# HIGH-PERFORMANCE EXACT AND METAHEURISTIC METHODS FOR AGRICULTURAL PATH PLANNING

**FABLIHA ZAHIN**
**Bachelor of Science, BRAC University, 2022**

A thesis submitted
in partial fulfilment of the requirements for the degree of

**MASTER OF SCIENCE**

in

**COMPUTER SCIENCE**

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

HIGH-PERFORMANCE EXACT AND METAHEURISTIC METHODS FOR AGRICULTURAL PATH PLANNING

FABLIHA ZAHIN

Date of Defence: December 12, 2025

| | | |
|---|---|---|
| Dr. Shahadat Hossain | Professor | Ph.D. |
| Dr. Robert Benkoczi | Professor | Ph.D. |
| Thesis Co-Supervisors | | |
| Dr. Saurya Das | Professor | Ph.D. |
| Thesis Examination Committee Member | | |
| Dr. Pascal Ghazalian | Professor | Ph.D. |
| Thesis Examination Committee Member | | |
| Dr. Wendy Osborn | Associate Professor | Ph.D. |
| Chair, Thesis Examination Committee | | |

# Abstract

Strategic operational planning is essential for making agriculture autonomous, while maintaining sustainability. Optimizing how agricultural machinery maneuvers can help reduce costs and greenhouse gas (GHG) emissions, while increasing productivity. This thesis addresses the Capacitated Path Planning Problem (CPP) in agricultural field operations which is a variant of the Vehicle Routing Problem (VRP) adapted for track-based field logistics with refilling constraints. The aim is to minimize non-working distance, the distance traveled by machinery while not performing productive fieldwork, under capacity and coverage constraints. The problem is first formulated as an Integer Linear Program (ILP) and solved using the Gurobi Optimizer, where sub-tour elimination is efficiently implemented through lazy constraints and callback-based separation routines. Then, a metaheuristic, Simulated Annealing (SA), is developed in C++ to overcome the scalability limitations of exact optimization. This implementation features customized neighbourhood operators and a capacity-aware route-splitting mechanism.

Both these methods are tested on real-world benchmark instances. Additionally, a comparative analysis with an Ant Colony Optimization (ACO) implementation demonstrates that the SA achieves near-optimal solutions with significantly reduced runtime compared to the ILP, which produces optimal baselines for small instances. The results highlight a clear trade-off between exactness and computational efficiency and confirm the suitability of metaheuristic approaches for large-scale, refill-aware field routing. This work thus bridges classical routing optimization with precision agriculture by extending CPP to realistic field geometries and providing a scalable framework for autonomous and sustainable agricultural logistics.

# Acknowledgments

Firstly, I would like to express my utmost gratitude to my thesis supervisor Dr. Shahadat Hossain for his unwavering guidance throughout the course of this research. His invaluable insights, patience and expertise in this field have been vital for shaping the direction and quality of this work.

I would also like to profoundly thank my co-supervisor Dr. Robert Benkoczi and my thesis committe members Dr. Saurya Das and Dr. Pascal Ghazalian for their thoughtful feedback and constructive suggestions. Their perspectives have helped me refine my approach at every step.

This work was supported by Verge Ag through the Mitacs Accelerate Program, whose financial and technical assistance made this research possible. This research was conducted in close collaboration with Verge Ag, whose expertise and industry insights were instrumental in supporting and validating the practical aspects of this study. The field image presented in this work is courtesy of Verge Ag.

Lastly, I want to extend my heartfelt appreciation to my friends and family for their endless love and support. Their belief in me and words of encouragement have been my source of strength and motivation throughout this journey.

# Contents

# List of Tables

# List of Figures

# Glossary

**Working Tracks** Parallel (or near-parallel) field paths along which the implement performs productive work (e.g., spraying, seeding, harvesting). Traversal of these tracks constitutes coverage.

**Headlands** Turning and maneuvering areas at field boundaries (or internal turning zones) used for turning and for transitioning between working tracks without damaging crops.

**Track Endpoints** The physical start and end points of a working track. In the graph model, these are represented as vertices that define feasible transitions between tracks.

**Non-Working Distance** Distance traveled while not performing productive work, including headland turns, transitions between tracks, and travel to/from the depot for refilling or unloading.

**Depot** The designated location (on-field or off-field) where tours start and end, and where the vehicle refills or unloads to restore capacity.

**Tour** A depot-to-depot route (one refill cycle) that services a subset of tracks without exceeding the vehicle capacity.

**Capacity ($Q$)** The maximum payload the vehicle can carry (e.g., fertilizer volume, tank capacity) before it must return to the depot to refill or unload.

**Demand ($d_i$)** The material requirement (or collected yield) associated with servicing track $i$.

**Refill Cycle** One operational cycle in which the vehicle departs the depot with full capacity, services tracks until capacity is depleted (or near depleted), and returns to the depot to restore capacity.

**Boustrophedon Pattern** A zigzag "back-and-forth" coverage strategy where adjacent tracks are traversed in alternating directions. It is simple but can be suboptimal when refills, obstacles, or irregular geometries are present.

**Controlled Traffic Farming (CTF)** A management system where machinery is restricted to permanent tramlines to reduce soil compaction across the rest of the field.

**Soil Compaction** Degradation of soil structure due to repeated heavy machinery passes, potentially reducing infiltration, root growth, and yield.

**Turning Radius Constraint** A physical constraint limiting the sharpness of turns a machine can execute, which affects whether a transition between two endpoints is feasible.

**Temperature** ($T$) A control parameter in SA that governs the probability of accepting cost-increasing moves; $T$ decreases over time according to a cooling schedule.

**Cooling Schedule** The rule for reducing temperature in SA (e.g., geometric cooling $T \leftarrow \alpha T$), balancing exploration early and exploitation later.

**Subtour** A closed cycle over a subset of vertices that is disconnected from the depot, yielding an invalid routing solution.

**Subtour Elimination Constraints (SECs)** Constraints that prevent disconnected cycles by forcing all selected arcs to form depot-connected tours.

**Lazy Constraints** Constraints (often SECs) that are not added initially but are generated on demand when a candidate integer solution violates them.

**Feasible Transition** A transition between two vertices that is physically executable given machine kinematics and field geometry (e.g., turning radius); infeasible transitions are excluded or assigned a large penalty.

**Support Graph** Given a solution, the subgraph induced by the selected arcs (e.g., arcs with $x_{ij}^{k} = 1$), used in callbacks to detect disconnected components/subtours.

**Best Bound** The best available lower bound on the optimal objective value produced by an exact solver during the search.

**Incumbent** The best feasible integer solution found so far by an exact solver during optimization.

# Chapter 1

# Introduction

## 1.1 Background and Motivation

Modern agriculture relies heavily on autonomous and semi-autonomous machinery for field operations such as fertilizing, spraying, and seeding. The efficiency of these machines depends both on their mechanical capacity and the quality of route planning. This includes how the machine traverses the field, manages refills, and minimizes redundant movement.

Agricultural vehicles have limited carrying capacity. This requires periodic returns to refill station (depot). Each return and subsequent repositioning contribute to *non-working distance*, which is the distance traveled without performing productive work, and, thereby, increasing fuel consumption. Optimizing machine movement to reduce this inefficiency is vital for sustainable and cost-effective field operations [8][9].

Apart from productivity, sustainability concerns such as soil compaction further motivate the optimization of routing. Repeated wheel passes over the same areas degrade soil structure, impede water infiltration, and, in turn, reduce yield potential [28]. Although the present work primarily focuses on travel efficiency, the broader aim aligns with the entire scope of sustainable machinery operation, by reducing unnecessary travel which also directly mitigates soil compaction and related environmental impacts.

