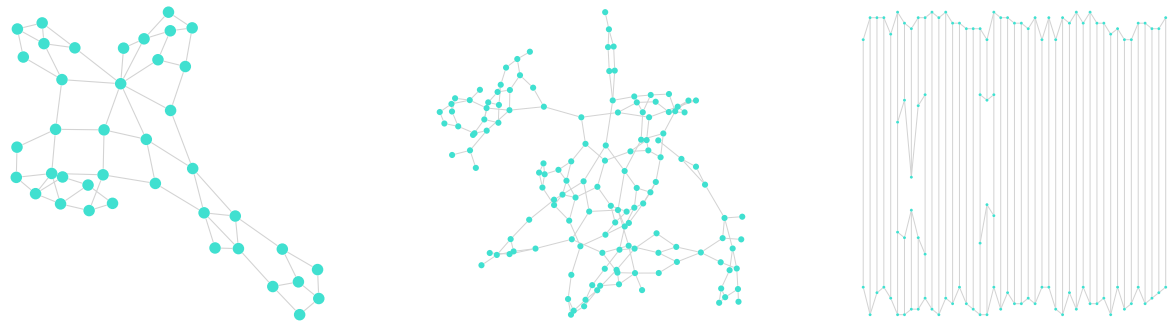
Table 3.5 demonstrate that our method provides better solutions for the instance egl-e1-B. Besides egl-e1-A, egl-e1-B, and egl-e1-C, the *Cutoff* parameter helps enhance the lower bounds. The graph structures of the val and egl datasets are markedly different from those typically found in practical orchard environments, which are the primary focus of our research. The layout of Ins.H, as depicted in Fig. 3.4, more closely resembles a real-world orchard, though an authentic orchard model would feature a smoother outline rather than directly mimicking the structure of Ins.H. Our model and solution techniques are straightforward and well-suited for orchard spraying applications, and they can be tailored to various data structures by integrating specially designed constraints. While this study does not primarily focus on these adaptations, they represent a promising area for future research to explore the impact of different network shapes.

3.5 Conclusions and Future Work

In this chapter, we modeled the OSRP as a basic model and a large demand model designed specifically to address scenarios with large edge demands. The effectiveness of

Table 3.5: Evaluation of the studys solution method along with state-of-the-art algorithms on the large-scale egl set, originally designed to establish lower bounds for CARP by [44]. The CARP-BKR column presents optimal results from previous CARP studies, while FTS showcases a forest-based tabu algorithm [37] tailored for SDCARP. Better solutions are highlighted in grey cells.

File	$\|\mathcal{E}\|$	r	p	FTS	CARP-BKR	Heur	Heur-Gap	LB1	LB1-Gap	LB2	LB2-Gap	STDEVP
egl-e1-A	51	5	305	3548	3548	3596	1.35%	3406	4.00%	3387	4.54%	17.13
egl-e1-B	51	7	220	4486	4498	4482	0.09%	4242	5.44%	4229	5.73%	37.23
egl-e1-C	51	10	160	5517	5595	5517	0.00%	5305	3.84%	5249	4.86%	140.07
egl-e2-A	72	7	280	5018	5018	5108	1.79%	4841	3.53%	4841	3.53%	150.51
egl-e2-B	72	10	200	6292	6317	6396	1.65%	5980	4.96%	6011	4.47%	58.71
egl-e2-C	72	14	140	8165	8335	8366	2.46%	7323	10.31%	7809	4.36%	256.90
egl-e3-A	87	8	280	5898	5898	5912	0.24%	5772	2.14%	5773	2.12%	326.37
egl-e3-B	87	12	190	7743	7775	8072	4.25%	7007	9.51%	7453	3.75%	1111.13
egl-e3-C	87	17	135	10175	10292	10803	6.17%	8921	12.32%	9374	7.87%	467.68
egl-e4-A	98	9	280	6456	6444	6815	5.76%	6178	4.13%	6335	1.69%	205.99
egl-e4-B	98	14	180	8880	8961	9457	6.50%	8065	9.18%	8261	6.97%	748.70
egl-e4-C	98	19	130	11401	11529	12424	8.97%	10182	10.69%	10217	10.39%	850.45
egl-s1-A	75	7	210	5010	5018	6022	20.20%	3695	26.25%	4403	12.12%	316.43
egl-s1-B	75	10	150	6376	6388	8237	29.19%	4331	32.07%	4495	29.50%	382.25
egl-s1-C	75	14	103	8435	8518	9685	14.82%	5419	35.76%	5378	36.24%	1066.37
egl-s2-A	147	14	235	9963	9874	13799	39.75%	7497	24.07%	7640	22.63%	366.46
egl-s2-B	147	20	160	13109	13078	16587	59.83%	9141	11.92%	9254	10.83%	756.44
egl-s2-C	147	27	120	16369	16425	21124	29.05%	10849	33.72%	10849	33.72%	385.34
egl-s3-A	159	15	240	10261	10201	14624	43.36%	7635	25.15%	7826	23.28%	413.01
egl-s3-B	159	22	160	13792	13682	18576	35.77%	9411	31.22%	9779	28.53%	279.31
egl-s3-C	159	19	120	17167	17188	22156	29.06%	11227	34.60%	11365	33.80%	527.10
egl-s4-A	190	19	230	12324	12216	16471	34.83%	9405	24.48%	9226	23.01%	690.11
egl-s4-B	190	27	160	16340	16201	21142	30.50%	11223	30.73%	11601	28.39%	678.68
egl-s4-C	190	35	120	20490	20476	25107	22.62%	13777	32.72%	15747	23.10%	453.18



(a) val7A (b) egl-s4-A (c) Ins.H

Figure 3.4: Structures of different benchmark instances. Network (3.4c) represents a real orchard more closely than the structures from existing benchmarks.

the large-demand model in managing these demands was validated through test results, highlighting its efficiency in overcoming the challenges posed by such scenarios. This work integrates the SDCARP framework with agricultural operations, establishing a foundation for broader applications such as seed planting and agricultural logistics.

The proposed solution methods for solving OSRP incorporate lazy constraints, symmetry eliminations, and additional constraints, alongside a heuristic repair technique. Numerical results from the generated OSRP test dataset, based on real-world orchards, confirm that the additional constraints are effective in raising the lower bounds. Meanwhile, symmetry eliminations significantly enhance computational efficiency, and the heuristic repair method demonstrates strong performance in finding upper bounds. Additionally, we conducted a comparison of the OSRP model against classical CARP instances using the val and egl datasets, first published by [11, 44], which are commonly used as benchmarks. While the graphs represented by these datasets differ significantly in structure from the typical layout of orchards, our methodology yielded superior results for portions of these datasets. Furthermore, this chapter reports lower bounds for the classical datasets within the context of SDCARP studies, an area that has received limited attention in the literature and could benefit from further exploration in future research.

The complexity of the OSRP poses significant challenges in obtaining high-quality solutions, particularly as the problem size increases. Chapter 4 leverages the structural similarities between orchard layouts and regular grid graphs, utilizing mathematical insights into solution patterns to propose a polynomialy solvable OSRP_{RG}. Moreover, Chapter 4 not only establishes a mathematical foundation for understanding the OSRP but also offers a fresh perspective on solution methods for real-world applications.

Chapter 4

A Polynomial Time Solvable Capacitated Arc Routing Problem on Regular Grid Graphs

There are numerous applications of simple graphs across various domains. These graphs, consisting of nodes connected by edges without loops or multiple edges, offer a straightforward structure for modelling relationships and connections in diverse fields. For example, Hadlock [34] introduced a shortest path algorithm specifically for grid graphs. Apollonio et al. [4] investigated complex path covering problems on grid graphs (general grids, grids with a fixed number of rows, ladders), including several collections of special, unweighted paths. Zachariasen [84] developed a framework to address routing design issues on the Hanan grid. Köhler et al. [36] focused on computing lower bounds for the minimum strictly fundamental cycle basis problem in square planar grid graphs.

In this thesis, we address a real-world orchard spraying problem, referred to as OSRP. We observe that the layout of orchards can be approximated by grid graphs, characterised by parallel demand edges and consistent distances between them. This observation inspired a theoretical investigation into solution patterns for the OSRP on these grid structures. The research is further motivated by a practical consulting issue where the vehicles could operate continuously, rendering pauses for refuelling unnecessary. Thus, unlike cases of energy-constrained routing problems [1], we emphasise the significance of spray refilling at the depot and aim to find an optimal routing plan for our problem. The current study will provide mathematical insights into the solution pattern.

4.1 Notation and Definitions

This section introduces notation and definitions that serve as a foundation for formulating several theorems concerning the regular grid graphs.

4.1.1 Notation

For the reader's convenience, a list of key symbols is provided below for quick reference. Some of these terms depend on concepts introduced later in the chapter and are also redefined progressively in context when first used. This structure aims to balance clarity with ease of reference. For a full list of symbols, refer to the List of Symbols.

List of Symbols

$\mathcal{RG}$ an undirected $K \times M$ regular grid graph, denoted by $\mathcal{RG} = \text{Reg}(K, M)$.

$N(i, j)$ nodes are on a grid, so are all identifiable by a vertical and horizontal coordinate, where $i \in \{1, 2, \cdots, K\}, j \in \{1, 2, \cdots, M\}$.

L the total number of layers in a regular grid graph, defined as $L = K - 1$. Each layer is indexed by i where $i \in \{1, 2, \ldots, K-1\}$. Specifically, L_i denotes the i-th layer and is given by $L_i = i$.

(L_i, j) the demand edge, which is the edge between $N(i, j)$ and $N(i + 1, j)$, where $i \in \{1, 2, \cdots, K-1\}$ and $j \in \{1, 2, \cdots, M\}$.

(i, H_j) the linking edge between nodes $N(i, j)$ and $N(i, j + 1)$, where $i \in \{1, 2, \ldots, K\}$ and $H_j = j$, $j \in \{1, 2, \ldots, M-1\}$. **The group of linking edges** is the set of all linking edges that share the same index j.

$D_{(L_i,j)}$ the demand of demand edge (L_i, j). In the regular grid graphs, the demand of each demand edge is the same. To simplify notation, we can represent the demand of any demand edge (L_i, j) as one unit. Therefore, we can use the notation $D_{(L_i,j)} = 1$ for regular grid graphs.

P capacity of the robotic sprayer. In regular grid graphs, this capacity is defined as the maximum number of demand edges that each robot can spray. In this chapter, P is an integer.

T the least common multiple of P and $2\,L$.

$\mathcal{R} = \{1, 2, \ldots, R\}$ the set of indices for the robotic sprayers' tours. R represents the minimum number of robots required to spray the orchard, typically given by $|\mathcal{R}| = R = \lceil \frac{ML}{P} \rceil$.

$C_{(L_i,j)}$ the cost of demand edge (L_i, j). In regular grid graphs, the cost of each demand edge is the same. To simplify notation, we can represent the cost of any demand edge (L_i, j) as C_d. Therefore, we can use the notation $C_{(L_i,j)} = C_d$ for regular grid graphs.

$C_{(i,H_j)}$ the cost of linking edge (i, H_j). In regular grid graphs, the cost of each linking edges in every group is identical. Therefore, we can use the notation $C_{(i,H_j)} = C_j$ for regular grid graphs.

m_r the number of demand edges sprayed by robot r.

A_r the cumulative cost of linking edges starting from the depot, where $A_r = \sum_{i=1}^{f_r-1} C_i$, and $f_r = \sum_{j=1}^{r} m_j$ when the solution is adjacent.

C_r^{robot} the cost incurred by robot r, which is equal to $2\sum_{i=1}^{f_r-1} C_i + \lceil \frac{m_r}{2} \rceil \times 2\, C_d$.

C_{total} the overall cost of a $OSRP_{\mathcal{RG}}$ solution, defined as $C_{total} = \sum_{r\in\mathcal{R}} C_r^{robot}$.

Decision Variables:

f_r the index of the demand edge that is furthest away from the depot, which is sprayed by robot r.

n_r the index of the demand edge that is nearest to the depot, which is sprayed by robot r.

I_r^l the subset of $\{1, 2, \ldots, M\}$ denoting the indices of the demand edges that the r-th robot sprays at the l-th layer.

n_r^l the index of the demand edge that is nearest to the depot at the l-th layer, which is sprayed by robot r. That is, $n_r^l = \min I_r^l$, and $n_r = \min_{l\in\{1,2,\ldots,L\}} n_r^l$.

f_r^l the index of the demand edge that is furthest to the depot at the l-th layer, which is sprayed by robot r. That is, $f_r^l = \max I_r^l$, and $f_r = \max_{l\in\{1,2,\ldots,L\}} f_r^l$.

In Section 4.2, we will sort the robots based on the values of f_r, so that $f_i \leq f_{i+1}$. Specifically, the furthest index of a demand edge sprayed by robot 1 is f_1, the furthest index of a demand edge sprayed by robot 2 is f_2, and so on for all robots in $\mathcal{R}$. This renaming of the robots will be based on the solution path pattern, which is explained in more detail in Section 4.2.

4.1.2 Definitions

The definitions provided below will be utilised throughout this chapter.

Definition 4.1.1. A **regular grid graph** $\mathcal{RG} = (K, M)$ is a grid graph such that the cost of each linking edges in every group is identical, the cost and demand of each demand edge is identical. When $\mathcal{RG}$ has only one layer, i.e., $L = 1$, it is called a **regular single-layer graph**. When $\mathcal{RG}$ has more than one layer, i.e., $L \geq 2$, it is called a **regular multi-layer graph**.

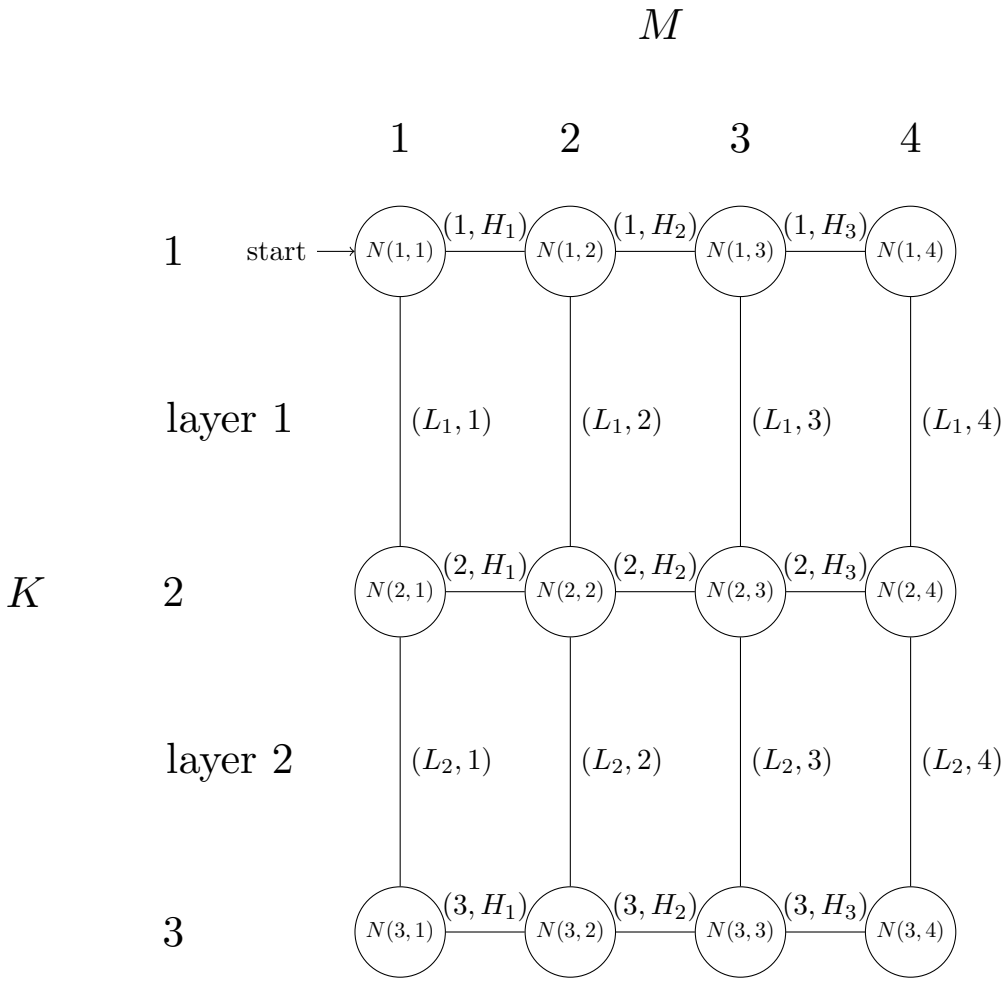


Figure 4.1: An example of a 3×4 multi-layer regular grid graph, where each vertex is uniquely identified by its vertical and horizontal indices, denoted as $N(i, j)$. The demand edges are denoted as (L_i, j), while the linking edges are denoted as (i, H_j).

Fig. 4.1 illustrates the definition and notation of a 3×4 regular grid graph with a depot located at $N(1, 1)$, identified as $\mathcal{RG} = (3, 4)$. This graph comprises linking edges, which connect grid points along the horizontal index between consecutive vertical indices. Demand edges, characterised by vertical edges within the grid graph, are also present.

Additionally, the graph demonstrates three groups of linking edges: $(1, H_1)$, $(2, H_1)$, and $(3, H_1)$; $(1, H_2)$, $(2, H_2)$, and $(3, H_2)$; and $(1, H_3)$, $(2, H_3)$, and $(3, H_3)$.

The current study focuses on addressing the OSRP on a regular grid graph, applied to agricultural routing with spraying. In this chapter, we apply Orchard Spraying Routing Problem on Regular Grid Graphs (OSRP_{RG}) to present the OSRP under a regular grid graph scenario, where all demand edges have *unit demand*, and the capacity P is an *integer multiple* of the unit demand, as defined in Definition 4.1.2.

Definition 4.1.2. $\text{OSRP}_{RG} = (K, M, C, P)$ defines an OSRP on an undirected **regular grid graph** $\mathcal{RG} = (K, M)$, where the cost of each edge is given by the vector C and the capacity of each vehicle is denoted by P. For the sake of simplifying the notation, the demand of any demand edge is assumed to be one unit, and the capacity P is an integer.

Proposition 4.1.1. If the capacity P is an integer in the $\text{OSRP}_{RG} = (K, M, C, P)$, then the vehicles are capable of servicing all edges without the need to split them among multiple vehicles.

As the capacity P is an integer multiple of the unit demand in the $\text{OSRP}_{RG} = (K, M, C, P)$, each demand edge can be satisfied by a single robot. It is not hard to see that any solution involving multiple robots spraying a demand edge is suboptimal. Hence in this study, there is no difference between CARP & SDCARP.

4.2 Characteristics of Solution Patterns for Regular Single-Layer Grid Graphs

This section focuses on analysing the theoretical structure of the optimal solution to determine the properties of routing paths on regular single-layer graphs. To simplify the problem and limit the range of potential solutions, we will arrange the sprayers in a specific order based on the index of the edge they spray that is furthest to the depot in the solution.

Assumption 4.2.1. The solution pattern will always be in the following order of the robotic sprayers.

$$f_1 \leq f_2 \leq \cdots \leq f_{|\mathcal{R}|}$$

The reordering of robots based on the furthest distance from the demand edge to the depot does not eliminate any optimal solution. This is because the total cost remains unchanged, and as a result, the problem remains unaffected by the renaming of robots.

Definition 4.2.1. Let $\text{OSRP}_{RG}= (2, M, C, P)$ be an OSRP on a regular single-layer grid graph. A solution S is a tuple $S = \big(S_1, S_2, \ldots, S_{|\mathcal{R}|}\big)$, where $S_r = \{n_r, \ldots, f_r\}$ represents the set of indices of demand edges sprayed by robot r. A **feasible solution** is a partition of the set of demand edge:

$$\left|\bigcup_{i\in\mathcal{R}} S_i\right| = M \quad \text{and} \quad S_i \cap S_j = \emptyset \;\forall\; i \neq j \in \mathcal{R}\,.$$

Definition 4.2.2. A feasible solution S is considered **adjacent** for an OSRP_{RG} on a regular single-layer grid graph if $f_r + 1 = n_{r+1}$..

An adjacent solution must satisfy that the first robot sprays the demand edges with indices from 1 to f_1, the second robot starts from $f_1 + 1$ and sprays demand edge indices from $f_1 + 1$ to $|\mathcal{R}|$. It is worth noting that $f_r = \sum_{i=1}^{r} m_i$, which represents the furthest index of demand edge for the i-th robot, equivalent to the cumulative count of demand edges sprayed from the first robot to the i-th robot in adjacent solutions.

Proposition 4.2.1. Given the indices of the edges that are sprayed by a robot in an adjacent solution, we can construct a **path pattern** as follows. First, find the furthest edge from the depot for the robot and connect it to the depot vertex. Next, follow a zigzag pattern to spray all the required edges. Finally, return to the depot.

Here, we present one path pattern as a list of nodes that satisfies the criteria outlined in Proposition 4.2.1. For $S_r = \{n_r, \ldots, f_r\}$, the list of nodes is:

- $N(1, i)$ for $i = 1, \ldots, f_r$ (travel to last demand edge)
- $N(1, k), N(2, k), N(2, k-1), N(1, k-1)$ for $k = f_r - 2i,\ 0 \leq i < \lfloor \frac{m_r}{2} \rfloor$ (zig-zag pattern)
- $N(1, n_r), N(2, n_r), N(1, n_r)$ if m_r is odd (duplicate demand edge)
- $N(1, i)$ for $i = n_r - 1, \ldots, 1$ (return to depot).

In Theorem 4.2.1 we show that this is optimal, and hence we call this an **optimal path pattern**. We provide an example and relevant details in Example 4.2.1 to further illustrate the statement of Proposition 4.2.1.

Example 4.2.1. Let $\text{OSRP}_{RG}= (2, M, C, P)$ denote an OSRP on a regular single-layer graph. We can construct the path patterns and determine the relevant associated costs for each robot based on the horizontal indices of the sprayed demand edges, as outlined in Proposition 4.2.1.

Fig. 4.2 shows an example of a regular single-layer grid graph $\mathcal{RG} = (2, 5)$ with the depot at $N(1, 1)$. If $P = 2$, then a solution for this problem involves three robotic sprayers, $S = (S_1, S_2, S_3)$, with $S_1 = \{1\}$, $S_2 = \{2, 3\}$, and $S_3 = \{4, 5\}$. A generated path for robot 1 is $[N(1, 1) \to N(2, 1)] \to N(1, 1)$, highlighted in orange with the sprayed demand edges denoted by square brackets. Similarly, the path patterns for robot 2 and robot 3 are highlighted in blue and brown in Fig. 4.2. In Theorem 4.2.1, we demonstrate that this solution is an optimal tour.

There are in general many symmetric solutions. In particular, it does not matter if the robot sprays while traveling away from the depot or on the return trip. In general, even a mix of the two options is possible. This symmetry allows easy construction of an optimal solution but can be expected to be problematic for any exact solution approach based on branch and bound.