## 1.2 Problem Statement

This research addresses the *Capacitated Path Planning Problem (CPP)* for agricultural fields characterized by parallel working tracks connected by headlands. Each track must be

visited exactly once in one direction by a machine with finite capacity. Directionality is also vital because servicing a track induces an entry and exit orientation. So traversing the same physical track in opposite directions yields different endpoint headings, which can change the feasibility and cost of subsequent headland transitions.

The problem is modeled as a directed graph $G = (V, A)$, where:

- $V$ represents the depot and the two endpoints of each track, and
- $A$ represents feasible arcs connecting these endpoints.

Each track $i$ has a demand $d_i$, and the vehicle has a capacity $Q$. When cumulative demand exceeds $Q$, the machine must return to the depot for refilling. The goal is to determine a sequence of refill-aware tours that:

1. begin and end at the depot,
2. visit each track exactly once,
3. obey capacity constraints, and
4. minimize total non-working distance, the unproductive travel between tracks and depot.

This problem is NP-hard, belonging to the same complexity class as the Capacitated Vehicle Routing Problem (CVRP) and Capacitated Arc Routing Problem (CARP) [42][45]. Therefore, exact optimization methods, such as Integer Linear Programming (ILP), can yield provably optimal solutions for small instances, while heuristic and metaheuristic algorithms are essential for larger fields.

## 1.3 Research Objectives

The overall goal of this thesis is to design and evaluate computational methods for efficient and refill-aware path planning in agricultural fields. The main objectives are:

This thesis pursues the following objectives and makes the corresponding contributions:

1. **ILP model development.** A directed, graph-based CPP formulation that models track directionality, depot refilling, and capacity constraints, together with an ILP that enforces feasibility and uses dynamic subtour elimination via lazy-constraint callbacks.

2. **Metaheuristic solver design.** A custom Simulated Annealing (SA) implementation in C++ featuring multiple neighborhood operators and a capacity-aware split procedure to generate feasible multi-tour solutions.

3. **Performance evaluation and trade-off analysis.** An empirical study comparing ILP and SA on benchmark/real field instances, reporting solution quality, runtime, and scalability trends across problem sizes.

4. **Benchmarking against an industrial baseline.** A comparative evaluation against an Ant Colony Optimization (ACO) method used in industry (Verge Ag), providing practical context for route quality and speed.

5. **Extension potential.** A roadmap for extending the framework toward additional objectives (e.g., turn minimization and soil-compaction-aware criteria) and richer operational constraints.

### 1.3.1 Innovation and Significance

The novelty of this work lies in the integration of exact and metaheuristic optimization methods specifically tailored to agricultural field routing. Unlike prior research works that treat vehicle routing and field coverage separately, this work unifies the two under a single capacitated framework.

The ILP leverages the problem structure of agricultural field logistics to dynamically manage subtour elimination which improves solver performance. On the other hand, the SA algorithm introduces multi-neighbourhood search and adaptive calibration to efficiently near-optimal solutions for large-scale fields.

This research was conducted under the *Mitacs Accelerate* program in collaboration with *Verge Ag*, which is a Canadian agri-tech company developing digital tools for operational planning. This partnership ensured that the proposed methods were evaluated on realistic datasets and problem setting directly relevant to precision agriculture.

Therefore, by benchmarking against an ACO-based industrial implementation, the study demonstrates that carefully structured academic optimization models can significantly enhance existing industry heuristics.

# Chapter 2

# Literature Review

## 2.1 Classical Vehicle Routing Problem (VRP) and CVRP

The Vehicle Routing Problem (VRP) is a fundamental problem in combinatorial optimization and operations research. Dantzig and Ramser first talked about it in 1959 as a way to improve fuel delivery routes [19]. The VRP is a more general version of the Traveling Salesman Problem (TSP) that works with more than one vehicle and serves customers in different locations. The goal is to lower the total cost of travel while making sure that each customer is only visited once and that every route starts and ends at the depot. Not long after, Clarke and Wright (1964) came up with the well-known ”savings heuristic,” which gave a computationally efficient way to build a solution [16]. Since then, the VRP has been used as a benchmark to test both exact and heuristic optimization methods. It also led to many different versions that could be used to model real-world problems like time windows, pickups and deliveries, random demands, and multiple depots.

Braekers, Ramaekers, and Van Nieuwenhuys (2016) present an extensive taxonomy of VRP variants and solution paradigms, categorizing more than 270 studies from 2009 to 2015 [12]. Their review shows how the VRP family is still evolving to include more realistic constraints, like dynamic, stochastic, and time-dependent routing. This is made possible by improvements in computation and modeling.

**Capacitated Vehicle Routing Problem (CVRP) -** The Capacitated Vehicle Routing Problem (CVRP) is one of the most important variants of VRP. In the CVRP, each vehicle can

only carry a certain amount of goods or materials, and deliveries of demands must be made without going over that limit. The CVRP is still an NP-hard problem, and it has been shown to be APX-hard in general metric spaces [14]. Here, APX-hardness indicates that it is unlikely to have PTAs in general metrics (unless $P = NP$), which helps explain why practical solvers rely heavily on exact methods for small to medium instances and heuristics/metaheuristics for larger ones. Because it is so hard to compute, there are two main ways to solve it: exact mathematical programming and approximation algorithms.

**Exact algorithms -** State-of-the-art exact methods use integer linear programming formulations that are solved with "branch-and-cut" and "branch-and-cut-and-price" algorithms. These algorithms add valid inequalities and create columns to efficiently represent routes. Bard, Kontoravdis, and Yu (2002) illustrated the efficacy of branch-and-cut for the Vehicle Routing Problem with Time Windows (VRPTW) [5]. More recent frameworks, like Pessoa et al. (2018), combine branch-and-cut with route pricing to find the best solution for medium-sized CVRP problems, filling in gaps that had been open for decades [37]. These approaches guarantee global optimality but suffer exponential growth in runtime with increasing instance size, limiting practicality for large-scale or real-time applications.

**Approximation and complexity -** Recent algorithmic developments concentrate on polynomial-time approximation schemes (PTAS) for Euclidean and geometric vehicle routing problems (VRPs). Starting with Das and Mathieu (2010) and continuing with Becker, and others, these works have created PTAS or quasi-PTAS methods that come very close to near- optimal when there are geometric constraints [20][6]. Chen (2023) [14] offers an extensive survey of these advancements, highlighting the theoretical distinction between general and geometric contexts. In general metric spaces, the Capacitated Vehicle Routing Problem (CVRP) is demonstrated to be inapproximable within any constant factor unless $P = NP$. In Euclidean spaces, where distance follows geometric rules, CVRP does allow PTAS algorithms that can get a solution within a factor of $(1+\varepsilon)$ for any fixed $\varepsilon > 0$. These results

clarify the boundaries of approximability and underscore the role of geometric structure in facilitating efficient approximation, offering crucial direction for both precise and heuristic approaches in capacitated routing.

**Capacitated Arc Routing Problem (CARP).** The Capacitated Arc Routing Problem (CARP) puts demand on edges (like road segments), which makes it better for services like street cleaning or waste collection. Golden et al. (1981) formalized the CARP and proved that it is NP-hard to even get close to it within a factor of 0.5 [25]. Exact algorithms for CARP, including the cutting-plane method by Belenguer and Benavent (2003), established the initial robust lower bounds for medium-sized instances [7]. Conversely, Wøhlk (2008) integrated ten years of research on both exact and heuristic CARP methodologies [45]. These traditional VRP and CARP formulations form the basis for contemporary routing in niche areas, such as agricultural field logistics, where customers and roads are replaced by track endpoints and headlands.

## 2.2 Field-Based Path Planning and Agricultural Applications

Agricultural field operations are a specific type of routing and coverage problem in which machines must travel all of the working tracks of a field while taking into account the field's layout, the ability to make turns, and the limited capacity. While coverage path planning (CPP) has been extensively examined in mobile robotics [35][15][24], the explicit formulation of in-field machine movement as a Vehicle Routing Problem (VRP) is a recent development. Bochtis and Sørensen were the first to apply this idea by coming up with the Vehicle Routing Problem in Field Logistics (VRPFL) [9][8]. They argued that almost every mechanized agricultural operation, including tillage, spraying, fertilization, or harvesting, can be perceived as a routing task across a network of spatially distributed tracks or swaths. In the past, field coverage was done in ad-hoc boustrophedon or back-and-forth patterns that were optimized for human operations. The VRPFL framework, on the other hand, lets

you systematically improve the order of the tracks and the logistics of refilling them using more flexible VRP methods.

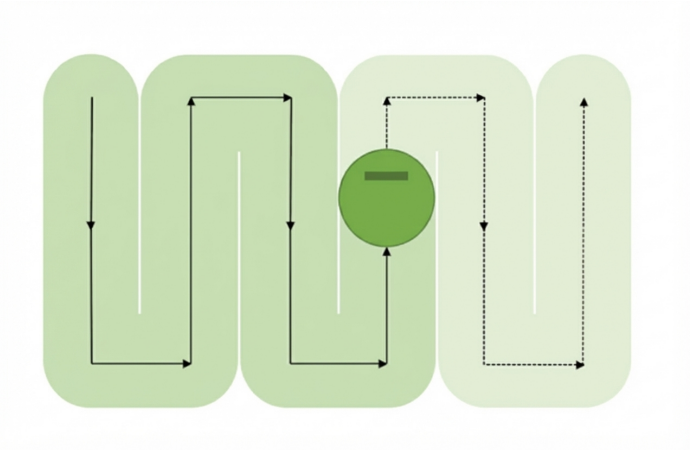


Figure 2.1: Typical boustrophedon (zigzag) coverage pattern used in regular field work adapted from Galceran and Carreras [24].

In Figure 2.1, the dark region show areas that have already been covered, and the lighter areas are ones to be covered. This traditional method makes it easier for operators, but it doesn't take into account refill capacity or headland optimization [24].

Bochtis and Sørensen [9][8] proposed the Vehicle Routing Problem in Field Logistics (VRPFL) framework, which turns the physical task of servicing an agricultural field into a graph-based routing problem. This abstraction shows the field layout as a directed graph $G = (V,A)$, where each working track is a "customer" node and the arcs are the possible headland connections between tracks. The "depot" is the field entrance or service trailer. Each track has a demand value, which is the amount of input material used or crop collected. The vehicle's limited capacity $Q$ limits how many tracks it can service before going back to the depot. The optimization looks for a set of tours that cover all the tracks exactly once and travel the least amount of non-operational distance.

**Benchmark developments -** Numerous studies have endeavored to establish reference cases for the assessment of capacitated coverage and routing algorithms in agricultural domains. Khosravani *et al.* [29] made a significant contribution by establishing a benchmark

arable field (2.97 ha, eight tracks) specifically designed for testing metaheuristic CPP algorithms. By thoroughly listing $1.3 \times 10^9$ possible sequences, the authors found the best way to reduce non-working distance in different capacity–distance situations. The benchmark, along with its distance and demand matrices, gives us an example of a small but realistic field instance with a known global optimum. This makes it a good reference for comparing algorithms. Their experiments demonstrated that both Simulated Annealing (SA) and Ant Colony Optimization (ACO) attained solutions within approximately 6% of the optimal, with SA displaying greater stability and accelerated convergence.

**Geometric and operational constraints -** Field geometry has a big effect on feasible route transitions. Vehicles are usually only allowed to turn in headland areas because turning in crop zones compacts the soil and harms standing plants. So, headland connectivity and turning radius limit the possible arcs between track endpoints [44][28]. Modern path planners show the field as a directed graph, with nodes representing the ends of tracks and arcs displaying the allowed transitions that take into account curvature limits and turning costs. This graph is then used to figure out non-working distances, which are then used to make the cost matrix for optimization. Recent research has incorporated Dubins-curve or curvature-bounded motion models to more accurately represent realistic maneuvering behavior for large agricultural machinery [28].

**Depot placement and multi-depot routing -** Vahdanjoo, Zhou, and Sørensen (2020) expanded the VRPFL to multi-depot contexts, necessitated by extensive farms that demand multiple refill or unloading stations [44]. For their study, they used a Simulated Annealing heuristic to plan routes for one to three depots. This was used in operations for spreading manure. The results showed that strategically placing extra depots cut down on total non-working travel by up to 20% compared to a baseline with only one depot. This shows that the location of the depot is an important design variable that affects both the length of the route and how the tracks are grouped together in each tour. These results are similar to

what we know about classical Multi-Depot VRP, but they are changed to fit field-geometric situations.

**Broader agricultural applications.** Later works stress the importance of optimizing routing and scheduling for the growth of sustainable and self-sufficient farming. Coverage planning for seeders, sprayers, and harvesters has a direct impact on fuel consumption and greenhouse gas (GHG) emissions. Numerous studies have employed metaheuristic and hybrid optimization techniques in this field. For instance, Ali and Seyyedhasani [1][39] devised capacity-constrained vehicle scheduling for crop harvesting, resulting in substantial decreases in fieldwork duration. Orfanou *et al.* [36] and Edwards *et al.* [22] expanded these concepts to multi-field operations and prototype route planners, respectively, demonstrating quantifiable reductions in non-productive travel.

## 2.3 Exact Methods for Field Routing (ILP, Branch-and-Cut, Lazy Constraints)

Exact optimization serves as the basis for demonstrating optimality in capacitated field-routing problems. In line with the tradition of Vehicle Routing Problem (VRP) research, these issues can be expressed as mixed-integer linear programs (MILPs) and resolved using branch-and-bound methodologies, including branch-and-cut or branch-and-cut-and-price techniques [5][37]. The same thing happens in agricultural field logistics where binary variables usually show whether a vehicle traverses through a track, while continuous variables may show flow or material balance. Constraints make sure that the depot starts and ends, that each track is covered exactly once, that the vehicle's capacity limits are met, and that the flow is conserved [11][44]. The resulting model is similar to the Capacitated VRP, but it has a cost matrix that is spatially structured. Because the working distance is fixed by the requirement to traverse all tracks, minimizing non-working distance targets the only part of travel that varies across feasible solutions.

**Subtour elimination and lazy constraints -** One of the biggest problems with these kinds of MILP formulations is preventing ”subtours”, which are disconnected cycles that do not include the depot. In traditional VRPs, subtour-elimination constraints (SECs) or connectivity cuts take care of this. It is too hard to list all SECs explicitly, so modern solvers add them dynamically within a branch-and-cut framework. Solvers like Gurobi and CPLEX use a ”lazy constraint” mechanism that lets these cuts be added whenever an infeasible cycle is found during the search. This method cuts down on the number of constraints in the root model by a huge amount without losing accuracy [37][7]. We use this strategy in our own ILP implementation: we solve a relaxed model iteratively, and cut any disconnected subtours we find in the current solution in a callback routine until we get global connectivity.