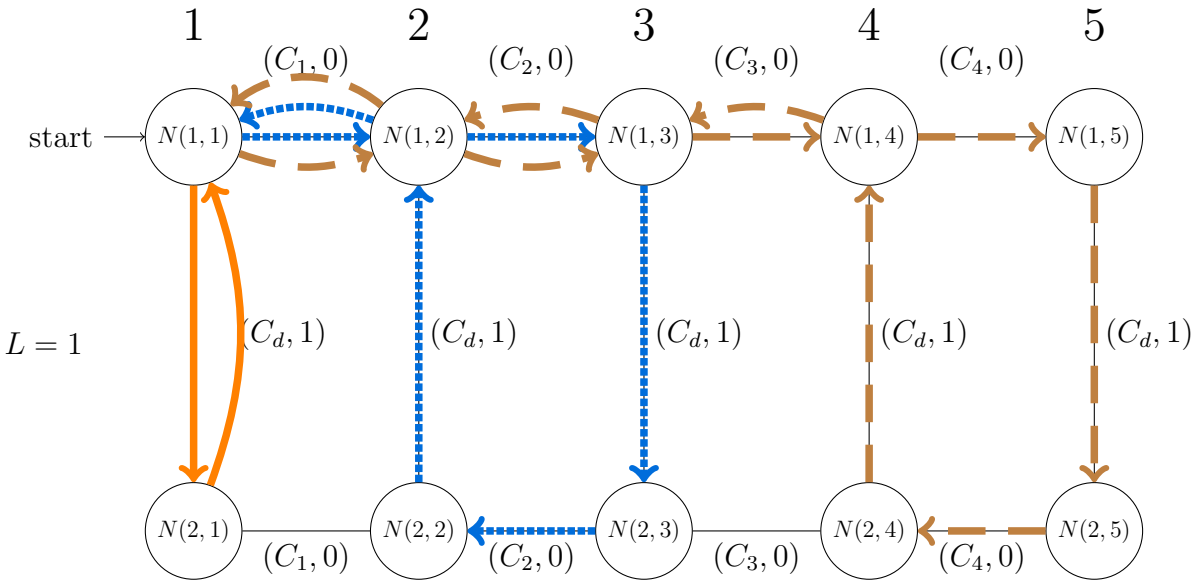


Figure 4.2: An instance of $\text{OSRP}_{RG}= (2, M, C, P)$ with depot $N(1, 1)$. C_i denote the cost of the linking edge, where $i \in \{1, 2, 3, 4\}$. The pair $(C_d, 1)$ represents the costs and demands of the demand edges. If $P = 2$, then a solution for this problem involves three robotic sprayers, $S = (S_1, S_2, S_3)$, with $S_1 = \{1\}$, $S_2 = \{2, 3\}$, and $S_3 = \{4, 5\}$.

In accordance with the path pattern detailed in Proposition 4.2.1, we have observed that the cost incurred by each robot is the aggregate of the costs associated with the linking edges and the demand edges. It is noteworthy that if a robotic sprayer covers an odd number of demand edges, duplicating the demand edge becomes imperative to ensure the sprayer's return to the depot for refilling. Conversely, in cases where an even number of edges are sprayed, duplication is unnecessary as it would result in an increased overall cost.

Therefore, the cost incurred by robot r, denoted as C_r^{robot}, can be computed using the formula (4.1), where C_i represents the cost of linking edges, C_d is the cost of demand edges, m_r is the number of demand edges sprayed by robot r. The overall cost of the problem denoted as C_{total}, can be expressed as Eq. (4.2).

$$C_r^{robot} = 2\sum_{i=1}^{f_r-1} C_i + \lceil \frac{m_r}{2} \rceil \times 2C_d \tag{4.1}$$

$$C_{total} = \sum_{r \in \mathcal{R}} C_r^{robot} \tag{4.2}$$

The variation in optimal solution path patterns has no impact on the overall cost. This suggests that the demand edge indices covered by each robot are adequate to determine both the solution path (as discussed in Proposition 4.2.1) and the cost.

Theorem 4.2.1. The **optimal solution** for the $\text{OSRP}_{RG}= (2, M, C, P)$ must be **adjacent**.

Proof. We prove this statement through **contradiction**.

Suppose there is an optimal solution in which robot r_i sprays non-adjacent demand edges. Let r_j be a robot that sprays at least one demand edge between n_{r_i} and f_{r_i}, the nearest and furthest index of sprayed demand edges for r_i respectively. There are four cases for the relationship between r_i and r_j: (1) $n_{r_i} < n_{r_j}$ and $f_{r_i} > f_{r_j}$; (2) $n_{r_i} > n_{r_j}$ and $f_{r_i} > f_{r_j}$; (3) $n_{r_i} < n_{r_j}$ and $f_{r_i} < f_{r_j}$; or (4) $n_{r_i} > n_{r_j}$ and $f_{r_i} < f_{r_j}$. We can perform a swap operation between rows n_{r_i} and f_{r_j} if $f_{r_i} > f_{r_j}$, otherwise swap rows n_{r_j} and f_{r_i}, as illustrated in Fig. 4.3. This swap operation, involving the exchange of n_{r_i} and f_{r_j}, means that the spray of demand edges corresponding to those indices is exchanged between the robots r_i and r_j. Based on Eq. (4.1) this results in a new solution that reduces the cost for robot r_j while keeping the cost of robot r_i unchanged, and vice versa.

The swapping operation described above results in a new solution that is less costly than the assumed optimal solution, which contradicts the optimality of the assumed solution. Therefore, we can conclude that the optimal solution must have adjacent rows for each robot. □

Based on the symmetry elimination described in Assumption 4.2.1, reordering the robotic sprayers does not alter the problem's properties. In Theorem 4.2.2, we clarify that the adjacent optimal solution path pattern, under Assumption 4.2.1, ensures that each robot sprays demand edges with increasing index numbers from the nearest depot to the furthest one. This property can be expressed as $m_r \leq m_{r+1}$.

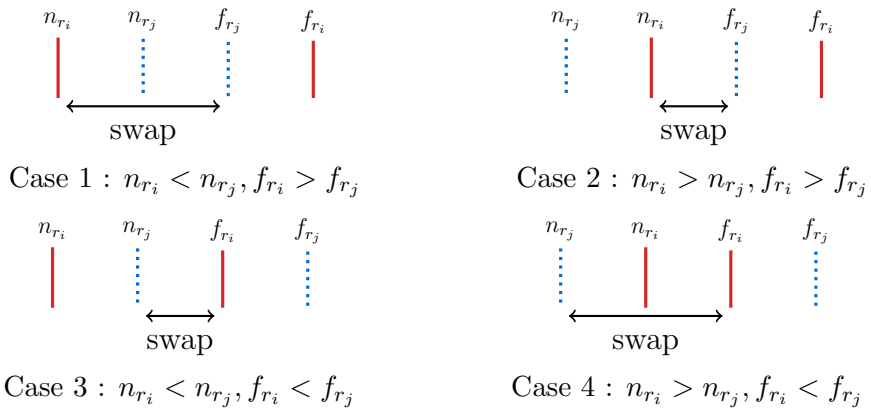


Figure 4.3: The swapping operation is depicted in the graph. The rows nearest to the depot sprayed by robots r_i and r_j are denoted by n_{r_i} and n_{r_j}, respectively. The corresponding furthest rows are f_{r_i} and f_{r_j}. In Cases 1 and 2, the rows n_{r_i} and f_{r_j} are swapped to obtain a new solution with a reduced cost for robot r_j and an unchanged cost for robot r_i. In Cases 3 and 4, the rows n_{r_j} and f_{r_i} are swapped to obtain a new solution with a reduced cost for robot r_i and an unchanged cost for robot r_j.

Theorem 4.2.2. In $\text{OSRP}_{RG}= (2, M, C, P)$, for any **optimal solution** the number of demand edges sprayed by each robot is ordered from **small to large**:

$$m_1 \leq m_2 \leq m_3 \leq \ldots \leq m_{|\mathcal{R}|}$$

Proof. According to Theorem 4.2.1, the optimal solution is adjacent. Let m_r be the number of demand edges sprayed by robot r. The cost of demand edges is a fixed number in this scenario. We prove the theorem through **contradiction** by focusing on the total cost of linking edges, denoted by $\hat{A}$.

By utilising Assumption 4.2.1, we can establish A_r as $\sum_{i=1}^{f_r-1} C_i$, representing the cumulative cost of linking edges from the depot to the furthest linking edge index of robot r, where $f_r = \sum_{i=1}^{r} m_i$ (as outlined in Definition 4.2.2). Then, we have $\hat{A} = 2\sum_{i=1}^{|\mathcal{R}|} A_i$.

Assume there exists a robot $i < |\mathcal{R}|$ with $m_i > m_{i+1}$. Construct a new solution by swapping m_i and m_{i+1}. In this new solution, $f'_r = f_r$ for all $r \neq i$, but $f'_i = f_i - m_i + m_{i+1} < f_i$. The linking costs remain unchanged: $A'_r = A_r$ for all $r \in \mathcal{R} \setminus \{i\}$. Thus,

$$\hat{A}' - \hat{A} = 2(A'_i - A_i) = 2\sum_{k=1}^{f'_i} C_k - 2\sum_{k=1}^{f_i} C_k = -2\sum_{k=f'_i+1}^{f_i} C_k < 0$$

Thus, it is demonstrated that the solution cost can be improved, thereby proving the theorem. □

We demonstrate this result using the example shown in Fig. 4.4. In this figure, a

robotic sprayer with a capacity of 3 units is depicted. An optimal solution can be achieved by assigning two robots to spray two consecutive rows and three consecutive rows, respectively, leading to $m_1 = 2$ and $m_2 = 3$. The cost of this configuration is $2\,(C_1 + C_1 + C_2 + C_3 + C_4) + 6\,C_d$, which is lower than the cost of an alternative configuration where $m_1 = 3$ and $m_2 = 2$, resulting in $2\,(C_1 + C_2 + C_1 + C_2 + C_3 + C_4) + 6\,C_d$.

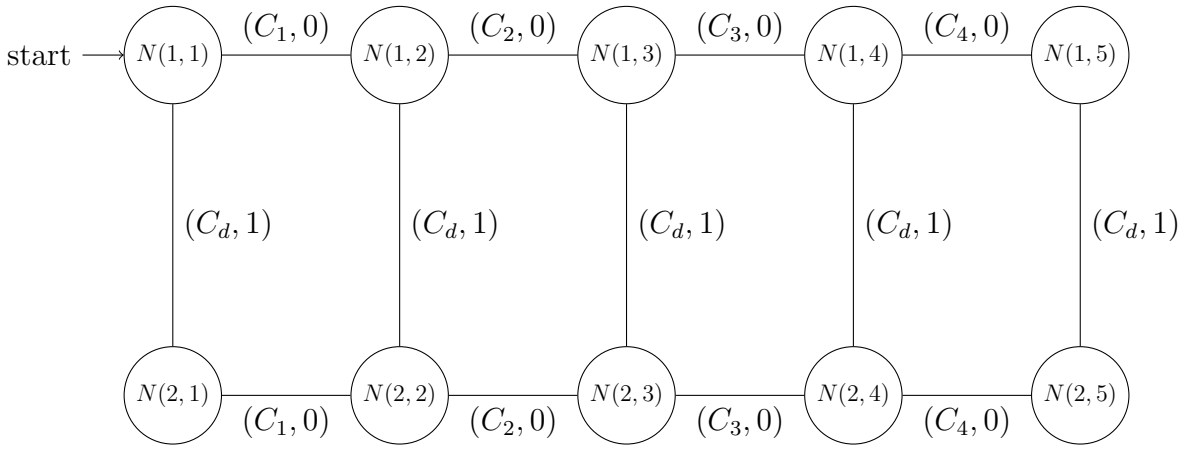


Figure 4.4: For a robotic capacity of 3 units, an optimal solution can be obtained by deploying two robots to spray two consecutive rows and three consecutive rows, respectively. This implies that $m_1 = 2$ and $m_2 = 3$. The cost of this configuration is $2\,(C_1 + C_1 + C_2 + C_3 + C_4) + 6\,C_d$, which is lower than the cost of the configuration with $m_1 = 3$ and $m_2 = 2$, which is $2\,(C_1 + C_2 + C_1 + C_2 + C_3 + C_4) + 6\,C_d$.

We will examine how choosing odd or even values for P and M influences the solution patterns. We will explore scenarios in Lemma 4.2.1, where P is an even number, irrespective of whether M is even or odd, along with cases outlined in Lemma 4.2.2 and Lemma 4.2.3, where P is an odd number.

The statement in Theorem 4.2.3 summarizes the findings of Lemma 4.2.1, Lemma 4.2.2, and Lemma 4.2.3. This theorem delineates the solution pattern for OSRP_{RG} on regular single-layer grid graphs. It's important to note that Lemma 4.2.1, Lemma 4.2.2, Lemma 4.2.3, and Theorem 4.2.3 specifically address scenarios where $M > P$. Otherwise, the optimal solution is to assign a single robot to spray all demand edges.

Lemma 4.2.1. For the $\text{OSRP}_{RG} = (2, M, C, P)$ with integer capacity P. If P **is an even number**, then the **adjacent optimal solution pattern** satisfies $m_1 = M - (|\mathcal{R}| - 1)P$ and $m_2 = m_3 = \cdots = m_{|\mathcal{R}|} = P$.

Proof. The overall cost is determined by two factors: the cost of linking edges and the cost of demand edges. If P and M are both even, every demand edge is traversed exactly once by any robot. By contrast, if M is an odd number, exactly one edge is traversed by more than one robot. In either case, the demand edge cost of demand edges is fixed, with

$M \times C_d$ for the even case and $(M+1) \times C_d$ for the odd case. Therefore, the challenge is to find the optimal route that minimizes the cost of linking edges, which is the main objective of the proof.

$m_1, m_2, m_3, \ldots, m_{|\mathcal{R}|}$, and let k be the largest index such that $m_k < P$ and $k > 1$. According to the order of m_r stated in Theorem 4.2.2, we have $m_1 \leq m_2 \leq m_3 \leq \cdots \leq m_k \leq \cdots \leq m_{|\mathcal{R}|}$.

We use A_r to denote the cost of linking edges for robot r in solution S. Specifically, we have $A_r = 2\sum_{i=1}^{f_r-1} C_i$, where $f_r = \sum_{i=1}^{r} m_r$. Let $\hat{A} = \sum_{r=1}^{|\mathcal{R}|} A_r$ represent the overall cost of linking edges.

If the value of M is odd, there will only be one odd m_r in the given context. Otherwise, if M is even, all m_r will be even, and there is no need to consider the odd case of m_r, which would result in duplicated cost on the demand edge.

Case 1: $\sum_{i=1}^{k} m_i < P$

In this case, we can construct a new solution S' by setting $m'_k = \sum_{i=1}^{k} m_i$, $m'_r = 0$ for $r < k$, and $m'_r = m_r$ for $r > k$. The cost of linking edges for robot r in S' is denoted by $A'_r = 2\sum_{i=1}^{f'_r-1} C_i$, where $f'_r = \sum_{i=1}^{r} m'_r$. It follows that the linking cost of S' is $\hat{A}' = \sum_{r\in\mathcal{R}} A'_r = \sum_{r=k}^{|\mathcal{R}|} A_r$, which is less than $\hat{A} = \sum_{r=1}^{|\mathcal{R}|} A_r$ of S. This contradicts the optimality of S.

Case 2: $\sum_{i=1}^{k} m_i \geq P$

We can construct a new solution S' by the following operations: First, set the value of m_k to P, and then decrease the values of $m_1, \ldots, m_{k-1}$, where some of these values may become zero. The new solution S' satisfies $0 \leq m'_r \leq m_r$ for $r < k$ and $m'_r = m_r = P$ for $r \geq k$. some of the values might be zero. Similarly, we use $A'_r = 2\sum_{i=1}^{f'_r} C_i$ to denote the cost of linking edges for robot r in S', where $f'_r = \sum_{i=1}^{r} m'_r$. The total cost of linking edges in S' is represented by $\hat{A}' = \sum_{r\in\mathcal{R}} A'_r$. Then we have:

$$
\begin{aligned}
f'_r &\leq f_r \quad \text{if } r < k-1 \\
f'_r &< f_r \quad \text{if } r = k-1 \\
f'_r &= f_r \quad \text{otherwise } (r \geq k).
\end{aligned}
$$

We get

$$\begin{aligned} A'_r &\leq A_r \quad \text{if } r < k-1 \\ A'_r &< A_r \quad \text{if } r = k-1 \\ A'_r &= A_r \quad \text{otherwise } (r \geq k). \end{aligned}$$

Thus

$$\hat{A}' = \sum_{r \in \mathcal{R}} A'_r < \sum_{r \in \mathcal{R}} A_r = \hat{A}$$

Hence, $\hat{A}' < \hat{A}$ $(k > 1)$, which contradicts our initial assumption that S is optimal. The process for obtaining the new solution S' is illustrated in Fig. 4.5.

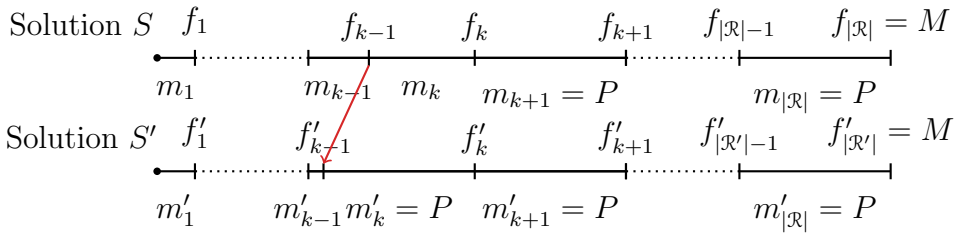


Figure 4.5: A scenario where P is even. Let S be an optimal solution such that $m_1 \leq m_2 \leq \cdots \leq m_k \leq \cdots \leq m_{|\mathcal{R}|}$, where k is the largest index such that $m_k < P$. If we modify S by setting $m'_k = P$, then we obtain a new solution S' where $f'_r \leq f_r$ for $r < k-1$, $f'_r < f_r$ for $r = k-1$, and $f'_k = f_k$ for $r \geq k$.

□

Lemma 4.2.2. For the OSRP$_{RG}$= $(2, M, C, P)$ with integer capacity $P < M$. If P **is an odd number** and $\sum_{i=1}^{M-1} C_i \leq C_d$, then the **optimal solution pattern** satisfies the following:

- If M is an even number, then $m_1 = M - (|\mathcal{R}| - 1)(P-1)$, $m_r = P - 1$ for $r > 1$.
- If M is an odd number, then $m_1 = M - P - (|\mathcal{R}| - 2)(P-1)$, $m_r = P - 1$ for $1 < r \leq |\mathcal{R}| - 1$, and $m_{|\mathcal{R}|} = P$

Proof. Given $\sum_{i=1}^{M-1} C_i \leq C_d$, then to get the lowest overall cost, a solution pattern should minimize instances where robots traverse demand edges multiple times.

Case 1: M is even.

If M is an even number, it is necessary to ensure that all m_r values are even numbers to avoid the increased cost of traversing demand edges twice. This situation can be seen

as the capacity equals $P-1$. According to Lemma 4.2.1, the adjacent optimal solution satisfies $m_1 = M - (|\mathcal{R}|-1)(P-1)$ and $m_2 = m_3 = \cdots = m_{|\mathcal{R}|} = P-1$.

Case 2: M is odd.

To avoid an increase in cost due to traversing demand edges twice, it is necessary to ensure that only one robot sprays an odd number of rows when the number of rows M is odd.

Prove by contradiction: Suppose there exists an adjacent optimal solution S with m_r for $r \in 1, 2, \ldots, |\mathcal{R}|$ and $m_{|\mathcal{R}|} < P$. From Theorem 4.2.2, we know that $m_1 \leq m_2 \leq \cdots \leq m_{|\mathcal{R}|}$.

If $\sum_{i=1}^{|\mathcal{R}|} < P$, then the optimal solution is to have only one robot spray all the demand edges, which is similar to the case for even M in Lemma 4.2.1. Now, let us consider the case when $\sum_{i=1}^{|\mathcal{R}|} \geq P$. We can set $m_{|\mathcal{R}|} = P$ and decrease the values of $m_{|\mathcal{R}|-1}$, $m_{|\mathcal{R}|-2}$, and so on until m_1 to obtain a new solution S' with $m'_{|\mathcal{R}|} = P$, $0 \leq m'_r \leq m_r$ for $r < k$ (labelled from right to left as $m'_{|\mathcal{R}|-1}$, $m'_{|\mathcal{R}|-2}$ until m'_1, the m'_i can be zero). As shown in Lemma 4.2.1, the total cost of S' is less than that of S, which contradicts the optimality of S. Therefore, the optimal solution must satisfy $m_{|\mathcal{R}|} = P$.

If $m_{|\mathcal{R}|} = P$ is an odd number, then $M - P$ is an even number. This is similar to the case for even M and odd P in case 1. Hence, we have proven the lemma. □

As shown in Lemma 4.2.2, the result is valid when the overall one-way cost of linking edges, $\sum_{i=1}^{M-1} C_i$, is less than or equal to the cost of each demand edge, C_d. In Lemma 4.2.3, we consider a situation where there exists a demand edge index j such that the sum of the costs of the linking edges, $\sum_{i=1}^{j-1} C_i$, is greater than the cost of a demand edge C_d. In such cases, we need to take into account both the linking cost and the cost of duplicating demand edges to find an optimal solution.

Lemma 4.2.3. For $\text{OSRP}_{RG} = (2, M, C, P)$ where P is an odd integer with $P < M$, in every optimal solution, there is a robot index k such that $m_1 = M - (\sum_{r=2}^{|\mathcal{R}|} m_r)$, $m_r = P-1$ for $2 < r \leq k$, and $m_r = P$ for $r > k$.

Proof. In Lemma 4.2.2, it was previously established that when the total one-way cost of linking edges, represented by $\sum_{i=1}^{M-1} C_i$, is no greater than the cost of demand edges, C_d, the lemma holds with $k = |\mathcal{R}|$ if M is even and $k = |\mathcal{R}| - 1$ otherwise. However, in situations where the sum of the costs of edges before M is greater than C_d, the cost of linking edges becomes greater than that of demand edges. To determine the optimal solution in such scenarios, it is necessary to compare the costs of linking edges with those of duplicating demand edges in order to identify the optimal solution pattern.

When $\sum_{i=1}^{M-1} C_i \geq C_d$, the optimal solution for the case where $\sum_{i=1}^{M-1} C_i \leq C_d$ can be viewed as an upper-bound for this problem. Theorem 4.2.2 ensures that $m_1 \leq m_2 \leq \cdots \leq m_{|\mathcal{R}|}$. According to the proof of Lemma 4.2.2, the robots should spray as many demand edges as possible. Otherwise, increasing m_k and decreasing m_r for $r < k$ can lead to a better solution. However, when P is odd, a dilemma arises regarding whether the robots should spray P demand edges, which may lead to the duplication of demand edges, or $P-1$ edges without duplicating any demand edges, which may increase the total cost of linking edges. We can conclude that robots $m_2, m_3, \cdots, m_{|\mathcal{R}|}$ have only two options to choose from: $P-1$ and P (m_1 equals the number of remaining demand edges). We only need to select the robot number k after which all remaining robots spray P edges. Thus, there exists a robot index k such that $m_1 = M - (\sum_{r=2}^{|\mathcal{R}|} m_r)$, $m_r = P-1$ for $2 < r \leq k$, and $m_r = P$ for $r > k$. □

It should be noted that the objective as a function of k is **non-convex** (as shown in Fig. 4.6, where $P = 5$ and $M = 200$ are fixed for randomly generated instances.) and there is no obvious closed-form solution for the optimal k based on C_i's, M and P. However, as M is a finite integer, the number of robots is also finite. Therefore, we can select the optimal value of k by testing each possibility in polynomial time. Example 4.2.2

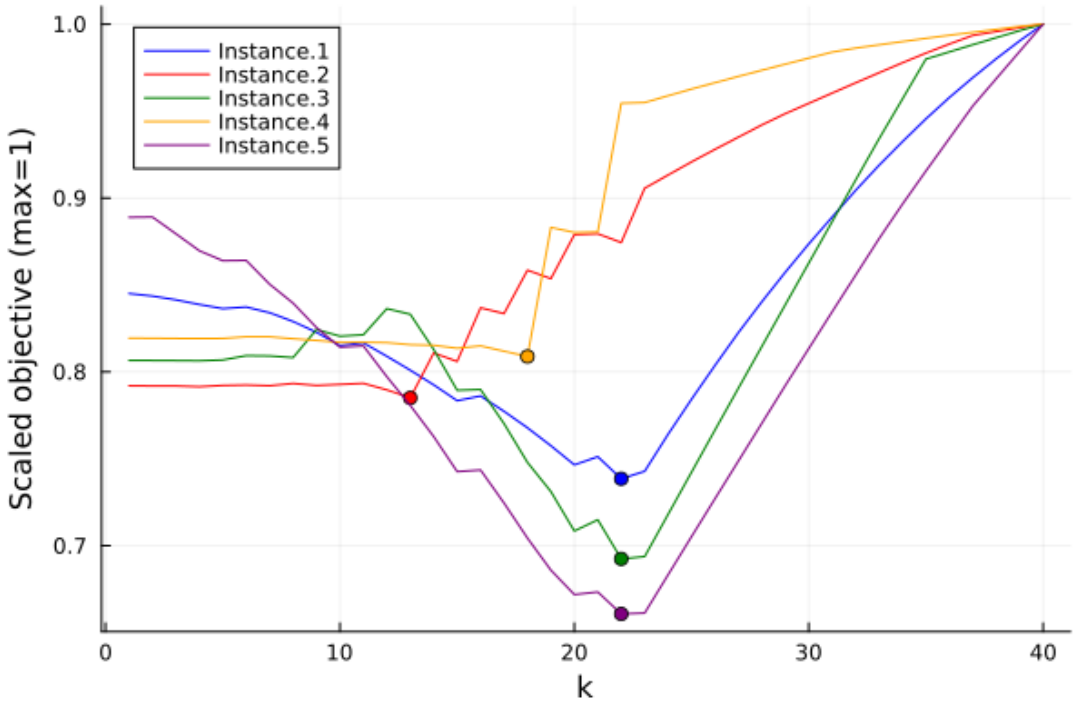


Figure 4.6: Five random datasets show that the function of k is non-convex. For these instances, we set P =5 and M =200.

illustrates the importance of comparing the costs of linking edges and demand edges to find

an optimal solution when P is an odd integer capacity and there exists a j $(1 \leq j \leq M)$ that is the minimum demand edge index for which the cost of the edges before demand edge j is greater than or equal to the cost of demand edge C_d (i.e., $\sum_{i=1}^{j-1} C_i \geq C_d$). It is important to note that in general, j is not the main factor that determines the turning point at which the robot needs to stop spraying due to a lack of capacity.

Example 4.2.2. Let's consider an example of $\text{OSRP}_{RG} = (2, 12, C, 5)$ where we need to spray 12 demand edges. According to the statement in Lemma 4.2.2, the optimal solution pattern should be one of the following:

(1) S_1: $m_1 = 4, m_2 = 4, m_3 = 4$

(2) S_2: $m_1 = 3, m_2 = 4, m_3 = 5$

(3) S_3: $m_1 = 2, m_2 = 5, m_3 = 5$

Based on the cost comparison presented in Table 4.1, we can conclude that:

(1) S_1 is optimal if $C_2 + C_3 + C_7 < C_d$

(2) S_2 cannot be optimal since the cost of S_2 is greater than that of S_3

(3) S_3 is optimal if $C_2 + C_3 + C_7 > C_d$

If $\sum_{i=1}^{j-1} C_i \geq C_d$, the optimal solution depends on the values of C_i for $i = 2, 3, 7$. When $C_2 + C_3 + C_7 < C_d$, S_1 becomes the optimal solution; otherwise, S_3 is the optimal choice.

Therefore, when P is odd and a minimum demand edge index j $(1 \leq j \leq M)$ is identified, where the cumulative cost of edges up to demand edge j is greater than or equal to the cost of demand edge C_d (i.e., $\sum_{i=1}^{j-1} C_i \geq C_d$), comparing the costs of linking edges and demand edges is necessary to determine the optimal value of k.

Table 4.1: Comparison of the cost for different solution patterns

	f_1	f_2	f_3	Cost of demand edges
S_1	4	8	12	$12\,C_d$
Compare with S_3	$+\,2\,(C_2 + C_3)$	$+2\;C_7$	same	no duplication
S_2	3	7	12	$14\,C_d$
Compare with S_3	$+\,2\,C_2$	same	same	2 duplications
S_3	2	7	12	$14\,C_d$
-	$2\,C_1$	$2\sum_{i=1}^{6} C_i$	$2\sum_{i=1}^{11} C_i$	2 duplications

Theorem 4.2.3. For $OSRP_{\mathcal{RG}} = (2, M, C, P)$ with integer capacity P, there exists $k \in \mathcal{R}$ that defines an adjacent optimal solution pattern where each robot r sprays m_r edges, with

$$m_r = \begin{cases} P & \text{if } r > k \\ P - 1 & \text{if } 2 \leq r \leq k \\ M - \sum_{r=2}^{|\mathcal{R}|} m_r & \text{for } r = 1 \end{cases}.$$

Furthermore, if P is even then $k = 1$ so that there are no robots that spray $P - 1$ edges.

Proof. As shown in Lemma 4.2.1, Lemma 4.2.2, and Lemma 4.2.3. □

In this Section, we examined the properties of solution patterns for **regular single-layer grid graphs**. We will focus on **regular multi-layer grid graphs** and explore the features of solution patterns in Section 4.3.

4.3 Characteristics of Solution Patterns for Regular Multi-Layer Grid Graphs

In contrast to the previous section where we analyzed solution patterns for regular single-layer grid graphs, we will now shift our attention to regular multi-layer grid graphs and investigate the characteristics of solution patterns in this setting. Unlike the single-layer problem, the multi-layer problem requires consideration of multiple layers, as well as more demand edges and linking edges, making it a more complex problem.

Assumption 4.3.1. To begin our exploration of solution patterns for regular multi-layer grid graphs, we will first focus on cases where P is even and at least twice the number of layers, denoted as $P \geq 2\,L$.

The Assumption 4.3.1 ensure that robots can always spray pairs of demand edges and complete spraying of at least one pair of demand edges across all layers of the orchard (i.e., from top to bottom in our grid graphs). All the results regarding the characteristics of multiple-layer problems in this section are under the Assumption 4.3.1. Later, in the Section 4.4, we will present cases that deviate from the assumptions stated in Assumption 4.3.1.

Lemma 4.3.1. In the context of $\text{OSRP}_{RG} = (K, M, C, P)$ with an even $P \geq 2L$, where $L = K - 1$ represents the number of layers, a **lower bound** (LB) for the problem can

be calculated as follows:

$$\text{Let} \quad \hat{f}_r = M - 2\lfloor\frac{(|\mathcal{R}|-r)P}{2L}\rfloor \quad \forall r \in \mathcal{R}$$

$$\text{and} \quad A_r = 2\sum_{i=1}^{\hat{f}_r-1} C_i$$

$$\text{then} \quad LB = \sum_{r\in\mathcal{R}} A_r + \begin{cases} (LM)\,C_d & \text{if } M \text{ is even} \\ (LM+L)\,C_d & \text{if } M \text{ is odd} \end{cases}$$

Proof. There are $M \times L$ demand edges, and the furthest demand edge index of the last robot is M. Since the solution pattern should avoid passing demand edges without spraying work, the robots should spray an even number of demand edges in each layer.

Then, the furthest index of the robots satisfies: (the robots are ordered according to the index of the furthest edge, as mentioned in Assumption 4.2.1).

$$\begin{aligned} f_{|\mathcal{R}|} &= M \\ f_{|\mathcal{R}|-1} &\geq M - 2\lfloor\frac{P}{2L}\rfloor \\ f_{|\mathcal{R}|-2} &\geq M - 2\lfloor\frac{2P}{2L}\rfloor \\ &\vdots \\ f_{|\mathcal{R}|-k} &\geq M - 2\lfloor\frac{kP}{2L}\rfloor \\ &\vdots \\ f_1 &\geq M - 2\lfloor\frac{(|\mathcal{R}|-1)P}{2L}\rfloor \end{aligned}$$

Then, the furthest index of robots in any optimal solution satisfies

$$f_r^* \geq M - 2\lfloor\frac{(|\mathcal{R}|-r)P}{2L}\rfloor \implies f_r^* \geq \hat{f}_r$$

Specifically, when P is divisible by $2L$, there is no surplus left for any robot $r > 1$. Then

$$f_r^* \geq M - 2\,(|\mathcal{R}|-r)\frac{P}{2L} \implies f_r^* \geq \hat{f}_r$$

The total travel cost consists of the cost of travelling the linking edges (A_r^*) and the cost of demand edges. Given the upper bound on f_r^* we can compute a corresponding lower

bound on the travel cost of linking edges:

$$A_r^* = 2\sum_{i=1}^{f_r^*-1} C_i \geq 2\sum_{i=1}^{\hat{f}_r-1} C_i = A_r$$

If M is even, it is not necessary to duplicate any demand edges, hence with $L\,M$ demand edges, the total demand edge travel cost is simply bounded by $L\,M\,C_d$. However for odd M at least one demand edge per layer must be duplicated, giving a lower bound $L\,(M+1)\,C_d$. This gives the required result:

$$\text{Optimal cost} \geq \sum_{r\in\mathcal{R}} A_r + \begin{cases} (L\,M)\,C_d & \text{if } M \text{ is even} \\ (L\,M+L)\,C_d & \text{if } M \text{ is odd} \end{cases} = LB$$

□

Next, we will define a **block adjacent** solution for a multi-layer problem, which is analogous to the adjacent solution described for the single-layer problem.

Definition 4.3.1. Consider $\text{OSRP}_{RG} = (K, M, C, P)$. A **solution** S is defined as the tuple $\left(S_1, S_2, \ldots, S_{|\mathcal{R}|}\right)$, where $S_r = \{I_r^l \mid l = 1, \ldots, L\}$.

Definition 4.3.2. Consider a solution S for OSRP_{RG}= (K, M, C, P). The **robot is block adjacent** if I_r^l is adjacent, and there are $j, k \in \{1, \ldots, L\}$:

$$n_r^l = \begin{cases} n_r^1 & \text{if } l \leq j \\ n_r^1 + 2 & \text{if } j < l < L \end{cases} \quad \text{and} \quad f_r^l = \begin{cases} f_r^1 & \text{if } l \leq k \\ f_r^1 + 2 & \text{if } k < l < L \end{cases}$$

A solution $\left(S_1, S_2, \ldots, S_{|\mathcal{R}|}\right)$ is **block adjacent** if every robot is block adjacent.

An example illustrating the path pattern for a block adjacent solution is shown in Fig. 4.7. Notably, n_r is the nearest index of the demand edges among all of the layers, which is defined as $n_r = \min_{l\in\{1,\ldots,L\}} n_r^l$. For a block adjacent solution, $n_r = n_r^1$, and similarly, $f_r = f_r^1$. In constructing the solution path for each robot, information on how far (in layers) the robot needs to spray is required, as described in Proposition 4.3.1, where $a_r = j, b_r = L - k$. The construction of the solution path in Proposition 4.3.1 is a block adjacent solution, satisfying Definition 4.3.2. The intuition behind the block adjacent solution is that the solution should maintain adjacency across different layers for each robot, with each robot spraying demand edge pairs across layers from left to right without leaving any unsprayed left corners. This approach avoids the duplication of linking edges and demand edges.

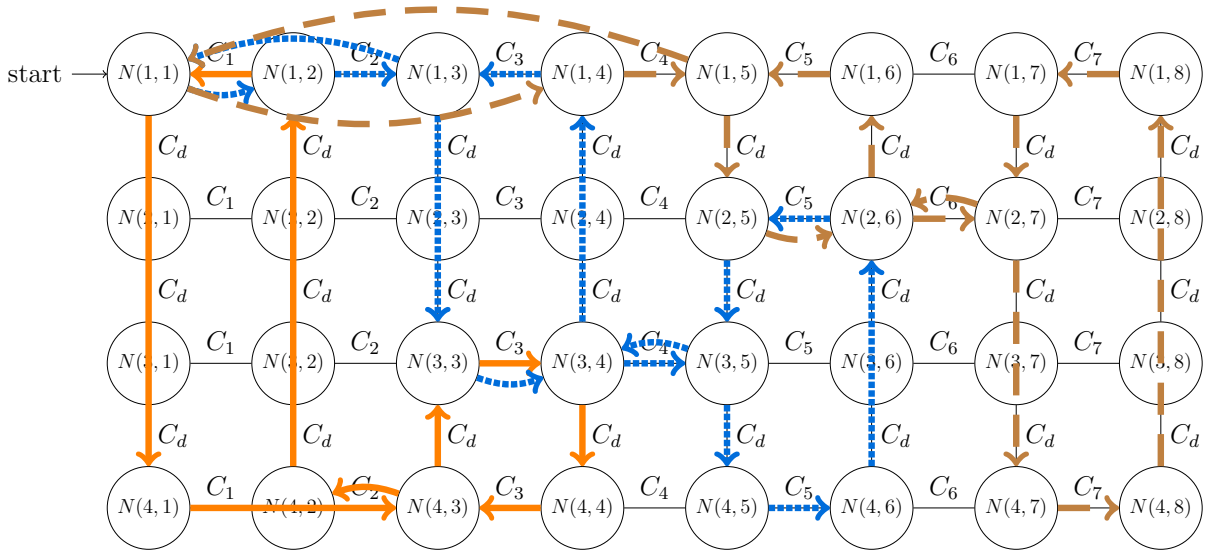


Figure 4.7: An example illustrating the **block adjacent** solution pattern with $M = 8$, $P = 8$, L= 3. In this instance, three robots are required, and the relevant solution patterns are highlighted in different colors. The path pattern for robot 1 is highlighted with red lines, showing that I_1^l is adjacent at each layer. Specifically, $I_1^1 = \{1, 2\}$, $I_1^2 = \{1, 2\}$, $I_1^3 = \{1, 2, 3, 4\}$, with $j = 3$ and $k = 2$, where $n_r^1 = 1$ and $f_r^1 = 4$. Similarly, robots 2 and 3 are block adjacent. Thus, the solution satisfies the definition of block adjacent in Definition 4.3.2, which is a block adjacent solution.

Proposition 4.3.1. Given four pieces of data for a robot r, denoted as n_r and f_r which indicate the nearest and furthest indices of demand edges respectively, and a_r and b_r representing the number of layers that robots with $r > 1$ can spray from top to bottom and bottom to top (as illustrated in Fig. 4.8), respectively. Then the solution path could be constructed as follows:

(1) Start from the depot and move to the top of demand edge n_r

(2) Spray demand edges up to layer a_r.

(3) Travel across the linking edge to the demand edge's index $n_r + 2$ and spray demand edges up to layer L.

(4) Cross the linking edge to demand edge's index f_r and spray demand edges down to the b_r's step to reach (but not including) layer $L - b_r$.

(5) Finally, return by travelling back one link edge at a time. While returning, spray demand edges in increasing layer numbers up to the layer of the previously travelled link edge for odd-numbered rows, and spray demand edges in decreasing layer numbers up to and including Layer 1 for even-numbered rows.

(6) Continue this process until completing demand edge $n_r + 1$, and then return directly to the depot.

For robot 1, a_1 must be L. If not, a better solution can be obtained by swapping the demand rows between robots. This indicates that robot 1 is already positioned at layer L. Consequently, we can skip step 3 and proceed directly to step 4 for robot 1.

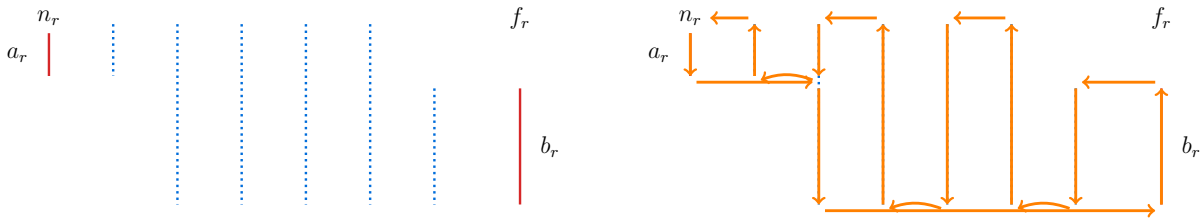


Figure 4.8: Path construction with four pieces of data: n_r, a_r, b_r, and f_r.

Note that:

(1) Each robot r travels up to demand edge $f_r = M - 2\lfloor \frac{kP}{2L} \rfloor = M - \lfloor \frac{(|\Re|-r)P}{2L} \rfloor$ with each linking edge traversed twice so $A_r = \sum_{i=1}^{f_r-1} C_i$ as per Lemma 4.3.1.

(2) Each demand edge is traversed once only for even M, and one robot traverses L demand edges twice for odd M.

(3) The pattern relies on $P \geq 2L$ to ensure that the robot can travel up to layer L and back again.

(4) Given that T represents the least common multiple of P and $2L$, if $T = M\,L$ then for $r = 1$ all of the spray is used, otherwise robot 1 has some spray left.

Notably, T is the least common multiple of P and $2\,L$, where $M\,L$ represents the total number of demand edges. If $T = M\,L$, it indicates that it is possible to construct a path pattern that fully utilizes all the capacity without leaving any unsprayed edges. However, if $T \neq M\,L$, it implies there are remaining capacities, complicating the problem.

In general, many symmetric solutions with the same costs exist. Specifically, it does not matter whether the robot sprays while traveling away from the depot or on the return trip. A mix of the two options is also possible. This symmetry allows for the easy construction of a solution but can be problematic for any exact solution approach based on branch and bound.

The technique for generating solution patterns characterized by block adjacency is described in Definition 4.3.2. Next, we will show the optimality in both scenarios described in Lemma 4.3.2 and Lemma 4.3.3 of the block adjacent solution.

Proposition 4.3.2. The calculation of n_r, a_r, b_r, and f_r can be achieved using the following rules, when the condition $M\,L \leq T$ holds.

$$1.\ f_r \leftarrow M - 2\left\lfloor \frac{kP}{2L} \right\rfloor \qquad \forall\ r \in \mathcal{R};\ k = |\mathcal{R}| - r$$

$$2.\ a_r \leftarrow \begin{cases} L & \text{if } r = 1 \\ L & \text{if } f_{r-1} \leq 2(r-1) \\ \frac{(k+1)P \pmod{2L}}{2} & \text{otherwise} \end{cases} \qquad \forall\ r \in \mathcal{R};\ k = |\mathcal{R}| - r$$

$$3.\ n_r \leftarrow \begin{cases} 1 & \text{if } r = 1 \\ f_{r-1} + 1 & \text{if } a_r = L \\ f_{r-1} - 1 & \text{otherwise} \end{cases} \qquad \forall\ r \in \mathcal{R}$$

$$4.\ b_r \leftarrow \begin{cases} L & \text{if } r = \mathcal{R} \text{ or } a_{r+1} = L \\ L - a_{r+1} & \text{otherwise} \end{cases} \qquad \forall\ r \in \mathcal{R}$$

The robot, while avoiding duplications, follows the strategy outlined in Fig. 4.8 to optimise its spraying efficiency. It strategically maximizes spraying and ensures pairs of demand edges are treated for a return to the depot without duplication in even M cases, or with duplication of L demand edges for odd M cases. This begins with spraying the final demand edge, followed by a return to the depot once capacity is used up. Subsequent robots cover the remaining area of the previous robot, continuing sequentially until the entire spraying task is finished.

For even M, if robot 1 has sprayed a minimum of fewer than $2\,L$ demand edges, then direct robot 1 to spray the initial two complete vertical edges, totalling $2\,L$. Similarly, reduce the count of demand edges sprayed by robot 2 and confirm if it's adjusted to or exceeds $2\,L$. If not, instruct robot 2 to spray the subsequent two complete demand edges, reaching $2\,L$ in total (shown in Fig. 4.9). Repeat this process for successive robots until the final one.

Lemmas 4.3.2 and 4.3.3 illustrate two cases regarding the properties of the existence of a *block adjacent solution*. According to our assumption stated in 4.3.1, where $P \geq 2\,L$, Lemma 4.3.2 addresses the case when $P = 2\,L$ and all scenarios where P is divisible by $2\,L$. In contrast, Lemma 4.3.3 discusses the cases when P is even and $P > 2\,L$.

Lemma 4.3.2. For OSRP_{RG} where P is divisible by $2\,L$, there **exists a block adjacent** optimal solution.