**Agricultural field formulations -** Bochtis and Vougioukas (2008) conceptualized headland field coverage as the traversal of a weighted graph, wherein each arc signifies a viable transition between track endpoints, and its weight reflects the corresponding non-working travel distance (turning or transfer) [11]. They simplified the sequencing problem to a shortest-tour search and set it up as a binary integer program (BIP) that makes sure each track is only visited once while keeping the total distance traveled for non-productive purposes to a minimum. Their findings indicated substantial reductions in dead travel compared to conventional boustrophedon or operator-defined coverage patterns.

Building on this line of work, Vahdanjoo *et al.* (2020) expanded the formulation to capacitated, refill-aware routing with multiple depots, incorporating explicit connectivity and subtour-elimination constraints (see their Eq. 9) and solving the model precisely using the Gurobi Optimizer [44].

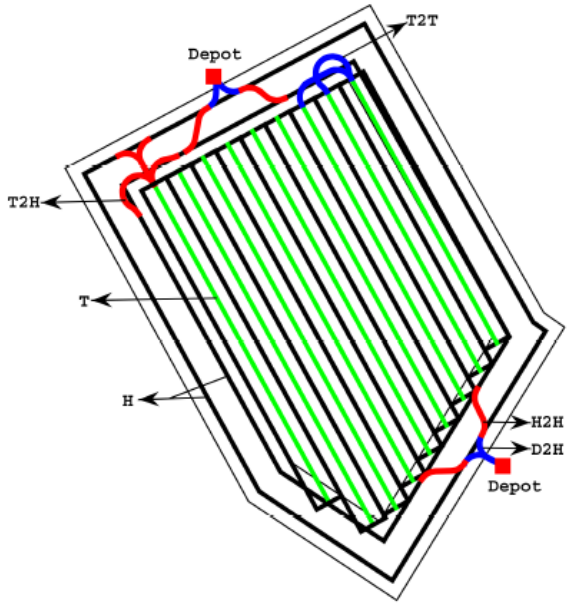



Figure 2.2: An illustrative field topology used to represent tracks, headlands, and depot connections from Vahdanjoo *et al.*, 2020 [44].

Figure 2.2 shows all the possible connecting paths in the geometric representation of a simple field where **T** represents a workable track, **D** is the depot and **H** is a feasible headland pass. A route that connects the ends of fieldwork tracks to the ends of nearby fieldwork tracks is denoted by **T2T**, a route that goes from each depot to the nearest headland pass is denoted by **D2H**, a route that goes from one headland pass to the next is given by **H2H**, and a route that connects the ends of fieldwork tracks leading to each headland pass is given by **T2H**.

Khosravani Moghadam *et al.* (2020) proposed a realistic 8-track field instance for benchmarking and validation. They then did exhaustive enumeration ($\approx 1.32 \times 10^9$ possible sequences) to find the best solutions for several capacitated coverage scenarios. Their dataset serves as a significant benchmark for evaluating metaheuristic algorithms, including Simulated Annealing (SA) and Ant Colony Optimization (ACO) [29] and have been part of the experimental instance during the initial development of this thesis.

**Modern exact algorithms -** In general VRP research, branch-and-cut algorithms have achieved steady improvements, notably for large-scale capacitated and time-windowed instances [5]. Branch-and-cut-and-price algorithms integrate column generation and cutting planes, reaching optimality for hundreds of customers in the CVRP [37]. In more complex cases, such as 3D loading or multi-layer constraints, recent works employ integrated decomposition or constraint programming hybrids [41, 31]. These advances underline how polyhedral insight and solver integration continue to extend the practical range of exact VRP solvers.

**Applicability to field routing -** For field logistics, precise MILP formulations seem to be only manageable for small to moderately sized fields. This is because the quantity of binary decision variables increases quadratically with the number of field endpoints (i.e., $|V|^2$), with each potential directed connection between nodes $(i, j)$ denoted by a variable $x_{ij}$. This quadratic increase in model size, along with an exponentially growing space of possible solutions, makes it impossible to find the best solution for big routing problems [32][18]. In the general vehicle routing literature, these limits on scalability are well known. Even small increases in the number of nodes can cause combinatorial explosions in the number of possible tours and the time it takes to compute them. Bochtis and Sørensen [8] observed that in agricultural field logistics, MILP-based formulations become computationally expensive beyond a few dozen tracks, which is why heuristic and metaheuristic approaches are used.

Nonetheless, precise optimization models continue to be essential as benchmarks. They give provable optimality certificates for small cases, lower bounds for bigger problems, and structural properties, like natural clustering of tracks or refill frequency, that help design heuristics and metaheuristics. This thesis utilizes Gurobi's branch-and-cut framework with lazy constraint callbacks to derive optimal or near-optimal solutions for benchmark fields, which serve as the ground truth for assessing the proposed Simulated Annealing (SA) methodology. These exact and approximate methods show a useful balance between guaranteed optimality and the ability to handle more data.

## 2.4 Metaheuristics for VRP and Field Routing (SA, ACO, GA)

Since the Capacitated Vehicle Routing Problem (CVRP) and its field variants are NP-hard, exact methods do not work for real-world problems of large scale. So, *metaheuristic* algorithms are the main tools used to improve routing today. They deliver high-quality, near-optimal solutions within acceptable computational time, especially in dynamic or large-scale scenarios [27][13]. Simulated Annealing (SA), Ant Colony Optimization (ACO), and Genetic Algorithms (GA) are some of the most popular metaheuristics for routing in transportation logistics and agriculture. They are ideal because they can balance exploration (global search) and exploitation (local refinement) without having to know about gradients or convexity.

**Simulated Annealing (SA) -** Kirkpatrick et al. (1983) [30] first talked about Simulated Annealing (SA). It is a stochastic search method based on trajectories and it uses a temperature schedule to probabilistically accept worse solutions, which lets it escape local minima. This is based on the annealing process in metallurgy. A random change is made to the current solution at each iteration. For example, two tracks might be swapped, a segment might be reversed, or a node might be moved. If the move lowers the total cost, it is accepted; otherwise, it may still be accepted with a chance of $P = \exp(-\Delta/T)$, where $\Delta$ is the cost increase and $T$ is the current temperature. Gradual cooling narrows the search to find the best minima.

In agricultural settings, SA has been utilized for capacitated field coverage and multi-depot routing, demonstrating significant stability and near-optimal outcomes for extensive fields [44]. In the 8-track benchmark field, SA found solutions that were about 3–6% away from the best non-working distance, and it took much less time than Ant Colony Optimization [29]. Recent evaluations validate SA's efficacy in combinatorial transportation optimization, especially when integrated with adaptive cooling or hybrid local search phases [13]. This thesis presents a tailored C++ implementation that utilizes various neigh-

borhood operators (swap, relocate, flip, and 2-opt) alongside adaptive cooling, resulting in consistently robust solutions with rapid convergence. Figure 2.3 illustrates the workflow of a basic SA algorithm.

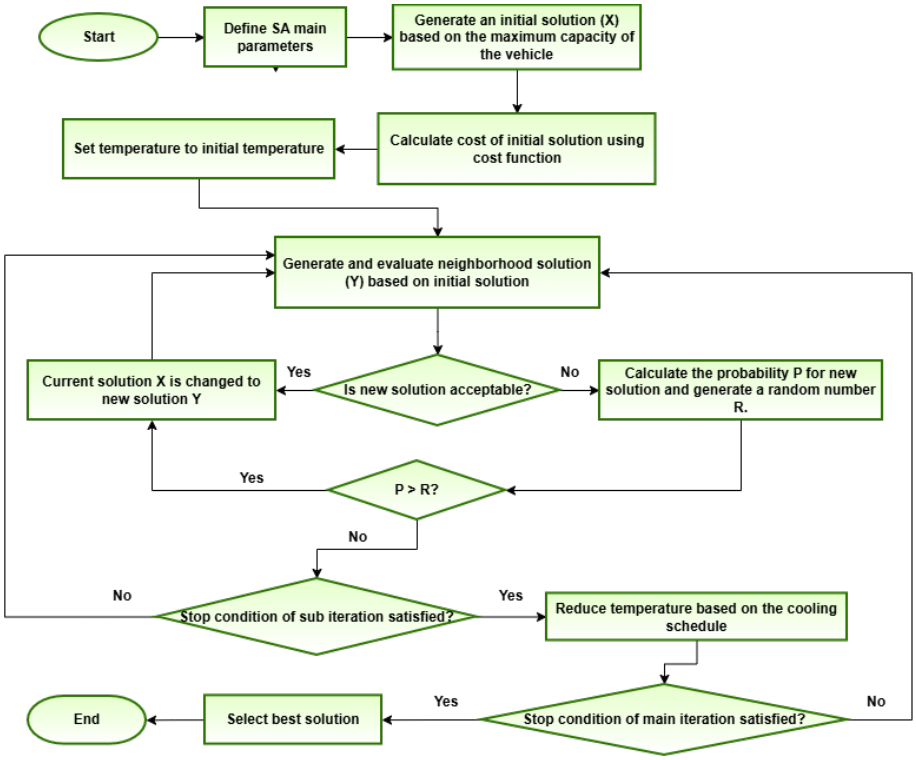



Figure 2.3: A flowchart illustrating how SA works (adapted from Vahdanjoo *et al.* [44]).

**Ant Colony Optimization (ACO) -** Dorigo and Stützle came up with ACO in the 1990s. It models how ants work together to foraging food by having agents build paths over and over again based on the strength of pheromones and how desirable they are (for example, by distance) [21]. After a solution is found, pheromone trails are changed to make good edges stronger, which changes how future searches are done. Variants like Ant Colony System (ACS) and Rank-Based Ant System (ASrank) make convergence better by using elitist reinforcement and local pheromone updates [27]. In the CVRP domain, ACO exhibits remarkable scalability and adaptability for extensive networks; however, it frequently necessitates parameter tuning to preserve solution diversity. Hameed *et al.* (2025) discovered that ACS-type algorithms surpass other ACO variants in both accuracy and runtime

for high-dimensional VRP instances, thereby validating ACO's effectiveness for practical routing challenges [27].

For field routing, ACO has been used to compare with SA on benchmark fields, where it made coverage patterns that were almost optimal but had higher variance between runs [29]. Even so, ACO is still appealing because it can explore in a distributed way and could work with real-time data or swarm-robotic systems. Modern adaptations speed up convergence in complex geometries by combining pheromone learning with reinforcement learning or hybrid local-search components.

**Genetic Algorithms (GA) -** Genetic Algorithms (GA) are evolutionary algorithms that work on a population of possible routes by using selection, crossover, and mutation to change them over time. Individuals (solutions) are usually shown as permutation of nodes, and genetic operators make new individuals by mixing up the traits of the parents. The groundbreaking genetic algorithm for the capacitated vehicle routing problem (CVRP) by Baker and Ayechew (2003) showed that it could find good solutions and work with larger problems by combining it with local search [4]. Recent surveys confirm GA's preeminence among population-based metaheuristics in transportation, highlighting their proficiency in balancing exploration and exploitation effectively [13]. For agricultural logistics, GAs have been adapted to multi-objective formulations (e.g., minimizing distance and soil compaction) and multi-depot routing problems, frequently in hybrid configurations that integrate GA with Tabu Search or Large Neighborhood Search.

**Hybrid and comparative trends -** Metaheuristics are often hybridized to integrate global exploration with local optimization. In the literature on multimodal and VRP, hybrid algorithms like GA+TS, SA+Local Search, and ACO+LNS always do better than standalone methods in terms of runtime and solution quality [13]. In agriculture, it's common to use combinations of constructive heuristics (like Clarke–Wright savings) with SA or GA to come up with workable initial solutions before improving them. New research trends in-

clude using GPU or cloud computing to run parallelized metaheuristics, adaptive parameter control, and hybrid reinforcement learning–metaheuristic frameworks for real-time optimization [13][27]. These methods fit with the need for scalable, data-driven path planning in precision agriculture.

## 2.5 Soil Compaction, Refilling, and Field Constraints in Literature

Repeated passes of machinery increase the bulk density of the soil and decrease the pore space, which makes it harder for roots to grow, water to get in, and crops to grow [3][43]. Controlled Traffic Farming (CTF) lessens these effects by keeping vehicles on permanent tramlines, which keeps crop zones from being disturbed while concentrating compaction. This puts limits on routing because vehicles have to stay in certain lanes and avoid traveling across fields when it's not necessary. Tamirat *et al.* (2022) discovered that while European farmers acknowledge the agronomic advantages of Controlled Traffic Farming (CTF), its adoption is constrained by financial considerations and equipment incompatibility [40]. Nørremark *et al.* (2022) showed that CTF optimization can cut down on non-working travel and traffic areas by as much as 25%, which lowers the risk of compaction [34]. Decision-support systems that schedule fieldwork based on soil moisture, load, and terrain can reduce compaction even when full CTF is not present. One example is the trafficability-based planning framework by Bochtis *et al.* (2012) [10]. In some cases, routing goals directly include the risk of compaction by using weighted combinations of travel distance and soil stress indicators. This links operational efficiency with sustainable soil management.

Another important part of field routing is refilling constraints. Because tanks and hoppers have limited capacity, machines have to go back to a depot or meet up with a mobile support unit every so often. Multi-depot configurations can significantly decrease non-productive travel in comparison to single-depot baselines [44]. Conversely, mobile depots, including tender trucks or grain carts, necessitate synchronization, which significantly complicates routing [2]. These logistical interactions make field routing a one-of-a-kind version

of the capacitated VRP.

Lastly, the shape of the field and the limits on how the machinery can turn also affect route optimization. Turning radii, steering-rate limits, and headland widths limit how track endpoints can be connected. In practice, uneven field shapes mean that smooth, curvature-continuous trajectories must be made that meet these physical limits [38]. Recent open-source tools like Fields2Cover [33] come with modules for making headlands, checking if a turning radius is possible, and predicting soil compaction using the built-in Soil2Cover model. These changes show how coverage path planning, vehicle routing, and soil-aware decision support are integrated together.

## 2.6 Positioning of This Work

Based on the above review, there are a few important research gaps at the crossroads of vehicle routing and field logistics:

1. **Integration of Coverage and Capacity Constraints -** Most coverage path planning (CPP) studies assume unlimited capacity or manage refills through ad hoc rules, while classical VRP/CVRP work seldom enforces explicit field-coverage and headland-only transitions. This thesis connects the two by creating a *capacitated coverage* model for track endpoints (CVRP/CARP-style) that requires one-time coverage, depot-based refilling, and possible headland connectivity [29][44][9][8].

2. **Limited Benchmarking of Exact vs. Heuristic Methods in Agricultural Layouts -** Aside from the 8-track arable benchmark, which has a known global optimum [29], not many studies directly compare exact MILP solvers with metaheuristics on the same field instances. In this study, we implement and assess both a Gurobi-based Integer Linear Programming (ILP) model and a Simulated Annealing (SA) heuristic on identical datasets to quantify the trade-offs between runtime and solution quality [5][37].