Proof. We will demonstrate this by identifying a solution pattern that satisfies the lower

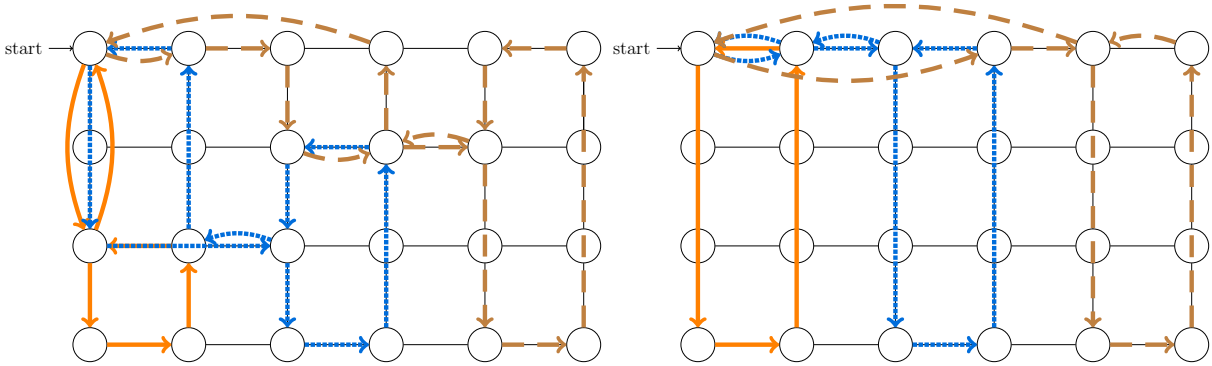


Figure 4.9: This example illustrates how robot 1's inspection can effectively prevent unnecessary travel across demand edges. We consider the parameters $M = 6$, $P = 8$, and $L = 3$. The graph depicts three robots forming a **block adjacent** solution pattern on the left side without verifying the first robot's path (as demonstrated in the proof of Lemma 4.3.3). On the right side, we inspect and adjust each robot's capacity, resulting in a reduction in the duplication of demand edges' cost.

bound defined in Lemma 4.3.1, thereby ensuring the optimality of the solution. It is worth noting that $0 \equiv P \pmod{2\,L}$ implies that $\frac{P}{2\,L}$ is a positive integer. Therefore, the robot evenly applies spray along the entire length of demand edges, leaving no excess. Hence we can construct a solution corresponding to the single layer problem $(2, M, C, P)$ as described in Lemma 4.2.1. Here

$$f_r = M - 2\,(|\mathcal{R}| - r)\frac{P}{2L} = f_r^l \quad \forall\, r \in \mathcal{R},\ l \in \{1, \ldots, L\}$$

Every demand edge is traversed once except if M is odd, where one demand edge must be traversed twice in each layer.

Therefore, the solution is equal to the lower bound described in Lemma 4.3.1. An example illustrating the transformation of a two-layer grid graph into a single-layer graph is provided in Fig. 4.10.

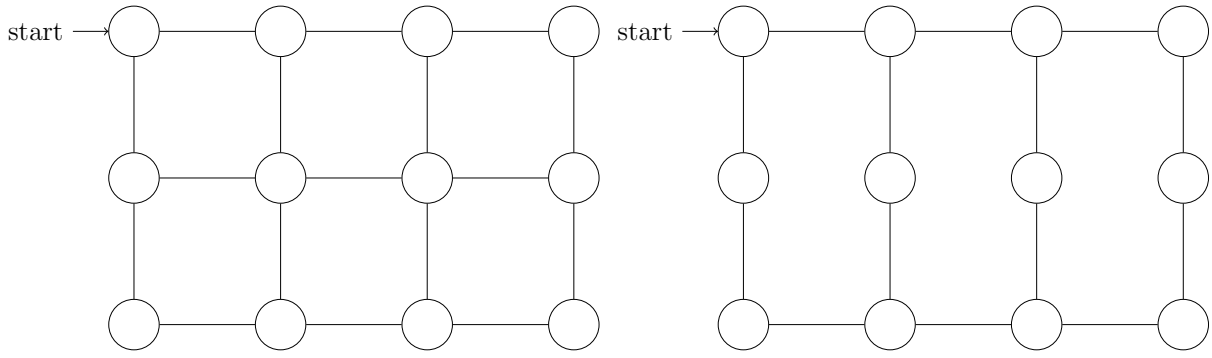


Figure 4.10: An example showcasing the transformation of a two-layer problem into a single-layer problem.

□

Lemma 4.3.3. In OSRP_{RG} where P is even and $P > 2\,L$, there exists an **optimal block adjacent** solution.

Proof. If $M\,L \leq T$, the pattern described in Proposition 4.3.1 and Fig. 4.8 indicates that each robot r travels up to demand edge $f_r = M - 2\lfloor \frac{kP}{2L} \rfloor = M - \lfloor \frac{(|\Re|-r)P}{2\,L} \rfloor$, with each linking edge traversed twice. Thus, $A_r = \sum_{i=1}^{f_r-1} C_i$ as shown in Lemma 4.3.1, resulting in a total cost equal to the lower bound presented in Lemma 4.3.1. This demonstration confirms the optimality of the solution. The situation in which P is divisible by $2\,L$ is addressed by the Lemma 4.3.2.

If $M\,L > T$, then T/P robots can be utilised to spray the last $T/2\,L$ rows, after which the pattern repeats. This approach also provides a solution that matches the lower bound presented in Lemma 4.3.1. □

4.4 Instances Where A Claimed property or Relationship Does Not Hold

In this section, we present some counterexamples for cases where no adjacent optimal solution exists, contrary to the assumptions stated in Assumption 4.3.1.

Counterexample 4.4.1. Different set of linking edges' cost in a single-layer problem.

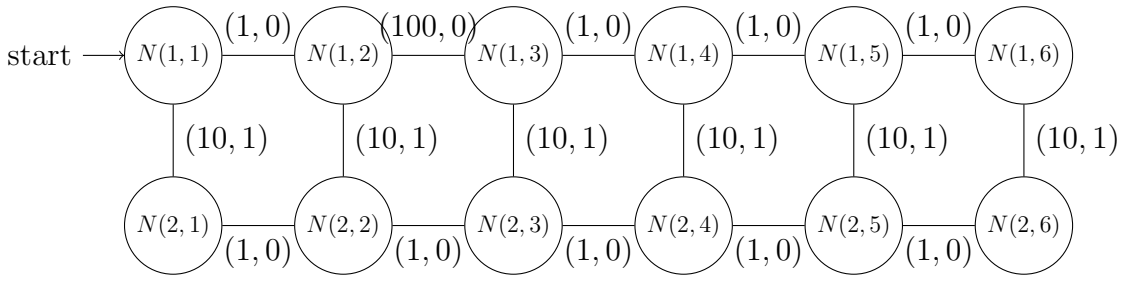


Figure 4.11: The numbers along each edge are the cost and demand respectively.

In the scenario described in Counterexample 4.4.1, where $P = 3$ and $M = 6$, the only way to avoid the excessive cost (over 100) is to avoid the linking edge (1,2)-(1,3) by using (2,2)-(2,3) instead. This results in:

(1) A non-adjacent solution where robot 1 sprays demand edges $\{1, 3, 4\}$ and robot 2 sprays $\{2, 5, 6\}$. The total cost is 96.

(2) An adjacent solution involving three robots spraying demand edges $\{1, 2\}$, $\{3, 4\}$ and $\{5, 6\}$, respectively. In this case, both robots 2 and 3 traverse an additional demand edge (say demand edge 1) twice to avoid the expensive linking edge. This increases the total cost to 98 (80 for demand edges and 18 for linking edges).

(3) An even more expensive solution with two robots: robot 1 sprays $\{1, 2, 3\}$ while robot 2 sprays $\{4, 5, 6\}$ with two traversals of demand edges for each robot taking the total cost to over 100.

Clearly, none of the adjacent solutions are optimal in this case. Hence, if the set of linking edge costs differs, the optimal solution may not be adjacent.

It is worth noting that the adjacent solution result would hold for **non-equal demand edge costs**, with just a minor modification: allowing robots to choose the shortest path whenever they travel empty.

Counterexample 4.4.2. The rows' demands are not equal

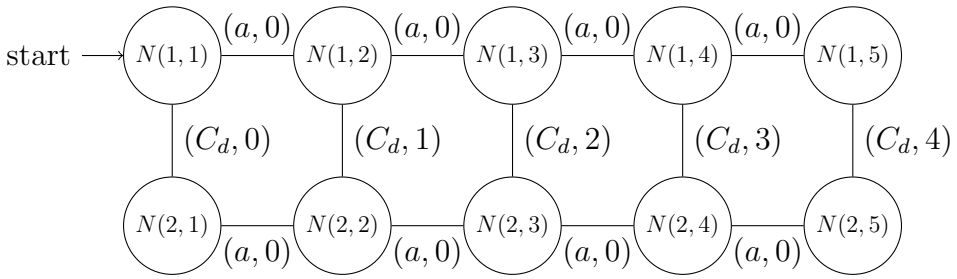


Figure 4.12: Demand edge cost C_d and linking cost a, with unequal demands.

In the case of Counterexample 4.4.2, where $P = 5$, the optimal solution is as follows: robot 1 sprays demand edges $\{2, 5\}$ while robot 2 sprays $\{3, 4\}$, resulting in a total cost of $14\,a + 4\,C_d$, which is lower than that of any adjacent solutions.

We will analyze the complexity of this problem by reduction from the 3-Partition problem. The decision version of the 3-Partition problem is as follows: given a list of $3\,k$ integers $n_1, n_2, \ldots, n_{3\,k}$ and integer T with $\sum_{i=1}^{3\,k} n_i = kT$, is it possible to partition the numbers into k groups of 3 integers such that the sum of each group is exactly T? This problem is NP-complete in the strong sense even if $T/4 < n_i < T/2$, so that the only subsets of the numbers that sum to T have exactly 3 elements [28].

Theorem 4.4.1. The OSRP_{RG} is **strongly NP-complete** when distinct demand values are allowed.

Proof. Consider an instance of the 3-Partition problem with $3\,k$ numbers n_i summing to kT. This can be mapped to a OSRP_{RG} instance as follows. Create a single-layer grid with

$3\,k+1$ demand edges, $d_i = n_i$ and additional demand edge $d_0 = 0$. Let $P = T$ with $|\mathcal{R}| = k$ robots. All linking edges and demand edge 0 have cost zero, with all other demand edges have cost 1. Clearly if the 3-Partition problem has a solution, then there is an equivalent OSRP_{RG} solution with each robot spraying exactly 3 demand edges and travelling back via demand edge 0 for a total cost of $3\,k$. Conversely, the only way OSRP_{RG} can have a solution with cost at most $3\,k$, assuming $P/4 < d_i < P/2$ and $\sum_i d_i = k\,P$, is to have all robots use their full capacity to spray 3 demand edges each. Hence OSRP_{RG} solves the decision version of the 3-Partition problem. Note that we could avoid the use of a demand edge with zero cost (i.e. have all demand edges have unit cost) by adding k additional demand edges and modifying demands slightly to ensure each robot sprays exactly 4 edges, but that makes the notation and explanation more complex without providing additional insight. □

Note that the same proof idea can be extended to some other variants of this problem. For example, the problem with just two robots is already weakly NP-complete by reduction from the subset sum problem (splitting a collection of numbers into two subsets with equal sum). Similarly a variant of OSRP_{RG} with an unrestricted number of robots (number of trips by a single robot), can be treated in the same way provided we place a non-zero cost linking edges incident to the depot only.

4.5 Conclusions and Future Work

The flow chart presented in Fig. 4.13 shows a schematic of our original research findings, offering a visual guide to the classification of different special cases of OSRP_{RG} and how these are covered by our theorems.

In this chapter, we utilised the structure of orchards, which resemble regular grid graphs, to address the general OSRP on regular grid graphs, referred to as OSRP_{RG}. Focusing on regular grid graphs enabled us to identify specific cases of the OSRP that can be solved in polynomial time.

This chapter provided polynomial-time complexity proofs and developed solution patterns for the OSRP_{RG}. These patterns serve as a foundation for solving large-scale problems and extend the utility of OSRP_{RG} to mathematical analysis for graphs with grid-like characteristics, making it highly relevant for real-world applications.

A key achievement of this chapter is the systematic categorisation of regular grid graphs into single-layer and multi-layer classifications. This categorisation highlights the practical significance of routing problems in agrochemical-spraying vehicle operations. Through mathematical analysis, we validated optimal solution patterns, such as the **ad-**

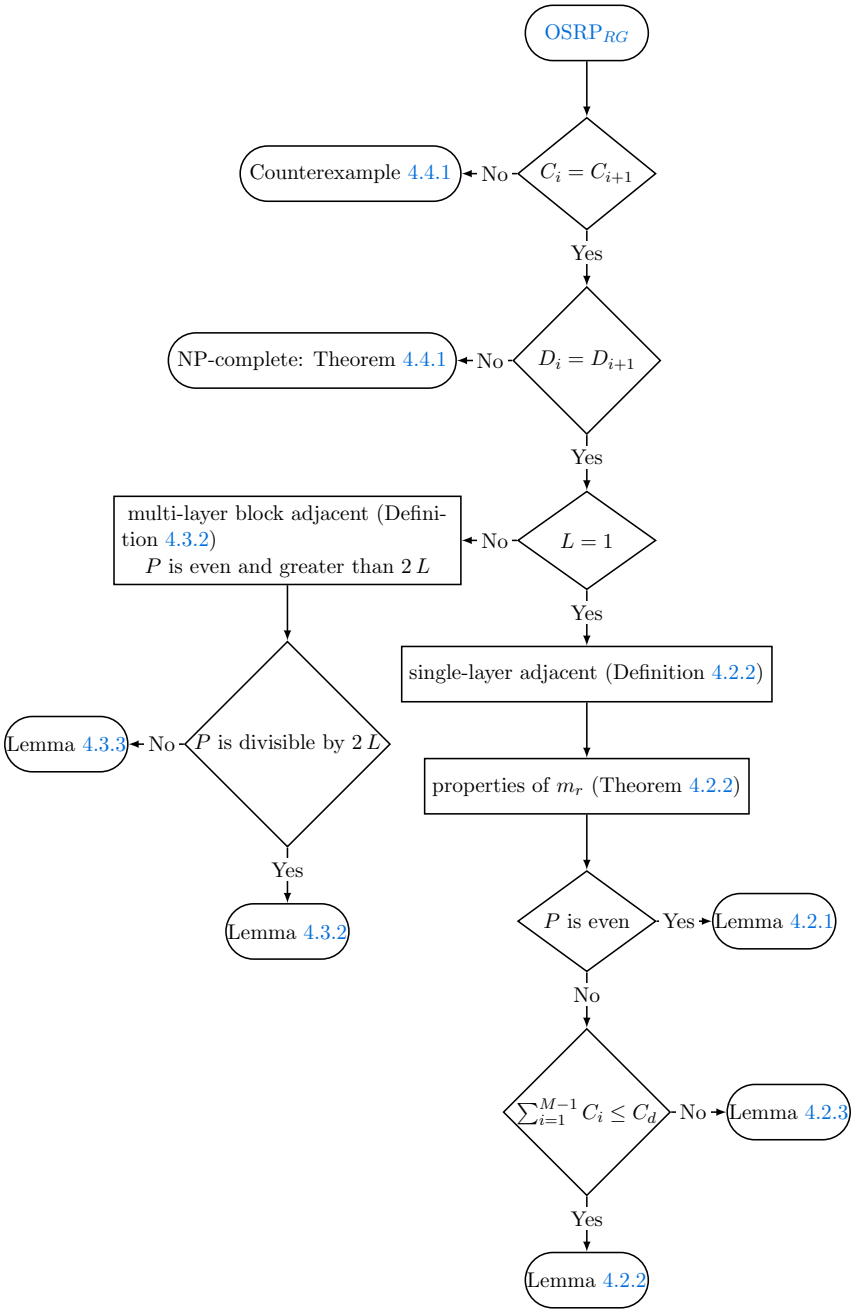


Figure 4.13: The conditions $C_i = C_{i+1}$ and $D_i = D_{i+1}$ imply that the cost of linking edges within a group is identical, and the cost of demand edges is the same. If these conditions are not met, a counter-example is presented to demonstrate non-adjacency. The case $L = 1$ corresponds to the single-layer problem, while $L > 1$ pertains to the multi-layer problem. The discussion covers various conditions, such as even or odd capacity P and the cost relationship. For further details on notation, please refer to Section 4.1, while Section 4.2 and Section 4.3 provide additional information on the solution patterns.

jacent solution for single-layer instances and the **block adjacent** solution for multi-layer instances. This chapter also considered various scenarios, including even and odd capacity requirements, and offered counterexamples where non-adjacent solutions may arise. Additionally, we proved that the OSRP$_{RG}$ becomes strongly NP-complete in cases with non-equal demands. While this chapter has contributed significantly to solving the OSRP on regular grid graphs, it also highlights opportunities for future research.

One limitation is the focus on regular grid structures, which may not always represent real-world graphs accurately. However, this opens the door to exploring approximation methods for irregular grid graphs, offering potential for solving practical routing problems more efficiently. Moreover, the findings in this chapter lay the groundwork for advancing the field with more sophisticated techniques, such as decomposition methods, to tackle large-scale and diverse scenarios.

The insights gained here motivate the next chapter, where we extend the study to real-world orchard layouts and introduce approximation methods that bridge the gap between theoretical grid structures and practical, irregular graphs. The solution patterns developed in this chapter provide a foundation for effectively addressing the complexities of real-world agricultural routing problems, as discussed in Chapter 5.

Chapter 5

Routing Plans for Automated Spraying Robots in Real-World Agricultural Areas

Orchards resemble regular grid graphs, characterized by parallel demand edges, connecting edges linking these demand edges, and full node accessibility. Chapter 4 demonstrated that it is possible to obtain optimal solutions and generate path patterns for regular grid graphs based on specific characteristics, effectively solving OSRP on such regular grid graphs in polynomial time. Regular grid graphs serve as idealized models for orchards, and the findings in Chapter 4 highlight that the solution paths consistently follow an adjacent pattern. Consequently, orchards can be approximated as regular grid graphs, utilizing this grid-like structure to simplify the problem and make it more manageable.

In this chapter, we extend the application of our methodology to optimise the robot vehicle routing problem to real-world orchards, which are irregular grid graphs. These irregular grid graphs often present edges with uneven demand and different lengths, and have multiple edges interrupted by obstacles. In this study, we aim to solve the OSRP by adapting mathematical insights originally developed for regular grid graphs to generate scalable and high-quality solutions for more realistic, irregular grid graphs that reflect the layouts of actual orchard blocks.

The methodologies are evaluated using three cases that reflect real-world constraints encountered by major food and agriculture businesses. The first case features a rectangular orchard resembling a regular grid graph, while the other two cases exhibit increasing irregularity, with the third being particularly complex due to the presence of an obstacle. Despite sharing the same underlying mathematical structure, the differing shapes of these orchards lead to varying outcomes when applying the same routing solution method. More

complex graph structures introduce additional challenges, often necessitating extra data processing or the incorporation of supplementary solution strategies before deploying the primary algorithms.

In this chapter, lower and upper bounds are obtained by approximating the orchard with small and large regular grid graphs, respectively, and by applying adjacent solution patterns specifically tailored to these regular grid structures. Additionally, a simple heuristic approach is introduced to address three distinct, realistic orchard graphs, and its performance is compared against the approximation method.

5.1 Notation and Definitions

This section introduces the necessary notation and explores the relevant theories and characteristics of regular grid graphs, forming the foundation for the approximation methods detailed in Section 5.2 and Section 5.3.

5.1.1 Notation

The following notation is used in this chapter. For a full list of symbols, refer to the List of Symbols.

List of symbols:

$\mathcal{N} = \{1, 2, ..., n\}$ set of all nodes. $|\mathcal{N}|$ represents the number of nodes.

$\mathcal{E} = \{(i, j) \in \mathcal{A} : i < j\}$ set of undirected edges (set of arcs when the direction is not a consideration). $|\mathcal{E}|$ represents the number of edges.

C_{ij} non-negative cost of arc $(i, j) \in \mathcal{A}$, where $C_{ij} = C_{ji}$.

D_{ij} non-negative demand for the arc $(i, j) \in \mathcal{E}$.

$\mathcal{R} = \{1, 2, ..., R\}$ set of the index of the *robotic sprayers tour*. We refer to robotic sprayers as *robots* for simplicity. R is the minimum number of robots required to spray the orchard (typically $|\mathcal{R}| = R = \left\lceil \frac{\sum_{(i,j)\in\mathcal{E}} D_{ij}}{P} \right\rceil$).

P the capacity of the robotic sprayer. In regular grid graphs, this capacity is defined as the maximum number of demand edges that each robot can spray. In this study, P is an integer.

M the total number of demand edges.

Decision Variables:

m_r the number of demand edges sprayed by robot r.

f_r the index of the demand edge that is furthest away from the depot, which is sprayed by robot r.

n_r the index of the demand edge that is nearest to the depot, which is sprayed by robot r.

$S_r = \{n_r, \ldots, f_r\}$ the set of indices of demand edges sprayed by robot r.

$S = (S_1, S_2, \ldots, S_{\mathfrak{R}})$ the solution of an OSRP_{RG}.

$\delta \in [0, 1]$ the *fraction* of demand edges to be included.

In this chapter, we assume that the capacity P is an integer multiple of the unit demand, which ensures that each demand edge can be serviced by a single robot when discussing the solution path pattern on a single-layer regular grid graph.

5.1.2 Definitions

An orchard can be roughly approximated as a rectangular, grid-like graph, with demand edges resembling parallel lines. We have developed an optimal solution strategy for OSRP on single-layer regular grid graphs in polynomial time in Chapter 4. In this section, we outline the properties and methods for generating the optimal solution paths on a single-layer regular grid graph. More detailed proofs and additional information can be found in Chapter 4.

Definition 5.1.1 (Single-layer Regular Grid Graph)**.** A *single-layer regular grid graph* is a grid graph consisting of M demand edges linked together to form a $2 \times M$ grid, where each pair of linking edges has the same cost, and every demand edge has identical cost and demand.

Chapter 4 discussed both multi-layer and single-layer regular grid graphs. In this chapter, however, we focus on the single-layer regular grid graph. Definition 5.1.1 provides the formal definition of a single-layer regular grid graph. For simplicity, we will refer specifically to the single-layer regular grid graph as "regular grid graph" in the remainder of this chapter.

The term "pair of linking edges" refers to the edges that are paired with their corresponding opposite linking edges, as illustrated in Fig. 5.1. For example, the edge $(1, 2)$

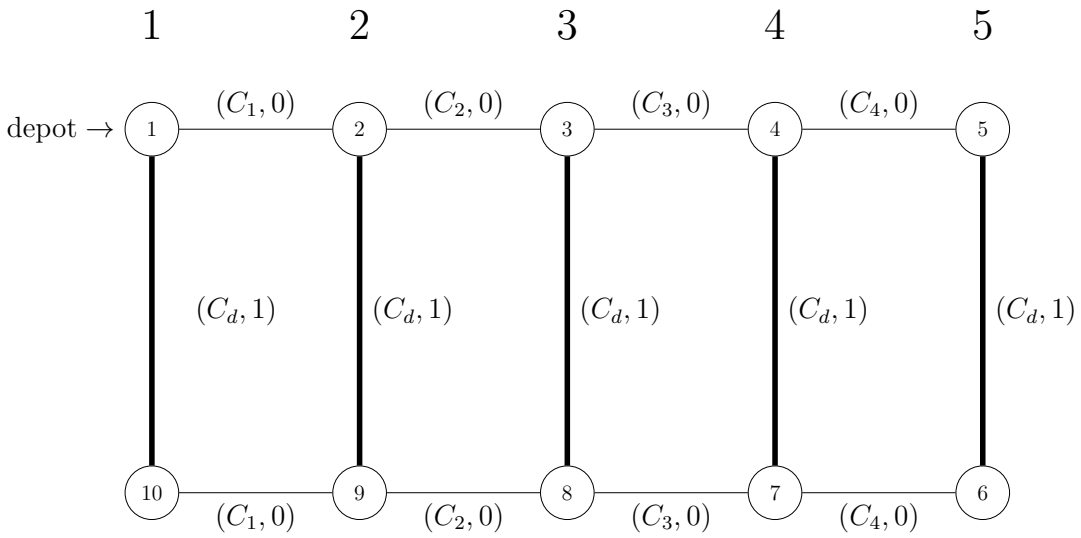


Figure 5.1: The depot is located at node 1, with the pair of numbers in parentheses representing the cost and demand associated with each edge. The linking edges are the top and bottom horizontal edges of the graph, where each pair of opposite linking edges has the same cost. In contrast, the vertical edges are demand edges (highlighted in bold), each having a uniform cost of C_d and a unit demand of 1.

and edge $(9, 10)$ form a pair of linking edges, both with the same cost C_1 and a demand of 0.