3. **Objective Clarity: Minimizing *Non-Working* Distance -** A lot of agricultural routing models try to cut down on total travel or time. Not as many try to separate the non-productive part (headland turns, empty moves) that best shows how efficient the operation is. Following [29][9], our formulation only minimizes non-working distance and assumes that on-track motion is required.

4. **Sustainability and Multi-Objective Criteria -** Soil compaction and geometric feasibility are frequently addressed qualitatively rather than through explicit optimization. New tools like Fields2Cover show that it is possible to reduce compaction with only small penalties for longer routes [33]. The proposed framework can easily be expanded to include multi-objective optimization (distance + compaction) and transitions that take into account curvature for sustainable field logistics.

5. **Standardized Datasets and Reproducibility -** Outside of the 8-track arable field [29], reproducible, capacity-annotated field instances with headland geometry are rare. We provide cleaned, capacity-aware instances (and logs) for both exact and heuristic baselining, in response to requests in VRP/CARP and field CPP for transparent, comparable benchmarks [45][7].

**Positioning -** This thesis is situated at the intersection of classical VRP/CVRP/CARP and precision agriculture. It provides:

- a field-specific capacitated coverage formulation with explicit non-working-distance minimization and headland feasibility;
- an exact ILP with lazy subtour elimination as an optimality baseline [5][37]; and
- a Simulated Annealing heuristic that reflects best practices in CVRP metaheuristics [30][13][27].By comparing both on common instances (including the arable benchmark [29]), we quantify the computational/quality trade-offs needed for deployable, refill-aware path planning in modern farm logistics [9][44].

Below Table 2.1 consists of all the routing solutions that have been explored in this chapter, formalised into a tabular format.

Table 2.1: Overview of routing solution methods (classical and applied to field logistics).

| **Method / Algorithm** | **Type** | **Key Characteristics** | **Typical Use Cases / Notes** |
|---|---|---|---|
| ILP + Branch-and-Cut | Exact | MILP with depot, coverage, and capacity constraints; cutting planes or lazy subtour elimination constraints (SECs) make sure that everything is connected | Best for small and medium-sized instances; standard in VRP/VRPTW [5]. For field routing, exact on small layouts but scales exponentially. |
| Branch-and-Cut-and-Price | Exact | Integrates column generation (route pricing) with branch-and-cut | State of the art for large CVRP instances [37]; conceptually transferable to field CPP but rarely applied due to implementation complexity. |

*Continued on next page*

*Table 2.1 (continued)*

| **Method / Algorithm** | **Type** | **Key Characteristics** | **Typical Use Cases / Notes** |
|---|---|---|---|
| Clarke–Wright Savings | Heuristic | Greedy route construction by repeatedly combining pairs based on how much distance they save | Classical constructive heuristic for VRP [16]; frequently employed as a starting point for metaheuristics in agricultural routing. |
| Simulated Annealing (SA) | Metaheuristic (single-solution) | Probabilistic local search with a cooling schedule look at neighborhood moves like swapping, relocating, reversing, and flipping | Exhibits strong anytime performance [30]; shown effective and stable for field CPP [29][44]. |
| Genetic Algorithm (GA) | Metaheuristic (population-based) | Evolutionary search with crossover, mutation, and local improvement | Works well for the classical CVRP [4]; can be used for routing problems with more than one goal or machine. |

*Continued on next page*

*Table 2.1 (continued)*

| **Method / Algorithm** | **Type** | **Key Characteristics** | **Typical Use Cases / Notes** |
|---|---|---|---|
| Ant Colony Optimization (ACO) | Metaheuristic (population-based) | Pheromone/heuristic-guided constructive search; ACS, ASrank variants | Competitive with poorly tuned parameters [21]; greater variance noted in field benchmarks compared to SA [29]. |
| Hybrid LNS/TS (examples) | Metaheuristic (hybrid) | Combines large neighborhood search (destroy/repair) with memory-based Tabu Search refinement | Gives good trade-offs between runtime and quality in transport VRPs; looks good for big, realistic field-routing problems [13]. |

# Chapter 3

# Problem Formulation

## 3.1 Field and Track Model

In capacitated agricultural operations, the field is modeled as a directed graph

$$G = (V, A)$$

where:

- $V$ is the set of nodes, representing the depot and the two endpoints of each field track
- $A \subseteq V \times V$ is the set of feasible directed arcs connecting these nodes.

Each track $i \in \{1, \ldots, n\}$ stands for a single working pass of the machine (e.g., spraying, fertilizing, harvesting). Its two physical endpoints are indexed as nodes $(2i-1)$ and $(2i)$. A *working arc* is the distance between two endpoints on the track. For each track we include both directions $(2i-1, 2i)$ and $(2i, 2i-1)$ in the model. However, only one will be chosen in the final solution so that every track is covered exactly once.

The depot node represents the out-of-field service location (refilling or unloading station). All tours start and end at the depot. Movement between track endpoints along headlands/boundaries is shown by *non-working arcs*. Accordingly, the arc set is divided into two parts:

$$A \; = \; A_t \cup A_0,$$

where $A_t$ are working (track) arcs and $A_0$ are non-working arcs (headland transitions, repo-

sitioning).

Each arc $(i, j) \in A$ is associated with a travel cost $c_{ij}$ (e.g., Euclidean or measured path length). Only non-working travel $A_0$ is penalized in the objective; distance along $A_t$ represents essential productive work.

Each machine has finite capacity $Q$ (e.g., tank volume or grain storage). Each track $i$ has a demand $d_i$ (material applied/harvested when that track is serviced). Each vehicle (or operation cycle) completes a *tour* $k$, which is a closed sequence of arcs that starts and ends at the depot node, visits a subset of tracks so that the total serviced demand does not exceed the capacity $Q$, and only goes through feasible headland (non-working) arcs between track endpoints. If $S_k \subseteq \{1, \ldots, n\}$ is the set of tracks serviced on tour $k$, then

$$\sum_{i \in S_k} d_i \leq Q.$$

Because the total field demand is usually more than $Q$, several depot-to-depot tours might be needed.

Consequently, the proposed formulation is structurally linked to the Capacitated Arc Routing Problem (CARP) [17], yet it incorporates several essential modifications tailored to agricultural field logistics.

First, the underlying graph separates *working* arcs (on-track traversal) from *non-working* or *headland* arcs. The objective function only penalizes the latter. This separation is based on the fact that productive passes help cover the field and do not make travel less efficient. Second, connectivity is limited to headland arcs, which show possible machine transitions between track endpoints. Third, each vertex does not represent an intersection in a road network but instead, it represents a physical track endpoint based on the field layout. This makes sure that routing decisions map directly to implement-level movements (i.e. how the machines enter, exit and transitions between tracks). Finally, the capacity constraint clearly shows the limits on material refills (like the capacity of a tank), with depot returns standing in for refilling operations.

These changes create a model called Capacitated Coverage Path Planning (CPP) that takes into account the geometry and operational limits of field logistics while still being closely related to the classical CARP formulation framework.

## 3.2 Mathematical Formulation (ILP)

We define the Capacitated Path Planning Problem (CPP) as an integer linear program that minimizes non-working travel distance while adhering to routing, coverage, and capacity constraints.

Let $K$ denote the set of tours. In this work, the number of depot-to-depot tours is not a free tuning parameter. A tour represents one refill cycle: it starts at the depot with full capacity $Q$, services a subset of tracks with total demand at most $Q$, and returns to the depot to refill. Given per-track demands $d_i$ and capacity $Q$, any feasible solution must use at least

$$K_{\min} = \left\lceil \frac{\sum_{i=1}^{n} d_i}{Q} \right\rceil$$

tours. Therefore, in the ILP implementation we set $|K| = K_{\min}$ (and, if an instance is infeasible due to arc feasibility/connectivity, we increment $|K|$ and re-solve). Thus, $|K|$ is determined by the instance demands and capacity rather than chosen arbitrarily.

For each directed arc $(i, j) \in A$ and tour $k \in K$, the binary decision variable is defined as:

$$x_{ij}^{k} = \begin{cases} 1, & \text{if tour } k \text{ traverses arc } (i, j), \\ 0, & \text{otherwise.} \end{cases}$$

**Objective Function: Minimization of Non-Working Distance -** The optimization problem aims to minimize travel cost over non-working arcs $A_0$, which represents movement without any productive output:

$$\min Z = \sum_{k \in K} \sum_{(i,j) \in A_0} c_{ij} x_{ij}^{k}, \tag{3.1}$$

where $c_{ij}$ denotes the cost (typically distance) of traversing arc $(i, j)$. Productive in-field arcs $(i, j) \in A_t$ are excluded from the cost summation as they are part of working motion.

**Model Constraints -**

**1. Depot Start and End (per tour).**

$$\sum_{j \in V \setminus \{0\}} x_{0j}^k = 1, \qquad \sum_{i \in V \setminus \{0\}} x_{i0}^k = 1, \quad \forall k \in K. \tag{3.2}$$

Each route must start and end at the depot. This makes sure that all tours have realistic refill cycles that start and end at the base station. In this formulation, the depot is set to index 0.

**2. Flow Conservation (non-depot vertices).**

$$\sum_{i:\{(i,v) \in A\}} x_{iv}^k = \sum_{j:\{(v,j) \in A\}} x_{vj}^k, \quad \forall v \in V \setminus \{0\}, \forall k \in K. \tag{3.3}$$

This ensures that the route stays the same: for each visited vertex, the number of incoming arcs is the equal to the number of outgoing arcs within the same tour.

**3. Track Coverage (exactly once).**

$$\sum_{k \in K} \left( x_{2i-1,2i}^k + x_{2i,2i-1}^k \right) = 1, \qquad \forall i = 1, \ldots, n. \tag{3.4}$$

Every field track must be serviced exactly once in one direction, ensuring full coverage without overlaps or gaps.

**4. Directional Exclusivity (one-way traversal).**

$$\sum_{k \in K} \left( x_{2i-1,2i}^k + x_{2i,2i-1}^k \right) \leq 1, \qquad \forall i = 1, \ldots, n, \; i \neq j. \tag{3.5}$$

This prevents simultaneous traversal of both directions of the same connection, which enforces that motion only goes in one direction along each track or headland path.

**5. Capacity Constraint (per tour).**

$$\sum_{i=1}^{n} d_i \left( x^k_{2i-1,2i} + x^k_{2i,2i-1} \right) \leq Q, \qquad \forall k \in K. \tag{3.6}$$

the total amount of material it needs to carry can't be more than its carrying capacity $Q$. This ensures periodic refilling at the depot when capacity is depleted.

**6. Subtour Elimination (connectivity constraint).**

$$\sum_{\substack{i,j \in S \\ i \neq j}} x^k_{ij} \leq |S| - 1, \qquad \forall S \subseteq V \setminus \{0\},\ |S| \geq 2,\ \forall k \in K. \tag{3.7}$$

These constraints prevent isolated cycles from forming that do not include the depot. This makes sure that all routes are connected and anchored to the depot. A subtour is a cycle over some $S \subseteq V \setminus \{0\}$ that is disconnected from the depot. If a cycle exists on $S$, it uses $|S|$ arcs inside $S$, but (3.7) allows at most $|S| - 1$, so the cycle is cut off. We add these constraints lazily: when an integer incumbent contains a subtour, we detect its node set $S$ and inject the violated inequality. Because listing all subsets $S$ is exponential, this constraint is not directly added to the model; instead, the solver's lazy constraint mechanism makes sure it is dynamically enforced.

**7. Binary Domain.**

$$x^k_{ij} \in \{0, 1\}, \qquad \forall (i, j) \in A,\ \forall k \in K. \tag{3.8}$$

All decisions are binary i.e. an arc is either used in a given tour or not.

**Complete Model.** Minimize (3.1) subject to constraints (3.2)–(3.8). This ILP makes sure that each track gets serviced only once during tours that start and end at the depot and are feasible within the limits of the capacity.

**Lazy Constraint Callback in Gurobi -** In the Gurobi Optimizer, the subtour-elimination constraints (3.7) are actually used with *lazy constraint callbacks*. Gurobi uses a branch-and-cut framework that includes LP relaxation, branching on integer variables, and the dynamic creation of cutting planes.

The solver starts the search by optimizing a relaxed version of the model that does not have any connectivity constraints. This gives a lower bound on the objective. As the search tree grows, candidate integer solutions (upper bounds) are looked at. The callback routine is called whenever a solution has more than one disconnected subtour. The callback builds the *support graph* consisting of the vertices and the arcs selected in the current incumbent solution (i.e., arcs with $x_{ij}^k = 1$), and then uses depth-first search to detect connected components. Any component $S \subseteq V \setminus \{0\}$ that is disconnected from the depot corresponds to a subtour, so the callback adds the violated cut for that set $S$. Then, it dynamically adds the following cut to make sure that the graph stays connected.

$$\sum_{i,j \in S, i \neq j} x_{ij}^k \leq |S| - 1$$

This integration of constraint generation into the branch-and-cut process keeps the model compact and accurate because only violated connectivity constraints are ever instantiated. The method is similar to how modern VRP implementations work (e.g., Pessoa *et al.*, 2018) and fits with Gurobi's internal branch-and-cut solver design [26]. So, moderately sized CPP instances can be solved to proven optimality with a lot fewer active constraints than in a static formulation.

## 3.3 Complexity Considerations

The Capacitated Path Planning Problem (CPP) is NP-hard, inheriting complexity from both the CVRP and the CARP [42][45]. The number of possible directed traversals increases by $O(2^n n!)$ for $n$ tracks, making it impossible to do an exhaustive search for any-

thing larger than small cases.

Even with advanced ILP solvers, the runtime grows quickly as $n$ increases. For small to medium-sized cases, it is easy to find the best solutions; but for bigger or multi-field cases, exact methods become too expensive to use. Heuristics and metaheuristics, including Simulated Annealing (SA), Ant Colony Optimization (ACO), and Genetic Algorithms (GA), yield high-quality solutions in polynomial time, sacrificing precision for scalability [21][4]. In this work, the ILP functions as an accurate benchmark for small fields, whereas a customized SA heuristic (Chapter 4) adapts the methodology to more extensive, practical situations.

# Chapter 4

# Solution Approaches

This chapter delineates the two complementary methodologies employed to address the Capacitated Path Planning Problem (CPP) as outlined in Chapter 3: an *exact* Integer Linear Programming (ILP) model resolved using Gurobi, and a *heuristic* Simulated Annealing (SA) metaheuristic. We also use an Ant Colony System (ACS) baseline to put performance and robustness in context; Chapter 5 will have more detailed quantitative comparisons.

## 4.1 Integer Linear Programming (ILP) with Gurobi

### Model Encoding and Constraint Structure

The CPP (Section 3.2) was encoded as an integer linear program and implemented in Python using the `Gurobi Optimizer 12.0.0`. Each binary decision variable $x_{ij}^k \in \{0,1\}$ indicates whether the machine travels directly from vertex $i$ to $j$ within tour $k$. The objective penalizes *only* non-working arcs $A_0$, while traversal of working-track arcs $A_t$ is treated as zero-cost productive motion.