Based on Definition 5.1.1, there are two main properties of a *regular grid graph*:

(1) The cost for each pair of linking edges is the same.

(2) All demand edges have the same cost and demand.

Typically, in real-world applications, the linking edges surrounding an orchard are access paths and have no demand associated to them. If the cost and demand of an orchard graph do not meet the criteria outlined in Definition 5.1.1, then the graph should be classified as an *irregular grid graph*. In actual orchards, uneven demands along demand edges and the presence of obstacles are common, making it inaccurate to assume that the costs along pairs of linking edges or demand edges are equivalent. Therefore, real-world applications generally involve irregular cases, while the regular grid graph serves as an idealized representation of orchards.

Definition 5.1.2 (Feasible Solution)**.** A solution of OSRP_{RG} is a tuple $S = (S_1, S_2, \ldots, S_{\mathcal{R}})$, where $S_r = \{n_r, \ldots, f_r\}$ represents the set of indices of demand edges sprayed by robot r. A *feasible solution* is a partition of the set of demand edge:

$$\left| \bigcup_{i \in \mathcal{R}} S_i \right| = M \quad \text{and} \quad S_i \cap S_j = \emptyset \quad \forall\ i \neq j$$

Definition 5.1.3 (Adjacent Solution)**.** A feasible solution S is considered *adjacent* for a OSRP_{RG} if $f_r + 1 = n_{r+1}$.

As shown in Fig. 5.1, if the capacity is $P = 2$, then one of the corresponding adjacent solutions is $S = (S_1, S_2, S_3)$, where $S_1 = \{1\}$, $S_2 = \{2, 3\}$, and $S_3 = \{4, 5\}$. In this configuration, the values are $n_1 = 1$, $n_2 = f_1 + 1 = 2$, and $n_3 = f_2 + 1 = 4$.

Proposition 5.1.1 introduces a method for constructing a solution path pattern for OSRP_{RG}, which has been proven to be an optimal solution in Chapter 4.

Proposition 5.1.1 (An Optimal Path Pattern Construction)**.** Given the indices of the edges that are sprayed by a robot in an *adjacent solution*, we can construct an *optimal path pattern* as follows. First, find the furthest edge from the depot for each robot and connect it to the depot. Next, follow a zigzag pattern to spray all the demand edges. Finally, return to the depot.

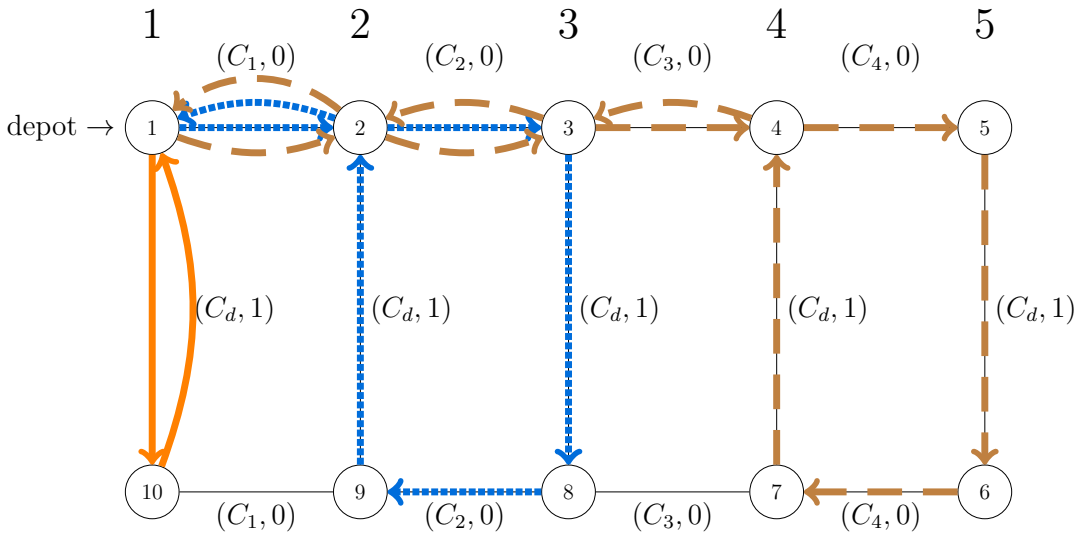


Figure 5.2: The depot is located at node 1, with the pair of numbers in parentheses representing the cost and demand associated with each edge. If $P = 2$, then a solution for this problem involves three robotic sprayers, $S = (S_1, S_2, S_3)$, with $S_1 = \{1\}$, $S_2 = \{2, 3\}$, and $S_3 = \{4, 5\}$. A generated optimal path for robot 1 is $[1 \to 10] \to 1$, highlighted in orange with the sprayed demand edges denoted by square brackets. Similarly, the path patterns for robot 2 and robot 3 are highlighted in blue and brown, respectively, according to Proposition 5.1.1.

As depicted in Fig. 5.2, if the capacity $P = 2$, the optimal path pattern is as follows: robot 3 follows the path $1 \to 2 \to 3 \to 4 \to [5 \to 6] \to [7 \to 4] \to 3 \to 2 \to 1$, with the sprayed demand edges indicated by square brackets. robot 2 follows the path $1 \to 2 \to [3 \to 8] \to [9 \to 2] \to 1$. Finally, robot 1 follows the path $[1 \to 10] \to 1$.

Lemma 5.1.1 (Solution Property)**.** For OSRP$_{RG}$ with integer capacity P, there exists a robot $k \in \mathcal{R}$ that defines an adjacent optimal solution pattern where each robot r sprays m_r edges, with

$$m_r = \begin{cases} P & \text{if } r > k \\ P - 1 & \text{if } 2 \leq r \leq k \\ M - \sum_{r=2}^{\mathcal{R}} m_r & \text{for } r = 1 \end{cases}.$$

Furthermore, if P is even, then $k = 1$.

The intuition behind Lemma 5.1.1 is as follows: every robot must travel an even number of demand edges to ensure they can return to the depot. If a robot sprays an odd number of demand edges, it will be forced to travel over at least one demand edge without spraying, potentially increasing the total cost. Therefore, if P is an even number, all robots except the first one will deplete their capacity, as the first robot will spray the remaining demand edges, which may not fully deplete its capacity. Conversely, if P is an odd number, the strategy is to send fewer robots without significantly increasing duplication costs. As proven in Chapter 4, $m_1 \leq m_2 \cdots \leq m_{|\mathcal{R}|}$, ensuring that there is a robot k that sprays $P - 1$ demand edges (an even number), while robots $k + 1$ through R deplete their capacity when P is odd.

In real-world scenarios, such as large orchards with distant demand edges, efficiently utilizing capacity and identifying the appropriate robot k becomes crucial.

Section 5.2 and Section 5.3 develop approximation methods to address the real-world irregular OSRP$_{RG}$ by leveraging the properties of adjacent solution patterns. Specifically, Section 5.2 introduces a small regular grid graph approximation method for generating a lower bound, while Section 5.3 presents a large regular grid graph approximation method for generating an upper bound.

5.2 Small Regular Grid Graph Approximation

Real-world orchards are typically irregular, with uneven demand and varying costs along the edges. To address these challenges, we approximate the irregular graph by segmenting and transforming it into a regular grid graph. We then apply Proposition 5.1.1 to construct an optimal solution, which serves as a lower bound for OSRP$_{RG}$, as demonstrated in Proposition 5.2.2.

Proposition 5.2.1 (Small Regular Grid Graph Approximation)**.** Consider a *OSRP* on an irregular graph G. Let $\delta \in [0, 1]$ denote the *fraction* of demand edges to be included. Let $\mathcal{D} \subseteq \mathcal{E}$ be the set of edges with non-zero demand. From these, let us denote $\mathcal{D}_E$ as

the subset of demand edges to be excluded, and $\mathcal{D}_I$ as subset to be included. Then, we have partition $\mathcal{D} = \mathcal{D}_E \cup \mathcal{D}_I$ such that $|\mathcal{D}_E| = \lfloor \delta|\mathcal{D}| \rfloor$ and $D_{kl} \leq D_{ij}\ \forall\ (k,l) \in \mathcal{D}_E$, $(i,j) \in \mathcal{D}_I$. That is, we select a fraction δ of edges with smallest demand to be excluded ($\mathcal{D}_E$) while the rest are included. For the selected demand edges ($\mathcal{D}_I$), approximate their demand and cost by assigning the demand and cost values corresponding to the minimum in $\mathcal{D}_I$, while setting the demand and cost of all other edges ($\mathcal{D}_E$) to zero. Similarly the cost of each pair of linking edges is set to be identical, using the *minimum value* between the two linking edges. The resulting approximation, denoted SRG, forms a small regular grid graph that is a part of the original graph G, including only a subset of the original demand edges.

To see the effect of δ on the approximation of the cost, consider that the true cost of traveling the demand edges is $\sum_{(i,j)\in\mathcal{D}} C_{ij}$ while in the approximation the cost is $C(\delta)$ (minimum cost in $\mathcal{D}_I$ for given δ). Hence, the error in the estimation of the solution cost is (at least)

$$\begin{aligned} C_{cutoff} &= \sum_{(i,j)\in D_E} C_{ij} + \sum_{(i,j)\in\mathcal{D}_I} (C_{ij} - C(\delta)) = \sum_{(i,j)\in\mathcal{D}} C_{ij} - C(\delta)|\mathcal{D}_I| \\ &= \text{constant} - C(\delta) \times \lceil\, (1-\delta)|\mathcal{D}|\, \rceil \end{aligned}$$

As δ increases, $C(\delta)$ increases and $\lceil\, (1-\delta)|\mathcal{D}|\, \rceil$ decreases, so that the optimum (minimum value of C_{cutoff}) can be expected to occur for some value $0 < \delta < 1$. We numerically optimise for the value of δ that gives the best (highest) lower bound.

In our case, it is often necessary to eliminate some demand edges that have a significant lower demand than the majority of edges, which introduces the parameter δ. This ensures that the approximation is not too small for the problem and that the small regular grid graph can encompass most of the demand edges in the original graph. Furthermore, to establish a valid lower bound, we reduce the linking edges to the length of the shorter edge in each pair. Since all demand edges are serviced at least once, adding the remaining length of the demand edges still maintains a valid lower bound, as proved in Proposition 5.2.2.

Proposition 5.2.2. The optimal solution of the small regular grid graph SRG, obtained by the *Small Regular Grid Graph Approximation* for any $\delta \in [0,1]$, with the addition of the cutoff demand edges cost, is a *lower bound* for the OSRP on the original graph G.

Proof. Let G be the original graph, S_G represents a *feasible solution* on G, and $\overline{S}_{SRG}$ denotes the *optimal solution* on the small regular grid graph SRG. Let C_{cutoff} represent the cost of the demand edges that are considered in G but excluded from SRG. For every feasible solution S_G on the original graph G, there exists a corresponding solution

Algorithm 5.1 Small Regular Grid Graph Approximation

Input: C_{ij}, D_{ij}, δ, $\mathcal{E}^+$
Output: Total Cost

$index = \lceil (1 - \delta) \times |\mathcal{E}^+| \rceil$ ▷ Determine the index in the sorted set of demand edges
$(i, j) \leftarrow \mathcal{E}^+[index]$
$\Delta \leftarrow D_{ij}$
$\Gamma \leftarrow C_{ij}$
$cutoff = 0$
if $D_{ij} \geq \Delta$ **then**
    $D_{ij} \leftarrow \Delta$ ▷ Reduce higher demands to Δ
    $cutoff \mathrel{+}= C_{ij} - \Gamma$
    $C_{ij} \leftarrow \Gamma$
else if $D_{ij} < \Delta$ **then**
    $D_{ij} \leftarrow 0$ ▷ Skip edges with small demand
    $cutoff \mathrel{+}= C_{ij}$
    $C_{ij} \leftarrow 0$
$Cost \leftarrow$ Proposition Proposition 5.1.1
Total Cost $\leftarrow Cost + cutoff$

path pattern. This same solution pattern can also be considered a *feasible solution* on the graph SRG, as it covers all demand edges in SRG.

According to Proposition 5.2.1, using the *minimum value* between pairs of linking edges implies that the solution pattern remains the same, but the linking edge cost is reduced. This means that the total cost of S_G, denoted by C_{S_G}, is greater than or equal to the total cost of the *feasible solution* S_{SRG}, denoted by $C_{S_{SRG}}$. Since $\overline{S}_{SRG}$ is the optimal solution on the small regular grid graph, its cost $C_{\overline{S}_{SRG}}$, when adding the cutoff cost, is less than or equal to the cost of the feasible solution S_{SRG}.

$$C_{S_G} \geq C_{S_{SRG}} \geq C_{\overline{S}_{SRG}} + C_{cutoff}$$

Thus, we have shown that the optimal solution for the small regular grid graph provides a lower bound for the $OSRP$ on the original graph G.

□

Algorithm Algorithm 5.1 outlines the principal steps involved in the *small regular grid graph approximation* process:

5.3 Large Regular Grid Graph Approximation

The *large regular grid graph approximation* (as shown in Algorithm 5.2) can be used to find an *upper bound* for OSRP_{RG}. The main idea is to apply the adjacent solution path

pattern from the regular grid graph to irregular cases, generating a feasible solution.

The Algorithm 5.2 starts with robot R, which sprays the furthest demand edge f_R. The robot then follows adjacent edges until its capacity is reached. Next, robot $R-1$ begins at $f_{R-1} = n_R - 1$ and continues along adjacent edges until its capacity is exhausted. This process continues for each robot down to robot $k+1$. For robots k through 1, path replanning may be required to prevent duplicating sprayed demand edges. It is essential to determine whether the number of unsprayed edges is even or odd before robot k begins its operation. If the number of unsprayed edges is even, then all subsequent robots must have path plans that include an even number of demand edges. Conversely, if the number is odd, one robot should be assigned a path with an odd number of demand edges, while the remaining robots will each handle an even number of sprayed edges. A more detailed explanation is available in Lemma 5.1.1, with further proofs and results in Chapter 4.

In Algorithm 5.2, $new\_paths[r]$ represents the set of sprayed demand edges for robot r, starting from the furthest demand edge $f_r = n_{r+1} - 1$ and continuing until its capacity is depleted. The capacities of robots with indices $r > k$ will be depleted. However, for the k-th robot, its capacity cannot simply be reduced to spray an even number of edges, as the demand along each demand edge varies, resulting in different numbers of sprayed edges. The number of unsprayed edges and the current number of demand edges in $new\_paths[r]$ are critical factors to consider. If the number of unsprayed edges is odd, it is sufficient for only one robot to spray an odd number of demand edges, and this robot should be the first from the set $\{k, k-1, \ldots, 1\}$. In cases where an odd number of demand edges have been sprayed, the last demand edge should be removed from $new\_paths[r]$ to ensure an even number of sprayed demand edges. If the number of unsprayed edges is even, then all robots r with $1 \leq r \leq k$ will spray an even number of demand edges.

Both of the small and large regular grid graph approximations are highly efficient and easy to implement, making them well-suited for realistic orchard spraying applications. These methods provide lower and upper bounds, respectively, that can be effectively applied to real-world problems. Although both the small and large regular grid graph approximation methods do not allow edge splitting, with each edge being serviced by only one robot, there remains potential for even better solutions when splitting is permitted. These approximation methods are not only straightforward and efficient, but can also be easily combined with other meta-heuristic approaches, providing an excellent starting point for agricultural spraying applications. Section 5.4 presents an alternative constructive heuristic method that produces high-quality solutions (upper bound).

Algorithm 5.2 Large Regular Grid Graph Approximation

Input: k, P, *unsprayed*, $\mathcal{R}$
Output: *new_paths*, Total Cost

odd_path ← **False**
for each robot $r \in \mathcal{R}$ **do**
    if $r > k$ **then**
        new_paths[r] ← Deplete the capacity of robot r
Update *unsprayed* by removing the sprayed demand edges
for each robot $r \in \{k, k-1, \ldots, 2\}$ **do**
    if |*unsprayed*| is even **then**
        if |*new_paths*[r]| is odd **then**
            new_paths[r] ← Eliminate the last sprayed demand edge
    else if not *odd_path* **then**
        new_paths[r] ← Deplete the capacity of robot r
        odd_path ← **True**
    else
        new_paths[r] ← Eliminate the last sprayed demand edge
    Update *unsprayed* by removing the sprayed demand edges
new_paths[1] ← Remaining unsprayed demand edges
Total Cost ← Proposition 5.1.1

5.4 Heuristic Method

In contrast to the regular grid graph approximation, the simple heuristic exploits the different spray demand of the rows to maximize the utilization of the robots' spray capacity. Due to that, a solution might not follow the adjacent solution pattern for regular grid graphs.

The idea of the heuristic is to create a solution, in which all robots spray an even number of demand edges, unless the total number of those edges is odd or the robots' spray capacity can only cover one long edge, and maximizing their spray capacity by allowing them to spray non-adjacent edges. In the case where there is an odd number of demand edges, the heuristic only allows one robot to spray an odd number of them.

Algorithm 5.3 greedily assigns first the maximum number of uncovered demand edges to a robot starting at the point furthest from the depot. This is motivated by the insight that, for regular grid graphs, there always is an optimal adjacent solution according to Lemma 5.1.1. The ordering of the edge list L depends on the reachability of the "boundary" nodes of the grid graph starting from the depot node, where a boundary node is located at the boundary of the orchard and the outside world. These boundary nodes are explored in anti-clockwise manner as the orchards are shown in Fig. 5.3 and Fig. 5.4 in the next section. The algorithm then tries to use any remaining capacity of the robot to cover additional low demand edges in such a way that the robot covers an even number of edges if possible. We note that we always check for additional edges to add to S_r in consecutive order from $L[\ell - 1]$ to $L[1]$. Since the demand edges are in a sorted order,

Algorithm 5.3 Simple Constructive Heuristic

Input: C_{ij}, D_{ij}, P

Let $L = [(i_1, j_1), (i_2, j_2), \ldots, (i_M, j_M)]$ be a list of demand edges
Sort L in the order of reachability of the edge from the depot node when the "boundary" nodes of the grid graph are explored in anti-clockwise manner starting from the depot node
$oddReq \leftarrow$ (Is M odd?) ▷ Whether a robot with odd demand edges is required.
$r \leftarrow 0$ ▷ Robot index
while $L \neq [\,]$ **do**
    $r \leftarrow r + 1$
    $\ell \leftarrow \min\{i : \sum_{k=i}^{|L|} D_{L[k]} \leq P\}$
    $S_r = \{L[\ell], L[\ell+1], \ldots, L[|L|]\}$ ▷ Largest set of consecutive edges that r can serve
    $p \leftarrow P - \sum_{k \in S_r} D_k$
    if ($|S_r|$ is odd & $\exists\ k \in L \setminus S_r : D_k \leq p$) || $oddReq$ **then**
        $S_r \leftarrow S_r \cup \{k\}, \quad p \leftarrow p - D_k$
    else if $|S_r|$ is odd & $|S_r| > 1$ & !$oddReq$ **then**
        $S_r \leftarrow S_r \setminus \{\ell\}, \quad p \leftarrow p + D_\ell$
    $\ell \leftarrow \max(L \setminus S_r)$
    while $\exists\ k \in L \setminus S_r : D_k + D_\ell \leq p$ **do**
        $S_r \leftarrow S_r \cup \{k, \ell\}, \quad p \leftarrow p - (D_k + D_\ell), \quad \ell \leftarrow \max(L \setminus S_r)$
    if $|S_r|$ is odd **then** $oddReq \leftarrow false$
    $L \leftarrow L \setminus S_r$ ▷ Remove covered edges from list
    Construct a shortest path tour SP_r based on S_r

the final shortest path tour is easy to construct. We start our tour by travelling from the depot to the node of the furthest demand edge which is closest to the depot, and then travelling back covering pairs of demand edges in alternating directions and finally take the shortest path back to the depot from the last demand edge.

5.5 Numerical Results and Discussion

In this section, we evaluate the algorithms proposed in Section 5.2 and Section 5.3 by testing them on three real-world orchards. We compare the outcomes with the solution method outlined in Chapter 3, which tackles the same NP-hard OSRP. In Chapter 3, we introduced a MILP method that incorporates lazy constraints, symmetry eliminations, and a heuristic repair technique. The results from the MIP algorithm are limited to 3600 working limits on Gurobi's parameter setting, which equates to approximately an hour.

Notably, the solution to a CARP serves as an upper bound for SDCARP. Finding an optimal solution for the NP-hard real-world spraying problem in polynomial time is challenging. While the solution method described in this chapter does not allow edge splitting, it is straightforward and efficient at finding both lower and upper bounds in polynomial time. Additionally, it is scalable for large problems, providing a reasonable solution that meets the expectations of orchard owners.

The three graph instances that we introduce next possess interesting and realistic

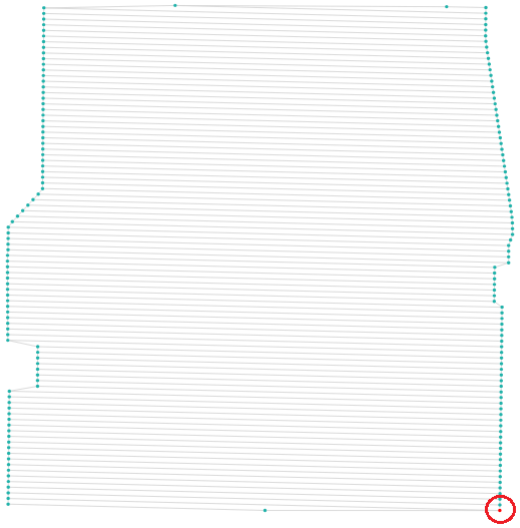
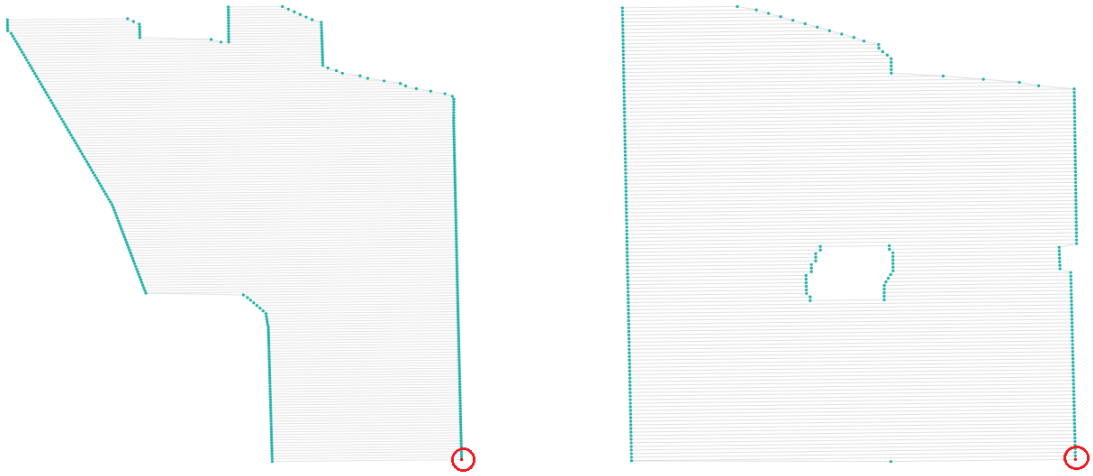


Figure 5.3: An orchard graph similar to a regular grid graph, referred to as *RegGrid*, with all internal edges designated as demand edges and the depot located at the bottom-right node (highlighted in red).

features of actual orchards. Fig. 5.3 resembles a regular grid graph, referred to as *RegGrid*, which simplifies the use of approximation and heuristic methods. In contrast, the orchard in Fig. 5.4a is irregular, with significant cost variations along the demand edge (*CostVar*). The orchard in Fig. 5.4b includes an obstacle area in the middle (*Obs*). The placement of the depot at any of the four corners (top left, top right, bottom left, or bottom right) does not affect the results. However, positioning the depot in the middle or at a specific spot within the orchard would introduce additional considerations and could be explored in future research. Each of the three orchards has a *fixed depot* located at one of the corner points.

In the large regular grid graph approximation, the choice of k values influences the quality of the solutions. We evaluate all three cases with different selections of k, and the numerical results indicate that both *RegGrid* and *Obs* achieve their optimal solutions when k is greater than or equal to the number of required robots. However, the best solution of *CostVar* is obtained when k takes any of the values $16, 17, 18, 19, 20, 21, 22$ or 23. Fig. 5.5 illustrates the performance across various k values for the *CostVar* case, with the optimal solution highlighted in red and enlarged for emphasis.

In the small regular grid graph approximation, some demand edges have significantly lower demand compared to others in the three instances, particularly in the cases of *CostVar* and *Obs*. The selection of δ is critical in the solution process. In the *CostVar* case, we remove the prominent demand edges on the right before applying the small regular



(a) An orchard with significant cost and demand variations along the demand edges, referred to as *CostVar*, with the depot located at the bottom-right node (highlighted in red).

(b) An orchard displaying a central obstacle area, termed *Obs*, with the depot situated at the bottom-right node (highlighted in red).

Figure 5.4: The orchard cases feature highly irregular conditions with significant cost and demand differences.

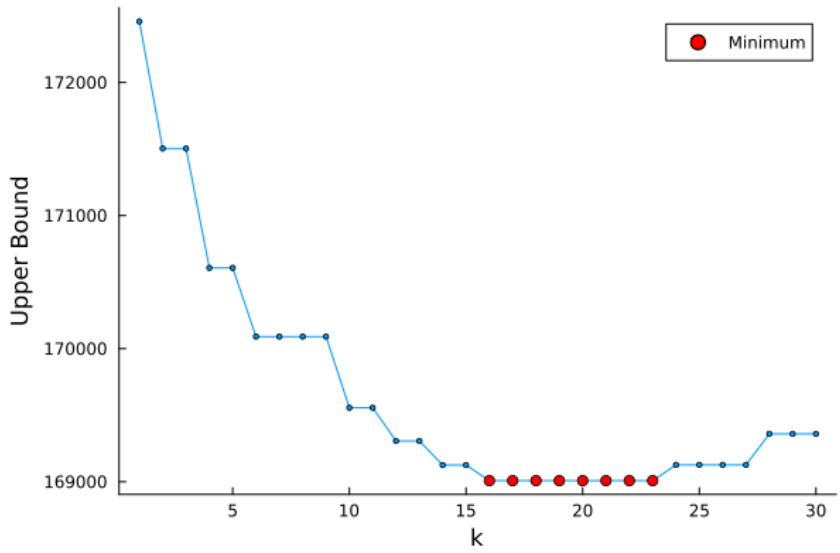


Figure 5.5: Comparison of upper bounds for *CostVar* computed using Algorithm 5.2 across different k values. The optimal solutions for *CostVar* occur at $k = 16, 17, 18, 19, 20, 21, 22, 23$.

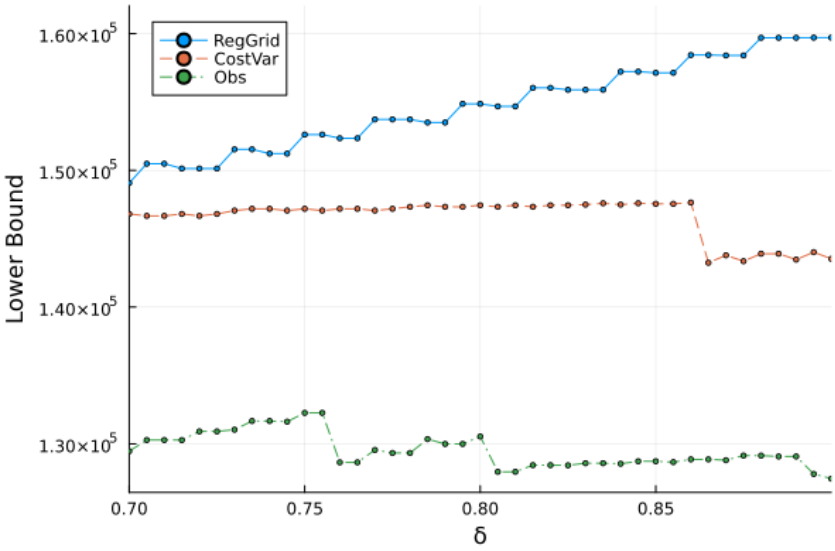


Figure 5.6: Comparison of lower bounds obtained using Algorithm 5.1 for various δ values.

grid graph to simplify the solution process. Similarly, for the *Obs* case, we eliminate the right part of the hole areas. Fig. 5.6 presents the results of varying the parameter δ to compare its impact on the solutions. During this process, we tested the solution with various k and δ values, and all three cases achieved their optimal solutions when k was greater than the number of required edges. Therefore, we set k for all three cases to be equal to the number of required edges plus 1.

Fig. 5.6 shows that the best small regular grid graph approximation typically occurs when the approximation includes most of the demand edges while excluding those with significantly lower costs, but without including all of them. Additionally, considering all potential δ values is feasible due to the limited number of demand edges.

Table 5.1 compares the results obtained using the solution method from Chapter 3 with those abbreviated as "MIP". The column "UB" reports the corresponding upper bound, while "LB" provides the lower bound. The "Small Grid" column presents the best results from the small grid graph approximation, serving as a lower bound for the problem. The "Large Grid" column presents the results from the large grid graph approximation, representing an upper bound. The "Simple Heuristic" column provides the heuristic solution, which can also be viewed as an upper bound for the problem. The "MIP-warm" column shows results from the MIP solver using the best upper-bound solution as a warm start. We report the "Gap" to evaluate the solution quality and the optimality feature, which represents the difference between the *best upper bound* and the *best lower bound* (with the best solutions highlighted with a star).

Table 5.1 shows that the small regular grid graph approximation provides a better lower bound for all three instances compared to the MIP. For the upper bound, both the

Table 5.1: Comparison of results obtained using the solution method from Chapter 3 and those abbreviated in "MIP". The "UB" column reports the corresponding upper bound, while "LB" reports the lower bound. The "Small Grid" column presents best the results from the small grid graph approximation, considered as a lower bound for the problem. The "Large Grid" column presents the best results from the large grid graph approximation, seen as an upper bound for the problem. The "Simple Heuristic" column provides the simple heuristic solution, considered an upper bound for the problem. The "MIP-warm" column shows results from the MIP solver using the best upper-bound solution as a warm start. The "Gap" column represents the difference between the best upper bound and the best lower bound. Best solutions are highlighted with a star ($*$).

	LB			**UB**				**Gap**
	MIP	Small Grid	MIP-warm	MIP	Large Grid	Simple Heuristic	MIP-warm	$\frac{UB^*-LB^*}{UB^*}$
RegGrid	144225.22	161034.50*	143454.89	191818.82	163039.17	161258.24	161257.27*	0.14%
CostVar	133131.05	147646.72*	132041.89	203001.49	169047.28	167990.61	167636.85*	11.92%
Obs	124818.62	132260.18*	126599.49	174652.49	145481.76	146086.74	145023.66*	8.80%

large grid graph approximation and the simple heuristic method outperform the MIP and are much more efficient in the process. Additionally, the simple heuristic appears to offer a better upper bound for *RegGrid* and *CostVar*, though it is slightly slower than the large grid graph approximation.

The MIP typically incurs higher computational costs and complexity, with a time limit of one hour and can not improve the solution a lot even though apply the best solution as a warm stat. In contrast, the small and large regular grid graph approximations, and the simple heuristic method take less than a second. These methods are significantly faster and simpler, making them more efficient, especially when computational resources are limited or rapid approximations are required.

5.6 Conclusions and Future Work

This chapter focused on developing a novel approach to addressing a real-world agricultural routing problem, starting with the structure of actual orchard properties and approximating them as regular grid graphs. The primary achievements of this chapter included the development of scalable solution methods that utilise two approximations: a small regular grid graph and a large regular grid graph, which provide lower and upper bounds on the solution cost, respectively. The large regular grid graph approximation has also offered a high-quality feasible solution with an accurate routing plan for automated spraying robots. Additionally, a simple heuristic has been proposed to prioritise capacity utilisation, improving resource efficiency while ensuring practical applicability in real-world agricultural settings.

To validate the approach, we introduced three examples of orchard graphs with realistic features, and provided a single representative instance for each type. The three *types* of orchard graphs include: *Regular Grid Graph* in Fig. 5.3: A highly regular and straightforward graph structure that facilitates ease of implementation and analysis. *Graph with Uneven Costs and Demands* in Fig. 5.4a: A graph with varying costs and demands on different edges, reflecting more complex and realistic scenarios that deviate significantly from the assumptions of the simplified regular grid graph for which we have theoretical results. *Graph with Internal Obstacles* in Fig. 5.4b: A graph incorporating internal obstacles within the orchards, adding another layer of realism and complexity to the problem.

This chapter established a foundation for more effective and efficient agricultural spraying strategies, bridging the gap between theoretical research and practical applications, and laying the groundwork for future studies in this vital field. It broadens the scope of research in the agricultural spraying field, with real-world data instances contributing to the enrichment of existing benchmarks for agricultural applications and supporting the testing of SDCARP and CARP.

While this work demonstrates a significant step forward, it has limitations. The current methods cannot guarantee exact solutions unless the lower and upper bounds coincide. Additionally, the current solution methods do not take advantage of the splittable properties of the model. Combining these methods with heuristics based on grid-graph solution patterns to explore splittable solutions has the potential to improve the upper bound, narrowing the feasible solution space and potentially delivering high-quality feasible solutions. Moreover, the approximation methods are very fast and could be effectively utilised in decomposition methods. They can also be combined with meta-heuristic techniques to further improve solution quality. Thus, the methods proposed here can be refined and integrated with other techniques to improve overall solution performance, making them a versatile and practical tool for addressing complex problems in agricultural applications.

Chapter 6 focuses on the practical applications of OSRP in more complex multi-block instances, introducing methods to address the challenges posed by diverse and irregular orchard structures consisting of multiple blocks.

Chapter 6

Algorithms for Routing Automated Spraying Vehicles in Complex Multi-Block Agricultural Areas

The mathematical MILP models for the OSRP were introduced in Chapter 3, where both exact and heuristic techniques were applied to find solutions for the problem. Chapter 4 analyzed the layout of orchards and the mathematical properties of regular grid graphs, proposing a polynomial-time solvable OSRP (a special case of the CARP) on regular grid graphs, named OSRP_{RG}. Moreover, theorems, lemmas, and mathematical proofs were developed to establish the properties of optimal solutions for the OSRP_{RG}. Chapter 5 builds on the properties detailed in Chapter 4, proposing straightforward and scalable small and large regular grid graph approximation methods and evaluating their performance on real-world orchard single orchard blocks. This chapter expands upon the approaches developed in previous chapters by integrating insights from regular grid graph methods and MILP-based optimisation methods. The regular grid graph approximation methods provide simple lower and upper bounds, offering a foundational understanding of the problem structure. However, they are not always sufficient to obtain high-quality solutions for complex multi-block applications. To address these challenges, an Enumeration-based Set Coverage Algorithm is introduced to tackle the multi-block problem by leveraging the adjacency properties of a multi-layer regular grid graph. Additionally, the chapter explores potential strategies to enhance the proposed OSRP solutions and highlights opportunities for future research in this field.

Figure 6.1: A plot of a real-world orchard consisting of two irregularly shaped blocks

6.1 Enumeration-Based Set Coverage Algorithm

Chapter 5 explores applications across three types of single orchard blocks, each characterized by distinct structural layouts. To establish lower bounds, a small regular grid graph approximation was introduced, while a large regular grid graph approximation provided upper bounds in Chapter 5. However, real-world orchards are often more complex, typically consisting of multiple interconnected blocks (as shown in Fig. 6.1), which may include various block types, further increasing the complexity of finding optimal solutions.

To address these challenges, this chapter introduces an Enumeration-Based Set Coverage Algorithm for solving multi-block orchard problems. Additionally, the MIP methods described in Chapter 3 are employed for comparison in addressing the multi-block scenarios.

6.1.1 Enumeration of Paths

The real-world orchards consist of a finite number of demand edges, although the number of demand edges increases significantly as the orchard size grows. This chapter introduces an Enumeration-based Set Coverage Algorithm to address the complexity of multi-block OSRP instances. The first step involves enumerating all possible paths for each robot starting from every node. Specifically, the process begins by generating all feasible com-

binations of demand edges that each robot can service, ensuring that the selected combinations satisfy adjacency constraints. In particular, demand edges must be either directly connected or linked by exactly one linking edge. In addition, capacity utilisation must be carefully considered. If the capacity utilisation is too low, the number of possible path combinations increases, which may reduce the efficiency of solution finding process. However, a higher number of combinations also enhances the potential for discovering better solutions.

As discussed in Chapter 4, the optimal solution properties outlined in Section 4.3 demonstrate the block adjacency properties of the OSRP on a multi-layer regular grid graph. Unlike the single-layer problem, where the optimal solution pattern typically sprays all demand edges from top to bottom, the multi-layer structure does not always follow this approach. However, the strategy aims to minimise the total linking cost, with the furthest robots typically exhausting their capacity to reduce the number of robots required and the associated linking travel costs. In some cases, removing the last few demand edges can prevent duplication of unserved demand edge traversal, which has the potential to further reduce the total cost. Therefore, the enumeration of paths should aim to fully utilise the available robot capacity while also considering whether removing several of the last demand edges is beneficial. In this part, we include paths that iteratively remove the last three demand edges for paths of length greater than or equal to three, and remove one demand edge if the length equals two. This approach is intended to enrich the enumeration without excessively increasing the number of paths. The idea is further discussed, and a Dijkstra-based strategy is proposed in Section 6.2.

The main steps for the *enumeration of paths* are as follows:

Step 1 Enumerate paths by starting from *each node* in the graph.

Step 2 For each starting node, begin by exploring all connected demand edges.

Step 3 For each demand edge, extend the path by sequentially connecting adjacent demand edges until the robot's capacity is exhausted. Note that consecutive linking edges are not permitted in this process to preserve adjacency.

Step 4 Remove both the last one and two demand edges, and add the resulting truncated paths to the set of enumerated paths. This step enriches the solution space by potentially reducing the total cost through the elimination of duplicated demand edge traversal when connecting to the depot, without significantly increasing the number of candidate paths. This step is skipped if the path already terminates at the depot.

Step 5 Convert each path into a complete tour by connecting both the start and end nodes to the depot using Dijkstra's algorithm.

Algorithm 6.1 illustrates the method for generating all feasible paths starting from a *given node*, subject to capacity constraints. This process is then repeated from each node to enumerate the complete set of candidate paths. Finally, Steps 4 and 5 enrich this set by including additional paths obtained by removing the last few demand edges that may help reduce the total cost.

Algorithm 6.1 Enumeration of feasible paths from a starting node i

Require: Start node i, capacity P, demand matrix $\mathcal{D}$, cost matrix $\mathcal{C}$, depot node n_0
Ensure: Set of feasible paths $\mathcal{T}$ starting from node i

$\mathcal{T} \leftarrow \emptyset$ ▷ Initialise set of feasible paths
function DFS(u, p, d, `zero_used`, $\mathcal{V}$)
  if $u = n_0$ **and** $p \notin \mathcal{T}$ **then**
    Add p to $\mathcal{T}$
    return
  for each $v \in N$ such that $\mathcal{C}[u, v] > 0$ **do**
    $e \leftarrow (\min(u, v), \max(u, v))$, $e_d \leftarrow \mathcal{D}[u, v]$
    $\mathcal{V}' \leftarrow$ copy of $\mathcal{V}$, $d_{\text{new}} \leftarrow d$
    if $e_d > 0$ and $e \notin \mathcal{V}'$ **then**
      Add e to $\mathcal{V}'$, $d_{\text{new}} \leftarrow d + e_d$
      `zero_next` $\leftarrow$ `false` ▷ Step 2: Extend via unvisited demand edge
    else if `zero_used` = `false` **then**
      `zero_next` $\leftarrow$ `true` ▷ Step 3: Allow one linking edge if needed
    else
      continue
    if $d_{\text{new}} \leq P$ **then**
      Append (u, v) to p'
      DFS(v, p', d_{new}, `zero_next`, $\mathcal{V}'$)
    else
      if $p \notin \mathcal{T}$ **then**
        Add p to $\mathcal{T}$
DFS(i, [], 0, `false`, $\emptyset$) ▷ Step 1: Start from node i
Add truncated versions of each path by removing the last 1 or 2 demand edges ▷ Step 4: Truncate to enrich path set
for each path $p \in \mathcal{T}$ **do**
  Connect start and end of p to depot n_0 using Dijkstra's algorithm ▷ Step 5: Reconnect to depot
  Update total cost of p
return $\mathcal{T}$

In Step 5, several methods can be employed to construct tours and calculate the total cost when assigning a group of demand edges to robots that neither start from nor return directly to the depot. One such method involves solving a CARP, which aligns with the optimisation model under the setting $\mathcal{R} = 1$. In this case, the start and end nodes of the enumerated paths may be adjusted, as the objective is to find the most cost-effective way to cover all demand edges. This can result in different route endpoints from those originally enumerated. However, this approach is computationally expensive due to

the large number of candidate paths generated during enumeration, which significantly increases the time required to compute optimal routes and their associated costs.

A more straightforward approach is the *Connection Strategy*, which retains the entire enumerated path assigned to a robot without altering its start or end nodes. If the start or end node is not directly connected to the depot, Dijkstra's shortest path algorithm is used to connect the start node to the depot and the end node back to the depot, forming a complete route that covers the assigned demand edges. The total cost is then calculated by summing the costs of all individual path segments. While this method is computationally efficient and easy to implement, it may result in slightly higher costs for certain routes compared to more complex optimisation-based approaches.

Within these enumerations, some paths service the same set of demand edges but follow different traversal patterns, such as covering the edges in opposite directions or generating identical paths through edge removal operations. Therefore, we only save the minimum-cost path based on the paths that service the same unique set of demand edges. It first extracts demand edges from each path in $\mathcal{P}$ and stores them in a dictionary. Then, it compares paths serving the same demand edge set and retains only the one with the lowest cost. This refinement reduces redundant enumerations while preserving all distinct candidate paths corresponding to unique sets of demand edges.

6.2 Optional Strategies for Reduced Path Enumeration

As orchard size increases, the number of enumerated paths grows significantly, leading to higher computational costs. In this section, we introduce strategies corresponding to Steps 2, 3, and 4, which serve as optional techniques or constraints designed to reduce the number of enumerations. While these strategies may result in a slight reduction in solution quality, they are included to offer flexibility, enabling users to balance the trade-off between computational efficiency and solution accuracy.

6.2.1 Spraying the Furthest Demand Edge First

In the original path enumeration strategy (Step 2), for each starting node, the algorithm begins by exploring all directly connected demand edges without prioritisation. As an optional strategy to reduce the number of enumerated paths, we propose prioritising the *furthest demand edges* from the depot during path construction, based on Dijkstra's shortest path strategy. This approach reduces the number of enumerated paths and helps eliminate redundancy by encouraging outward exploration and avoiding repeated traversal

near the starting node. For example, if the depot is located at the bottom-left corner and two demand edges are connected bidirectionally to the starting node, the algorithm always selects the upward direction, as it is further from the depot.

6.2.2 Following Linking Edges Toward the Depot Side

In the original path enumeration strategy (Step 3), for each demand edge, the algorithm extends the path by sequentially connecting adjacent demand edges until the robot's capacity is exhausted. To preserve adjacency, consecutive linking edges are not permitted in this process.

As an optional strategy to reduce the number of enumerated paths, we propose guiding the path extension along the direction of the linking edges toward the depot side (i.e., left or right, depending on the depot's location). For example, if the depot is located in the bottom-left corner, the direction refers to leftward movements along linking edges. This strategy guides paths more efficiently toward the depot and reduces redundant traversals in the enumeration process.

6.2.3 Enriching the Enumeration by Adding Truncated Paths

In the original path enumeration strategy (Step 4), the last one or two demand edges are removed, and the resulting truncated paths are added to the set of enumerated paths. This step enriches the solution space by potentially reducing the total cost, particularly by eliminating duplicated traversal of demand edges when connecting to the depot, without significantly increasing the number of candidate paths. This step is skipped if the path already terminates at the depot.

However, limiting removal to only one or two edges may be overly simplistic. Therefore, as a generalisation of this idea, we propose an optional strategy that iteratively removes the last few demand edges, guided by the shortest path from the end node to the depot, computed using Dijkstra's algorithm. This process continues until no further cost reduction is observed. For robots whose end node is already directly connected to the depot, no removal is performed. To ensure path adjacency, consecutive linking edge coverage is not permitted during this truncation process.

6.3 Set Coverage Optimisation Model

In the final step, a set coverage optimisation model is applied to determine the optimal combination of potential paths that ensure all demand edges in the orchards are covered in at least one direction. The set $\mathcal{P}$ represents the collection of enumerated demand edge

combinations, where each path $p \in \mathcal{P}$ consists of a subset of edges covered by a single robot. The decision variables are binary and defined as:

$$z_p = \begin{cases} 1, & \text{if path } p \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

The objective is to minimize the total cost of the selected paths:

$$\min \sum_{p \in \mathcal{P}} C_p z_p \tag{6.1}$$

where C_p represents the total cost of path p, including the travel cost from the depot to the start of the path and back to the depot. The depot-related cost is computed using Dijkstra's algorithm to ensure an accurate cycle cost.

Each demand edge must be covered by at least one selected path in at least one direction:

$$\sum_{p \in \mathcal{P}: E_p \ni e} z_p \geq 1, \quad \forall e \in \mathcal{D} \tag{6.2}$$

where:

$\mathcal{D}$ is the set of all demand edges with $i < j$.

E_p the set of *undirected* edges visited at least once in path p.

It would also be possible to explore alternative approaches to further improve the solution, including:

(a) Using the set coverage solution as an initial feasible solution for the MIP introduced in Chapter 3, which incorporates both exact and heuristic algorithms to solve the SDCARP.

(b) Solving restricted MIPs by considering only a subset of edges identified from the set coverage solution.

(c) Constructing a coverage MIP with additional y variables to track the allocation of spray to edges within a path, allowing split deliveries. However, this approach may be computationally expensive, as it introduces $\sum_{p \in \mathcal{P}} |E_p|$ additional continuous variables, where E_p includes all edges traversed, even those not sprayed.

(d) Implementing column pricing to generate additional paths that could enhance the solution, even if they do not initially meet the connectivity constraints used in the original path generation.

(e) Incorporating 'big' routes that contain one more demand edge than a single robot can fully service. This approach ensures that at least one edge is double-covered per such 'big' route in the coverage solution by modifying the right-hand side constraint to $z_p \geq 1 + s_e$, with binary variables s_e and the condition $\sum_e s_e \geq (\sum_{p \in big} z_p) - 1$. The rationale is that any robot exceeding its capacity and requiring a split must have at least one edge double-covered. Additionally, a constraint such as $\sum_p z_p \geq \lceil \sum_e D_e/P \rceil$ can be included to ensure coverage feasibility. While these constraints are necessary but not sufficient, feasibility of demand allocation to edges can be easily verified for integer solutions, with any infeasible solutions eliminated using lazy constraints.

We applied the MIP method to instances Ins.H and MB1, using the set cover solution as a warm-start, with the aim of optimising the routes assigned to robots that cover the same demand edges. The optimisation took over 100 seconds and resulted in only minor improvements (0.253% and 0.11%, respectively). Due to the high computational cost and limited gains, this approach is not pursued further in this chapter. Future work may explore more efficient methods to improve solution quality within practical time limits.

The Figure 6.2 illustrates the solutions obtained using the Enumeration-based Set Coverage Algorithm method for the "Ins.H" instance, where different colours represent distinct robot path patterns. As observed in Figure 6.2, the solution follows an adjacent pattern for a single block. However, in a multi-block scenario, as shown in Figure 6.3, the solution becomes more complex.

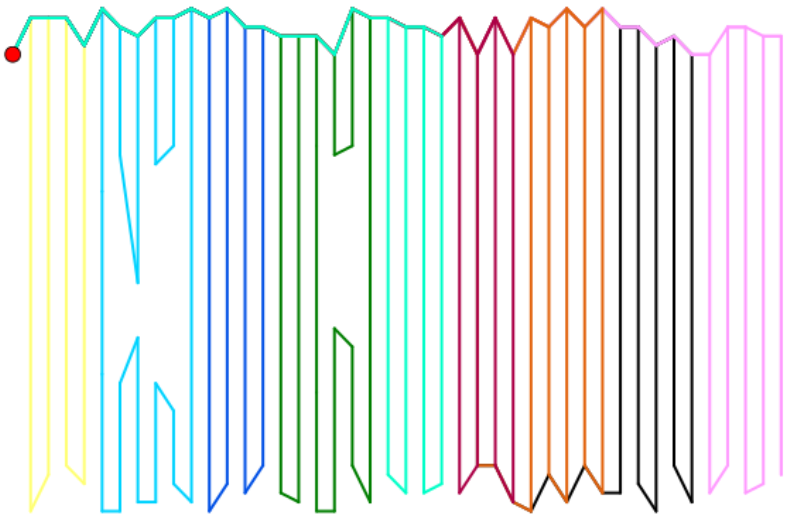


Figure 6.2: The solution obtained using the Enumeration-based Set Coverage Algorithm for "Ins.H", where different colours represent different robot path patterns, and the depot is highlighted in red.

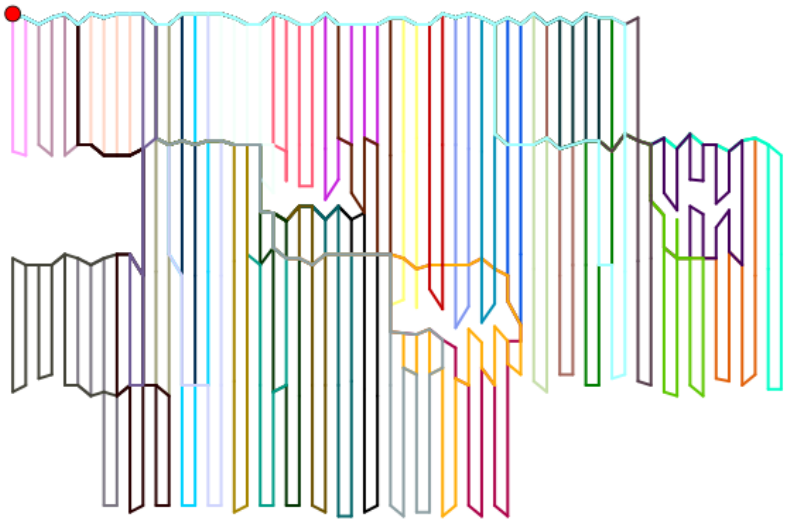


Figure 6.3: The solution obtained using the Enumeration-based Set Coverage Algorithm for "MB4", where different colours represent different robot path patterns, and the depot is highlighted in red.

6.3.1 Numerical Results

In this section, the performance of the solution techniques, including the Enumeration-based Set Coverage (with a Gurobi work limit of 3600 units, approximately one hour) and MIP methods, is compared in Table 6.1. Additionally, a valid lower bound method is provided and proven in Proposition 6.3.1.

This chapter introduces a set of multi-block datasets generated based on real-world properties. Specifically, the vehicle capacity is approximately 5.2 times the demand of each demand edge across all datasets. Additionally, the cost of a linking edge is approximately $\frac{1}{90}$ of the demand edge cost, reflecting real-world conditions. To enhance realism, adjustments are made to the cost (increased by a factor of 10) and demand (increased by a factor of 18) in the "Ins.H" dataset. The datasets "MB1" to "MB5" exhibit distinct characteristics, varying in the number of blocks, edges, and obstacles. For example, Fig. 6.4 illustrates the shape of "MB5", highlighting the key properties of the generated dataset based on real-world conditions, including uneven edges and obstacles.

For the Small Regular Grid Graph Approximation method, as proposed in Chapter 5 and stated in Propositions 5.2.1 and 5.2.2, its applicability to multi-block orchards is limited due to the complex layout and interconnections between blocks. However, based on the block adjacency pattern lower bound shown in Lemma 4.3.1 in Chapter 4, an adapted version suitable for the current scenario is presented in Proposition 6.3.1.

Proposition 6.3.1. For the OSRP with the largest layer containing M demand edges

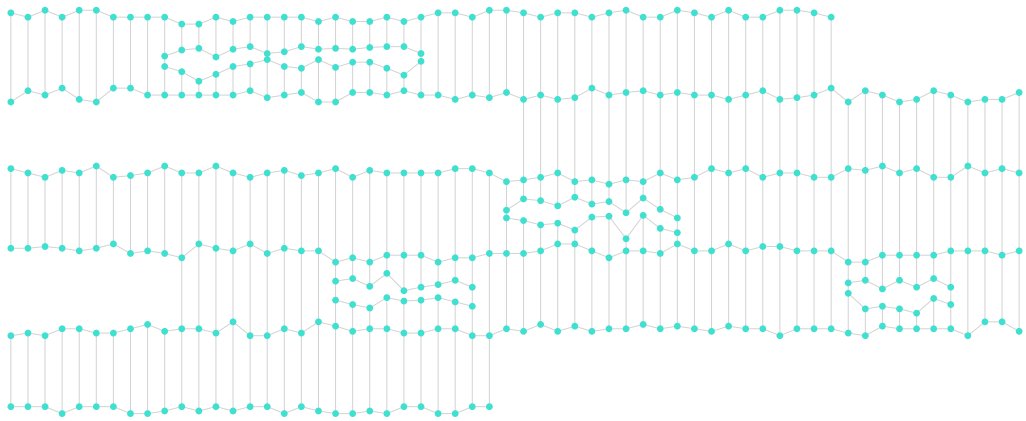


Figure 6.4: The shape of MB5. This graph illustrates the main properties of the generated dataset based on real-world conditions, featuring uneven edges and obstacles.

across L layers and a capacity of P, the total demand corresponding to the same vertical index is aggregated and denoted as $D_1, D_2, \ldots, D_M$. The minimal linking edges of a group are represented as $C_1, C_2, \ldots, C_{M-1}$. A **lower bound (LB)** for the problem can be calculated as follows:

$$\text{Let} \quad R_k = \min \left\{ j \mid \sum_{i=j}^{N} D_i \geq kP \right\}.$$

$$\text{Then } f_{|\mathcal{R}|-k} \geq M - R_k,$$

$$\text{and} \quad A_r = 2 \sum_{i=1}^{f_r - 1} C_i.$$

$$\text{Thus } LB = \sum_{r \in \mathcal{R}} A_r + \sum_{i=1}^{M} D_i.$$

Proof. The index of the furthest demand edge across all layers is M. Consequently, the furthest index reached by the robots is determined by this value, with the robots ordered

based on the index of the furthest edge, as referenced in Assumption 4.2.1.

$$\begin{aligned} f_{|\mathcal{R}|} &= M \\ f_{|\mathcal{R}|-1} &\geq M - R_1 \\ f_{|\mathcal{R}|-2} &\geq M - R_2 \\ &\vdots \\ f_{|\mathcal{R}|-k} &\geq M - R_k \\ &\vdots \\ f_1 &\geq M - R_{|\mathcal{R}|-1} \end{aligned}$$

Then, the furthest index of robots in any optimal solution satisfies

$$f_r^* \geq f_r$$

The total travel cost consists of the cost of travelling the linking edges (A_r^*) and the cost of demand edges. The corresponding lower bound on the travel cost of linking edges is:

$$A_r^* \geq 2 \sum_{i=1}^{f_r - 1} C_i = A_r$$

Thus,

$$\text{Optimal cost} \geq \sum_{r \in \mathcal{R}} A_r + \sum_{i \in \{1,2,\ldots,M\}} D_i$$

□

In Proposition 6.3.1, for datasets with randomly placed obstacles, the total demand along a vertical edge is computed by summing the demand from all vertical edges sharing the same vertical index. Additionally, selecting the minimum linking cost among a group of linking edges provides a valid lower bound, as illustrated in Fig. 6.5.

For the Large Regular Grid Graph Approximation method discussed in Chapter 5, which follows an adjacent solution pattern for the single-layer problem, this pattern's solution can be considered one of the feasible solutions in Section 6.1.1. Thus, the comparison of the upper bound focuses only on the solutions obtained from the MIP and Enumeration-based Set Coverage Algorithm.

The MIP method, which incorporates lazy constraints, additional constraints, symmetry elimination, and heuristic techniques, is discussed in Chapter 3. It is warm-started using a feasible solution generated by the Enumeration-based Set Coverage Algorithm. Although the set cover solution may involve over-coverage, where certain demand edges

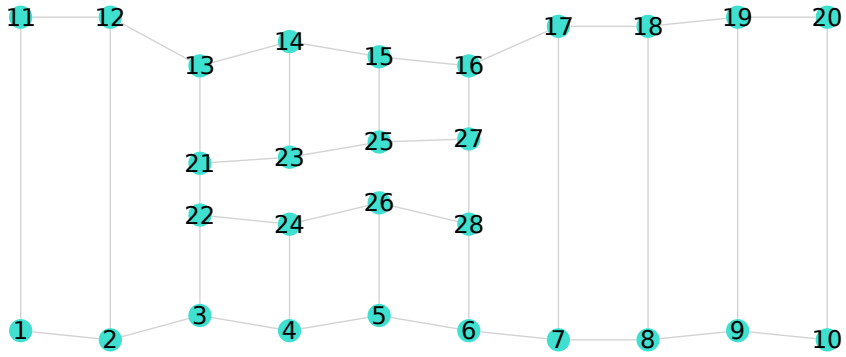


Figure 6.5: In the lower bound process, as shown in Proposition 6.3.1, the total demand of an edge is obtained by summing the demands of corresponding vertical edges. For example, the demand of edge (4,14), denoted as D_4, is calculated by adding the demands of edges (4,24) and (14,23). The minimal linking cost for C_3 is determined as the minimum value among the edges (3,4), (22,24), (21,23), and (13,14).

are covered more than once, it remains a feasible solution with complete routing plan information. This solution is used to initialise the group of variables x_{ij}, which are fed into Gurobi using the callback technique described in Section 3.3.4. The initial set cover solution provides a collection of feasible paths, which are then assigned to robots and further optimised during the MIP solving process. A time limit of 3600 seconds (approximately one hour) is imposed on Gurobi, and the solver is run with seed values ranging from 1 to 3. While different seeds may yield slightly different solution paths, the differences are generally minor. For consistency, the results reported correspond to seed = 1.

Table 6.1: Comparison of results obtained using the MIP solution method. The "UB" column reports the corresponding upper bound, while "LB" reports the lower bound. The "Lower Bound" column presents the lower bound from Proposition 5.2.2. The "EnuCover" column displays results from the Enumeration-Based Set Coverage Algorithm introduced in Section 6.1.1. The "|Enu|" column reports the number of enumerated paths, and the "CPU Time" column presents the computation time (in seconds) required by EnuCover. For MIP, the Gurobi solver's WorkLimit parameter is set to 3600 seconds (i.e., one hour). The "Gap" column represents the relative difference between the best upper and lower bounds.

	LB		UB		|Enu|	CPU Time (s)	Gap
	Lower Bound	MIP	EnuCover	MIP	EnuCover	EnuCover	$\frac{UB^*-LB^*}{UB^*}$
Ins.H	2354.600	2782.000	3087.300	3079.500	291	17.20	9.661%
MB1	7188.652	7631.926	9011.989	9002.189	4268	26.07	15.221%
MB2	12508.092	15067.876	18481.506	18480.082	36482	693.71	18.464%
MB3	14248.360	16251.845	20644.844	20605.658	27518	285.50	21.129%
MB4	19658.550	25026.042	32908.371	32864.643	86555	3979.71	23.851%
MB5	21632.608	27143.914	44792.775	44791.776	102718	42796.94	39.400%

Table 6.1 shows that the Enumeration-based Set Coverage Algorithm provides a reasonable upper bound, which is then used as a warm start for the MIP method. While MIP has limited capacity to significantly improve the solutions, it achieves a better lower bound. However, MIP is computationally expensive and fails to reach optimality within the predefined time limit of approximately one hour. The "MB5" instance is particularly complex, making it difficult to solve. Even the set coverage problem reaches its computational limit, with a gap of 0.387%. The 0.387% gap in the set coverage problem represents the difference between the best integer solution found and the lower bound. While relatively small, reducing this gap could lead to slight improvements in the overall solution quality.

A comparison of the Enumeration-based Set Coverage Algorithm on the single-block instances, as described in Chapter 5, is presented in Table 6.2. This comparison highlights the performance of different methods in terms of upper bound quality. As shown in the table, the Enumeration-based Set Coverage Algorithm yields results that are better than those obtained by the Large Regular Grid Approximation introduced in Section 5.3, while requiring a relatively small number of enumerated paths. Although its upper bounds are slightly worse than the best results, which are computed as the minimum of the objectives from the Large Grid and Simple Heuristic methods presented in Chapter 5, for instances *RegGrid* and *CostVar*, the enumeration-based approach still provides a practical and scalable solution framework. Additionally, we incorporated the best solution paths from the Large Grid and Simple Heuristic methods into the set of enumerated paths to assess whether the solution quality could be improved. While this enriched enumeration leads to a slight improvement in solution quality, the gains are minor. Nevertheless, the method remains efficient across diverse instance types and serves as a strong baseline or a valuable component in hybrid algorithm designs that aim to balance solution quality with computational cost.

As discussed in Section 6.2, several strategies were introduced to improve the efficiency of the path enumeration process. The term "Reduced_Enu" refers to the combined application of all these strategies. Table 6.3 presents a comparative analysis of the original Enumeration-based Set Coverage Algorithm ("EnuCover") and its reduced variant across instances Ins.H and MB1 to MB5. As shown in the table, the use of reduction strategies substantially decreases the number of enumerated paths, particularly in larger instances. For example, in instance MB5, the number of paths drops from over 100,000 to just over 10,000. This reduction results in a dramatic improvement in computational time. In MB5, the time decreases from 42,796.94 seconds to 1,668.66 seconds, which corresponds to a 96% reduction. Similar improvements are observed in MB4 (a reduction of 88%) and MB2 (approximately 88%). In contrast, smaller instances such as Ins.H and MB1 show

Table 6.2: Comparison of results obtained using the Enumeration-based Set Coverage Algorithm to solve single-block instances, as discussed in Chapter 5. "$UB_{\text{Large Grid}}$" denotes the upper bound from the Large Regular Grid Approximation introduced in Section 5.3; "UB_{Best}" refers to the best upper bound obtained by taking the minimum of the objectives from the Large Grid and Simple Heuristic methods presented in Chapter 5; "UB_{EnuCover}" is the upper bound obtained from the Enumeration-based Set Coverage Algorithm; "$UB^{\text{warm}}_{\text{EnuCover}}$" denotes the upper bound obtained by enriching the enumeration with solution paths from the Large Grid and Simple Heuristic methods; and "|Enu|" indicates the number of enumerated paths.

	$UB_{\text{Large Grid}}$	UB_{Best}	|Enu|	UB_{EnuCover}	$UB^{\text{warm}}_{\text{EnuCover}}$
RegGrid	163 039.17	161 258.24	185	162 017.24	160 829.61
CostVar	169 047.28	167 990.61	471	168 388.79	167 983.29
Obs	145 481.76	145 481.76	470	144 663.00	144 663.00

limited or negligible runtime gains, as the original enumeration sizes are already relatively small. However, this gain comes at the cost of a slight deterioration in solution quality, as reflected in the marginal increase in upper bound values.

Therefore, the choice of whether to apply these strategies should depend on the specific priorities of the problem at hand. When computational efficiency is a key concern, the full application of reduction strategies is advantageous. Conversely, if preserving solution quality is more important, selectively applying only a subset of the strategies may provide a better trade-off between runtime and performance.

Table 6.3: Comparison of the Enumeration-based Set Coverage Algorithm with and without reduction strategies on instances Ins.H and MB1 to MB5, as discussed in Section 6.2. "Reduced_Enu" refers to the application of all strategies combined. "EnuCover" refers to the original Enumeration-based Set Coverage Algorithm introduced in Section 6.1.1. "|Enu|" indicates the number of enumerated paths, and "CPU Time" reports the computation time (in seconds) required by EnuCover.

	UB		|Enu|		CPU Time (s)	
	EnuCover	Reduced_Enu	EnuCover	Reduced_Enu	EnuCover	Reduced_Enu
Ins.H	3087.300	3321.500	291	254	17.20	17.73
MB1	9011.989	9267.741	4268	1269	26.07	19.83
MB2	18481.506	19576.038	36482	4877	693.71	86.58
MB3	20644.844	21945.068	27518	4527	285.50	200.87
MB4	32908.371	33512.769	86555	10779	3979.71	476.43
MB5	44792.775	47969.720	102718	10265	42796.94	1668.66

6.4 Conclusions and Future Work

This chapter focused on methods for solving multi-block OSRP applications, which are more complex and challenging to address. An Enumeration-based Set Coverage Algorithm was proposed to tackle the problem, along with a discussion of various algorithms that can be applied to these complex cases. The MIP method was initialised with a warm start from the Enumeration-based Set Coverage Algorithm to solve the problem.

To validate the solution approaches, six orchard graph examples with realistic features were introduced. The numerical results demonstrated the effectiveness of the proposed solution method, showing that the Enumeration-based Set Coverage Algorithm is computationally more efficient than MIP. However, while MIP is more computationally expensive, it is limited in its ability to improve the solution but provides a better lower bound.

As discussed in Section 6.1.1, it is possible to explore alternative approaches to further improve the solution quality, such as using restricted MIPs that consider only a subset of edges identified from the set coverage solution. This ensures that each robot follows an efficient, connected path while reducing unnecessary travel. Given the computational challenges, strategies such as restricted MIPs, column generation, and the introduction of 'big' routes can help enhance solution quality. While these methods offer potential improvements, their effectiveness depends on achieving a balance between computational efficiency and solution accuracy. Future research could also explore advanced heuristics, adaptive optimisation techniques, or hybrid models to improve scalability and performance, particularly for larger problem instances. Additionally, incorporating split-delivery properties may further improve solution flexibility and cost-effectiveness.

Chapter 7

Conclusion

The Orchard Spraying Routing Problem (OSRP) is inspired by a real-world orchard spraying application, where achieving high-quality solutions remains a challenge in agricultural spraying areas. This thesis focuses on solving the OSRP by optimising a fleet of identical robotic sprayers to perform spraying tasks on demand edges within the orchard while meeting capacity and demand constraints.

In Chapter 1, the properties of orchards were described. The OSRP is presented as a real-world application of the Split Delivery Capacitated Arc Routing Problem (SDCARP), a variation of the Capacitated Arc Routing Problem (CARP). The classical SDCARP is an NP-hard problem [37], making it challenging to obtain optimal solutions. In the context of the OSRP, the network structures of orchards differ from those studied in other SDCARP solution approaches. Moreover, tasks such as spraying, planting, and harvesting occur regularly, with routing problems playing a crucial role in agricultural applications. This highlights the significance and value of research on the OSRP. This research not only aligns with the goals of Precision Agriculture (PA) but is also well-suited to the era of automated robotics, underscoring its significance for advancing agricultural practices.

Real-world orchards exhibit diverse layouts, adding complexity to the spraying problem. In particular, some orchards may have large demand edges where the spraying demand exceeds the robot's capacity, although in most cases, the demand remains within the robot's capacity. Consequently, two Mixed-Integer Linear Programming (MILP) models were developed to address the OSRP: one tailored for basic spraying applications and another designed for scenarios with large demand edges, as detailed in Chapter 3. Theorems and mathematical proofs were provided to demonstrate the properties of the OSRP used in formulating the large demand model, offering valuable mathematical insights into the spraying routing problem. Specifically, the key idea behind the large demand model involves classifying robots into two groups: those that spray only one edge and those that spray multiple edges. As proven in the theorems, the number of robots spraying multiple

edges is fewer than the total number of demand edges. This classification helps reduce the number of variables, thereby enhancing the efficiency of finding solutions.

In Chapter 3, both exact and heuristic methods were developed to solve the two models of the OSRP. The exact algorithms incorporate lazy constraints, which are useful when the full set of constraints is too large to include in the initial formulation. The main idea is to exclude costly constraints initially and reintroduce them only if the relaxed solution violates any omitted constraints. To further improve quick utilisation and efficient variable branching, new constraints were introduced. These include a set of variables representing the total number of times robots pass each edge, preventing fractional solutions for the newly introduced variables. Rounding cuts were also applied during the branch-and-bound process to accelerate the search for optimal solutions. Furthermore, symmetry elimination constraints were incorporated by renumbering the robots and orienting the routes to remove redundancies caused by symmetrical solutions. The exact algorithms are unable to fully solve complex NP-hard problems, as noted in various studies in Chapter 2. To address this, a heuristic repair technique was proposed for the OSRP. This approach constructs feasible solutions from infeasible ones, serving as new incumbents to accelerate the solution-finding process and deliver high-quality, practical solutions for the OSRP.

In Chapter 4, the regular grid graph layout of the orchard was analysed, leading to the exploration of the OSRP on specific network structures. The structure of regular grid graphs differs from that of other existing CARP and SDCARP datasets, creating a distinct special case of the OSRP, referred to as the Orchard Spraying Routing Problem on Regular Grid Graphs (OSRP_{RG}). An adjacent solution pattern was introduced for the OSRP_{RG}, supported by theorems, lemmas, and mathematical proofs to provide deeper mathematical insights into the OSRP_{RG}. These findings confirm the existence of polynomial-time solvable CARP cases and highlight the potential to solve the OSRP_{RG} within polynomial time.

Chapter 4 introduced two groups of regular grid graphs, classified based on their layout information layers. Single-layer graphs consist of only one layer, while multi-layer regular grid graphs comprise multiple layers, significantly increasing the complexity of finding optimal solutions. To address the OSRP_{RG}, a method was proposed to construct adjacent solutions for single-layer graphs and block-adjacent solutions for multi-layer graphs. These constructions enable the generation of optimal solutions under specific conditions when the graph layout meets the criteria of a regular grid graph structure. Counterexamples are presented to highlight the limitations and provide deeper insights into the properties of the adjacent structure when these conditions are not met. Furthermore, these insights enhance the understanding of the problem's complexity and inform the development of more efficient algorithms for solving OSRP_{RG} cases.

Chapter 5 established the connection between mathematical models and real-world applications. This chapter focuses on the practical aspects of solving the OSRP. Based on the properties of the regular grid graph, two approximation methods are proposed: a small regular grid graph method, which provides a lower bound, and a large regular grid graph method, which provides an upper bound. Both methods are straightforward and practical. Additionally, a simple heuristic approach is introduced, prioritising capacity utilisation while leveraging the structural properties of regular grid graphs. Three real-world orchard OSRP instances are tested using the algorithm, and the numerical results demonstrate the effectiveness and efficiency of the proposed method in solving practical OSRP.

Different from Chapter 5, which focuses on solving single-block OSRP, Chapter 6 explores algorithms for handling more complex multi-block cases. An Enumeration-based Set Coverage Algorithm is proposed and integrated with the MIP method to address these challenges. Additionally, several datasets are generated to incorporate real-world properties, demonstrating the effectiveness of the proposed method in solving OSRPs.

This thesis not only models real-world complex OSRP and provides mathematical proofs for solving the problem but also applies both exact and heuristic methods to tackle OSRP efficiently. The integration of real-world applications validates the mathematical insights and theoretical proofs, demonstrating the practicality and effectiveness of the proposed solution methods.

This research establishes a strong connection between mathematical optimisation and real-world agricultural operations, with the potential for extension to other agricultural domains. One of the key challenges in this field is the lack of comprehensive agricultural datasets. Many classical models depend on randomly generated datasets or are designed for other applications, making them less suitable for agricultural spraying scenarios. To address this gap, this study provides real-world datasets and also generates specifically tailored datasets for testing and benchmarking spraying routing problems, enriching the available data resources for future research and practical applications. In future work, we intend to compile and publish a definitive set of benchmark instances for orchard problems based on the datasets used in this study. This would facilitate reproducibility and further research in this area.

Furthermore, the methods proposed in this thesis have the potential to be integrated with other algorithms, making them applicable to a broader range of logistics and transportation problems. By bridging mathematical modelling, real-world implementation, and algorithmic innovation, this work provides a foundation for further advancements in automated agricultural vehicle routing and beyond.

This thesis has explored SDCARP in orchard spraying operations, demonstrating its

advantages over traditional CARP formulations. The results highlight efficiency improvements through split deliveries, optimised vehicle routes, and reduced operational costs.

Despite the promising results, this study has several limitations. One key challenge is the ongoing trade-off between solution quality and computational efficiency, particularly for large-scale or complex instances. A promising direction for future research is the incorporation of regular grid graph block adjacency solutions with the split delivery properties of SDCARP, leveraging the structured layout of orchards to enhance solution quality. The current regular grid graph solutions do not fully benefit from split deliveries, and tailored heuristics for grid-shaped orchards could improve efficiency, such as through the combined use of decomposition methods. Decomposition methods, widely used to handle large-scale optimisation problems. A hybrid approach combining heuristic or enumeration-based set covering with Large Neighbourhood Search (LNS) could refine solutions by analysing adjacency relationships and optimising non-adjacent components separately, improving computational efficiency and solution quality.

Another important direction for future work is developing a compact model to prevent exponential growth in problem size. Compact models enhance computational efficiency, enabling faster and more scalable solutions with modern optimisation solvers. This would be particularly useful for applying mixed-integer programming methods and the set coverage model used in this thesis. By incorporating grid-based heuristics, decomposition strategies, and compact models, future studies can further refine and extend these methodologies to address the increasing complexity of real-world agricultural logistics.

This study provides a foundation for extending the problem to more complex applications, such as mobile nurse cart scheduling. A nurse cart is a support vehicle used in agricultural operations to resupply autonomous spray robots with essential resources such as chemicals, batteries, or water. Coordinating the movement and scheduling of nurse carts alongside spray robots introduces significant logistical challenges, especially in large-scale or multi-block orchards. The Enumeration-based Set Coverage Algorithm could be equally effective in determining the shortest path between different locations where nurse carts may wait for a period of time. This approach naturally extends to routing spray robots given a predetermined nurse cart schedule. The nurse cart scheduling problem can also be framed as a split-delivery problem on a modified graph. Major linking edges represent nurse cart travel routes, while spray robots operate between pairs of demand edges before refilling. In this framework, the routing of spray robots depends on the spatial arrangement of demand rows: If a demand row is adjacent to only one orchard road, the routing decision for spray robots is trivial, as they must follow that road. If a demand row is located between two orchard roads, an allocation decision must be made regarding which road the robots should use. This decision could be optimised through the

model or determined arbitrarily in advance. In this way, the upper-level problem could be formulated as an SDCARP, where nurse carts must be routed efficiently along major linking edges to supply robots. The lower-level robot routing then becomes straightforward if orchard roads for spraying are predetermined. This hierarchical structure allows for a more flexible and scalable approach to optimising robotic spraying operations.

In conclusion, this research models the practical and complex OSRP application as a MILP within the SDCARP framework. It develops optimisation techniques that integrate exact and heuristic methods while leveraging mathematical insights and providing theorem proofs. These techniques address various scenarios across different types of applications. The proposed solution methods not only deliver high-quality solutions but also enrich datasets for future testing and studies, thereby strengthening the connection between mathematical models and agricultural applications. Looking ahead, future research could focus on enhancing heuristic and enumeration-based set covering methods, as well as conducting an in-depth study of split delivery properties. Furthermore, the current study could be extended to tackle more complex scenarios, such as nurse car scheduling problems.

Bibliography

[1] Abousleiman, R. and Rawashdeh, O. (2014). Tabu search based solution to the electric vehicle energy efficient routing problem. In *2014 IEEE Transportation Electrification Conference and Expo (ITEC)*, pages 1–6. IEEE.

[2] Amaya, C., Langevin, A., and Trépanier, M. (2010). A heuristic method for the capacitated arc routing problem with refill points and multiple loads. *Journal of the Operational Research Society*, 61:1095–1103.

[3] Amponsah, S. K. and Salhi, S. (2004). The investigation of a class of capacitated arc routing problems: the collection of garbage in developing countries. *Waste Management*, 24(7):711–721.

[4] Apollonio, N., Caccetta, L., and Simeone, B. (2004). Cardinality constrained path covering problems in grid graphs. *Networks*, 44(2):120–131.

[5] Arakaki, R. K. and Usberti, F. L. (2019). An efficiency-based path-scanning heuristic for the capacitated arc routing problem. *Computers & Operations Research*, 103:288–295.

[6] Archetti, C. and Speranza, M. G. (2012). Vehicle routing problems with split deliveries. *International transactions in operational research*, 19(1-2):3–22.

[7] Barrientos, A., Colorado, J., Cerro, J. d., Martinez, A., Rossi, C., Sanz, D., and Valente, J. (2011). Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, 28(5):667–689.

[8] Barth, R., Hemming, J., and van Henten, E. J. (2016). Design of an eye-in-hand sensing and servo control framework for harvesting robotics in dense vegetation. *Biosystems Engineering*, 146:71–84.

[9] Belenguer, J. M. and Benavent, E. (2003). A cutting plane algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 30(5):705–728.

[10] Belenguer, J.-M., Benavent, E., Labadi, N., Prins, C., and Reghioui, M. (2010). Split-delivery capacitated arc-routing problem: Lower bound and metaheuristic. *Transportation Science*, 44(2):206–220.

[11] Benavent, E., Campos, V., Corberán, A., and Mota, E. (1992). The capacitated arc routing problem: lower bounds. *Networks*, 22(7):669–690.

[12] Boyacı, B., Dang, T. H., and Letchford, A. N. (2023). Fast upper and lower bounds for a large-scale real-world arc routing problem. *Networks*, 81(1):107–124.

[13] Braekers, K., Ramaekers, K., and Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313.

[14] Brandão, J. and Eglese, R. (2008). A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research*, 35(4):1112–1126.

[15] Campbell, J. F. and Langevin, A. (2000). Roadway snow and ice control. In *Arc Routing*, pages 389–418. Springer.

[16] Carter, A. E. and Ragsdale, C. T. (2006). A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European journal of operational research*, 175(1):246–257.

[17] Cay, A., Kocabiyik, H., and May, S. (2018). Development of an electro-mechanic control system for seed-metering unit of single seed corn planters part i: Design and laboratory simulation. *Computers and Electronics in Agriculture*, 144:71–79.

[18] Chakraborty, S., Elangovan, D., Govindarajan, P., Elnaggar, M., Alrashed, M., and Kamel, S. (2022). A comprehensive reviewl of path planning for agricultural ground robots. *Sustainability*, 14:9156. https://doi.org/10.3390/su14159156.

[19] Chen, Y., Hao, J.-K., and Glover, F. (2016). A hybrid metaheuristic approach for the capacitated arc routing problem. *European Journal of Operational Research*, 253(1):25–39.

[20] Chiang, T.-C. and Hsu, W.-H. (2014). A knowledge-based evolutionary algorithm for the multiobjective vehicle routing problem with time windows. *Computers & Operations Research*, 45:25–37.

[21] Dror, M. (2000). *Arc Routing: Theory, Solutions and Applications*. Springer US, Boston, MA.

[22] Dror, M., Leung, J. M., and Mullaseril, P. A. (2000). Livestock feed distribution and arc traversal problems. *Arc Routing: Theory, Solutions and Applications*, :443–464.

[23] Dror, M. and Trudeau, P. (1989). Savings by split delivery routing. *Transportation Science*, 23(2):141–145.

[24] Edmonds, J. and Johnson, E. L. (1973). Matching, euler tours and the chinese postman. *Mathematical programming*, 5:88–124.

[25] Eiselt, H. A., Gendreau, M., and Laporte, G. (1995a). Arc routing problems, part i: The chinese postman problem. *Operations Research*, 43(2):231–242.

[26] Eiselt, H. A., Gendreau, M., and Laporte, G. (1995b). Arc routing problems, part ii: The rural postman problem. *Operations Research*, 43(3):399–414.

[27] Gan, X., Huo, Z., and Li, W. (2024). Dp-a*: For path planing of ugv and contactless delivery. *IEEE Transactions on Intelligent Transportation Systems*, 25(1):907–919.

[28] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

[29] Golden, B. L. and Wong, R. T. (1981). Capacitated arc routing problems. *Networks*, 11(3):305–315.

[30] Gonzalez-de Soto, M., Emmi, L., Perez-Ruiz, M., Aguera, J., and Gonzalez-de Santos, P. (2016). Autonomous systems for precise spraying–evaluation of a robotised patch sprayer. *Biosystems Angineering*, 146:165–182.

[31] Gracia, C., Velázquez-Martí, B., and Estornell, J. (2014). An application of the vehicle routing problem to biomass transportation. *Biosystems engineering*, 124:40–52.

[32] Grötschel, M., Padberg, M. W., et al. (1985). Polyhedral theory. *The traveling salesman problem*, :251–305.

[33] Guan, M. (1989). Graph theory in china. *Annals of the New York Academy of Sciences*, 576(1):203–218.

[34] Hadlock, F. O. (1977). A shortest path algorithm for grid graphs. *Networks*, 7(4):323–334.

[35] Khazimov, Z. M., Bora, G., Khazimov, K., Khazimov, M., Ultanova, I., and Niyazbayev, A. (2018). Development of a dual action planting and mulching machine for vegetable seedlings. *Engineering in Agriculture, Environment and Food*, 11(2):74–78.

[36] Köhler, E., Liebchen, C., Wünsch, G., and Rizzi, R. (2009). Lower bounds for strictly fundamental cycle bases in grid graphs: Special issue on trees. *Networks*, 53(2):191–205.

[37] Lai, Q., Zhang, Z., Jin, X., and Qin, H. (2018). Forest-based tabu search to the split-delivery capacitated arc-routing problem. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 1237–1242. IEEE.

[38] Lai, Q., Zhang, Z., Yu, M., and Wang, J. (2022). Split-delivery capacitated arc-routing problem with time windows. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2882–2887.

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

[40] LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

[41] Lawler, E. L. (1985). *The Traveling Salesman Problem : A Guided Tour of Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, Chichester.

[42] Lenstra, J. K. and Kan, A. R. (1976). On general routing problems. *Networks*, 6(3):273–280.

[43] Lenstra, J. K. and Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227.

[44] Li, L. Y. and Eglese, R. W. (1996). An interactive algorithm for vehicle routeing for winter—gritting. *Journal of the Operational Research Society*, 47(2):217–228.

[45] Li, X., Giles, D. K., Andaloro, J. T., Long, R., Lang, E. B., Watson, L. J., and Qandah, I. (2021). Comparison of uav and fixed-wing aerial application for alfalfa insect pest control: evaluating efficacy, residues, and spray quality. *Pest Management Science*, 77(11):4980–4992.

[46] Longo, H., De Aragao, M. P., and Uchoa, E. (2006). Solving capacitated arc routing problems using a transformation to the CVRP. *Computers & Operations Research*, 33(6):1823–1837.

[47] López-Santana, E., Méndez-Giraldo, G., and Franco-Franco, C. (2016). A hybrid scatter search algorithm to solve the capacitated arc routing problem with refill points. In Huang, D. and Jo, K., editors, *Intelligent Computing Theories and Application, ICIC 2016, PT II*, volume 9772 of *Lecture Notes in Computer Science*, pages 3–15,

Gewerbestrasse 11, Cham, CH-6330, Switzerland. Springer International Publishing AG.

[48] Lu, L.-C. and Yue, T.-W. (2019). Mission-oriented ant-team aco for min–max mtsp. *Applied Soft Computing*, 76:436–444.

[49] Lubin, M., Dowson, O., Dias Garcia, J., Huchette, J., Legat, B., and Vielma, J. P. (2023). JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, 15:581–589.

[50] Lysgaard, J., López-Sánchez, A. D., and Hernández-Díaz, A. G. (2020). A matheuristic for the minmax capacitated open vehicle routing problem. *International Transactions in Operational Research*, 27(1):394–417.

[51] Mahajan, S., Das, A., and Sardana, H. K. (2015). Image acquisition techniques for assessment of legume quality. *Trends in Food Science & Technology*, 42(2):116–133.

[52] Mahmud, M. S. A., Abidin, M. S. Z., Emmanuel, A. A., and Hasan, H. S. (2020). Robotics and automation in agriculture: present and future applications. *Applications of Modelling and Simulation*, 4:130–140.

[53] Misara, R., Verma, D., Mishra, N., Rai, S., and Mishra, S. (2022). Twenty-two years of precision agriculture: a bibliometric review. *Precision Agriculture*, 23:2135–2158.

[54] Monteiro, A., Santos, S., and Gonçalves, P. (2021). Precision agriculture for crop and livestock farming—brief review. *Animals*, 11(8):2345.

[55] Mu, X., He, L., Heinemann, P., Schupp, J., Karkee, M., and Zhu, M. (2024). Ugv-based precision spraying system for chemical apple blossom thinning on trellis trained canopies. *Journal of field robotics*.

[56] Mullaseril, P. A., Dror, M., and Leung, J. (1997). Split-delivery routeing heuristics in livestock feed distribution. *Journal of the Operational Research Society*, 48:107–116.

[57] Muthukumaran, S., Ganesan, M., Dhanasekar, J., and Babu Loganathan, G. (2021). Path planning optimization for agricultural spraying robots using hybrid dragonfly–cuckoo search algorithm. *Alinteri Journal of Agriculture Sciences*, 36(1):2564–7814.

[58] Muyldermans, L. and Pang, G. (2015). Chapter 10: Variants of the capacitated arc routing problem. In *Arc routing: problems, methods, and applications*, pages 223–253. SIAM.

[59] Oberti, R., Marchi, M., Tirelli, P., Calcante, A., Iriti, M., Tona, E., Hočevar, M., Baur, J., Pfaff, J., Schütz, C., et al. (2016). Selective spraying of grapevines for disease control using a modular agricultural robot. *Biosystems Engineering*, 146:203–215.

[60] Orloff, C. S. (1974). A fundamental problem in vehicle routing. *Networks*, 4(1):35–64.

[61] Ozguven, M. M. and Adem, K. (2019). Automatic detection and classification of leaf spot disease in sugar beet using deep learning algorithms. *Physica A: Statistical Mechanics and Its Applications*, 535:122537.

[62] Panadero, J., Juan, A. A., Ghorbani, E., Faulin, J., and Pagès-Bernaus, A. (2024). Solving the stochastic team orienteering problem: comparing simheuristics with the sample average approximation method. *International Transactions in Operational Research*, 31(5):3036–3060.

[63] Papadimitriou, C. H. (1976). On the complexity of edge traversing. *Journal of the ACM (JACM)*, 23(3):544–554.

[64] Pecin, D. and Uchoa, E. (2019). Comparative analysis of capacitated arc routing formulations for designing a new branch-cut-and-price algorithm. *Transportation Science*, 53(6):1673–1694.

[65] Plessen, M. G. (2019). Optimal in-field routing for full and partial field coverage with arbitrary non-convex fields and multiple obstacle areas. *Biosystems Engineering*, 186:234–245.

[66] Quintero-Araujo, C. L., Caballero-Villalobos, J. P., Juan, A. A., and Montoya-Torres, J. R. (2017). A biased-randomized metaheuristic for the capacitated location routing problem. *International Transactions in Operational Research*, 24(5):1079–1098.

[67] Reichart, M. and Jürgens, C. (2009). Adoption and future perspective of precision farming in Germany: results of several surveys among different agricultural target groups. *Precision Agriculture*, 10:73–94.

[68] Santos, L., Santos, F., Pires, E. S., Valente, A., Costa, P., and aes, S. M. (2020). Path planning for ground robots in agriculture: a short review. In Lau, N., Silva, M., Reis, L., and Cascalho, J., editors, *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC 2020)*, pages 61–66. IEEE.

[69] Sethanan, K. and Neungmatcha, W. (2016). Multi-objective particle swarm optimization for mechanical harvester route planning of sugarcane field operations. *European journal of operational research*, 252(3):969–984.

[70] Shi, Y., Rex, S. X., Wang, X., Hu, Z., Newman, D., and Ding, W. (2019). Numerical simulation and field tests of minimum-tillage planter with straw smashing and strip laying based on edem software. *Computers and Electronics in Agriculture*, 166:105021.

[71] Singh, V. and Misra, A. K. (2017). Detection of plant leaf diseases using image segmentation and soft computing techniques. *Information Processing in Agriculture*, 4(1):41–49.

[72] Soylu, B. (2015). A general variable neighborhood search heuristic for multiple traveling salesmen problem. *Computers and Industrial Engineering*, 90:390–401.

[73] Tang, K., Mei, Y., and Yao, X. (2009). Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 13(5):1151–1166.

[74] Toth, P. and Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

[75] Utamima, A. and Djunaidy, A. (2022). Agricultural routing planning: a narrative review of literature. *Procedia Computer Science*, 197:693–700.

[76] Vakilian, K. A. and Massah, J. (2017). A farmer-assistant robot for nitrogen fertilizing management of greenhouse crops. *Computers and Electronics in Agriculture*, 139:153–163.

[77] Valente, J., Del Cerro, J., Barrientos, A., and Sanz, D. (2013). Aerial coverage optimization in precision agriculture management: A musical harmony inspired approach. *Computers and electronics in agriculture*, 99:153–159.

[78] Volte, G., Bourreau, E., Giroudeau, R., and Naud, O. (2023). Using VRPSolver to efficiently solve the differential harvest problem. *Computers and Operations Research*, 149(106029). https://doi.org/10.1016/j.cor.2022.106029.

[79] Williams, H. A., Jones, M. H., Nejati, M., Seabright, M. J., Bell, J., Penhall, N. D., Barnett, J. J., Duke, M. D., Scarfe, A. J., Ahn, H. S., et al. (2019). Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms. *Biosystems Engineering*, 181:140–156.

[80] Wøhlk, S. and Laporte, G. (2018). A fast heuristic for large-scale capacitated arc routing problems. *Journal of the Operational Research Society*, 69(12):1877–1887.

[81] Xiong, Y., Peng, C., Grimstad, L., From, P. J., and Isler, V. (2019). Development and field evaluation of a strawberry harvesting robot with a cable-driven gripper. *Computers and Electronics in Agriculture*, 157:392–402.

[82] Xu, J., Lai, J., Guo, R., Lu, X., and Xu, L. (2022). Efficiency-oriented mpc algorithm for path tracking in autonomous agricultural machinery. *Agronomy*, 12(7):1662.

[83] Xu, J., Liu, C., Shao, J., Xue, Y., and Li, Y. (2024). Collaborative orchard pesticide spraying routing problem with multi-vehicles supported multi-uavs. *Journal of cleaner production*, 458:142429.

[84] Zachariasen, M. (2001). A catalog of hanan grid problems. *Networks*, 38(2):76–83.

[85] Zhang, G., Ji, C., Wu, Q., Liu, H., Zhou, Y., and Fu, J. (2022). Study on path planning of mechanized harvesting of ratoon rice in the first season based on the capacitated arc routing problem model. *Frontiers in Plant Science*, pages 1–16. https://doi.org/10.3389/fpls.2022.963307.