Constraints mirror the formulation in Chapter 3:

- **Depot start and end** (Eq. (3.2)): every tour begins and terminates at the depot node.
- **Flow conservation** (Eq. (3.3)): in-flow equals out-flow at non-depot nodes.
- **Track coverage** (Eq. (3.4)): each track is serviced exactly once in one direction.
- **Direction exclusivity** (Eq. (3.5)): a connection is used in at most one direction.
- **Capacity** (Eq. (3.6)): cumulative demand per tour is limited to $Q$.

- **Subtour elimination** (Eq. (3.7)): enforced via a lazy-constraint callback.

During optimization, Gurobi calls a callback (see (Section 3.2)) that performs a depth-first search on the support graph of $x^k$-arcs to detect disconnected subtours that exclude the depot. When found, the corresponding inequality is added *on demand*. This branch-and-cut with lazy constraints keeps the model compact and accelerates convergence compared to enumerating all connectivity cuts a priori.

**Advantages and Limitations**

The Integer Linear Programming (ILP) formulation has many benefits that make it an important standard for research on field routing.

- It guarantees optimality i.e. for small and medium-sized fields, the solver can find routes that are provably optimal, resulting in a quantitative certificate of solution quality.
- The formulation makes sure that there is *transparency*. Each constraint is directly related to an operational element, such as covering all tracks, maintaining vehicle capacity, and connecting depot and field nodes. This makes the model easy for practitioners to understand and use.
- The model helps with rich *diagnostic insight*. Solver outputs such as node counts, cut families, and bound gaps provide information about instance hardness and algorithmic efficiency, which are useful for analyzing problem structure and guiding the design of scalable heuristics.

Even with these strengths, the ILP method has built-in limits when it comes to computation.

- The most difficult part is *scalability*. As the number of field nodes grows, the number of binary decision variables grows quadratically, which, in turn, makes the runtime grow exponentially.

- The need for memory increases a lot as dense distance matrices and the bookkeeping needed for lazy-constraint callbacks consume a lot of computing power.
- The method has a *practical ceiling* i.e. for fields with more than about 30–40 tracks, the model becomes too expensive to run even with multi-threaded optimization.

## 4.2 Simulated Annealing Heuristic

### Overview and Representation

Simulated Annealing (SA) provides a scalable, flexible alternative to the exact ILP, trading optimality guarantees for speed and robustness [30]. Our implementation is defined by: (i) a problem-aware *state (solution) representation*; (ii) a set of *neighborhood operators* that perturb the current state; (iii) a capacity-feasible *split/repair* procedure; and (iv) a calibrated *cooling schedule* with an *acceptance criterion* based on a temperature parameter.

A candidate solution is depicted as a *flattened sequence of track visits*, with each visit linked to a direction (forward or reverse). The word ”flattened” means that all of the track visits from all of the tours are stored in one long list instead of separate lists for each tour. This representation encodes both routing and sequencing decisions in a compact, order-based form.

During evaluation, the sequence is ”split” into separate tours that end at the depot based on the vehicle's capacity $Q$ using a split operator that is capacity-aware. This method makes it easier for the solution operators to generate neighborhoods, whether that means swapping places or moving to a new one, because they can work directly on the linear sequence without having to keep track of tour boundaries. The split procedure rebuilds valid tours and checks capacity and connectivity after each change to make sure that every candidate is still possible. So, the flattened sequence connects the ILP model's combinatorial structure to the metaheuristic's flexible, permutation-based search space.

The objective function $C$ evaluates a state by summing only *non-working* travel costs (headland/transition and depot-refill movements) between track endpoints; productive on-

track coverage is not penalized, consistent with the formulation in Chapter 3. Given a candidate move producing $s'$, the cost difference is $\Delta C = C(s') - C(s)$. SA then applies the standard *Metropolis acceptance rule*: improving moves are accepted, while worsening moves may be accepted with probability $\exp(-\Delta C/T)$, where $T$ is the current *temperature*. The temperature is decreased according to the chosen *cooling schedule* (e.g., geometric cooling $T \leftarrow \alpha T$), across a specified number of *main iterations* (temperature updates) and *sub-iterations* (neighbor evaluations at fixed $T$). The algorithm returns the *best-so-far* solution found under the selected stopping criterion.

**Initialization (Greedy Heuristic)**

The initializer constructs a single *flattened* sequence of track visits

$$R = \big[(i_1, \texttt{fwd}_1), \ldots, (i_m, \texttt{fwd}_m)\big],$$

where each pair $(i_t, \texttt{fwd}_t)$ states that track $i_t \in \{0, \ldots, n-1\}$ is serviced exactly once, and $\texttt{fwd}_t \in \{\texttt{true}, \texttt{false}\}$ is its traversal direction (forward or reverse). $R$ does not store any depot markers or tour boundaries. This is deliberate: SA's neighborhoods (swap, relocate, reverse, flip) are easier and more powerful on one linear sequence than on multiple lists. The split routine brings back tour boundaries later, which enforces capacity and depot returns. So, initialization aims for a high-quality ordering and orientation of tracks, and it puts off deciding where tours start and stop until later.

**Helper maps -** The node where the machine enters track $i$ in the direction of `fwd` is $\texttt{Enter}(i, \texttt{fwd})$, and the opposite endpoint reached after completing the working traversal is $\texttt{Exit}(i, \texttt{fwd})$. The depot is a unique node in the same index space (typically the first or the last node). These maps make costs explicit are and make it clear where headland travel starts and ends.

**Forbidden arcs -** The non-working distance matrix $C$ may contain transitions that are physically impossible, unsafe, or disallowed (e.g., blocked passages, excessive turning radius, islands within the fields, etc.). We encode such cases by a large sentinel value, $\tau$ and define a single feasibility predicate

$$\texttt{Feasible}(u,v) \equiv [C[u,v] < \tau].$$

Using one global threshold $\tau$ means finite distances are never mixed with "infinity", so acceptance probabilities are not distorted.

**Stored information -** For the initial solution, the algorithm keeps three pieces of state: (1) the current endpoint $u$ (the last headland node reached; initially the depot), (2) a boolean array $\texttt{used}[i]$ that prevents placing any track more than once, and (3) the remaining capacity $cap$ of the current *implicit* tour (initially $Q$). This mirrors the operational reality: we extend a tour while capacity permits and while the field geometry permits continuing; otherwise we close the tour and restart at the depot enforcing the heuristic typically used to solve the "bin-packing" problem.

**Candidate tracks -** At each step, we examine **unused tracks** and keep only those that:

1. **Fit** $d_i \leq cap$. This is because capacity violations deferred to the split stage would waste evaluations, so filtering early prevents doomed insertions.

2. **Are reachable from $u$.** There exists an orientation $\texttt{dir} \in \{\texttt{true}, \texttt{false}\}$ such that $\texttt{Feasible}(u, \texttt{Enter}(i, \texttt{dir}))$ so that dead-ends that can be entered from the current position are avoided.

3. **Can be closed by going back to the depot after service** with the same orientation, $\texttt{Feasible}(\texttt{Exit}(i, \texttt{dir}), \texttt{depot})$. This guarantees that once the track is appended, the

current partial tour can still return to the depot; otherwise, a geometrically valid step could strand the machine.

All three conditions are necessary to enforce resource realism, ensure forward progress, and preserve global recoverability.

**How the initializer works -** For a candidate $i$ that is feasible in both orientations, the initializer computes the two entry distances

$$c_i^{\text{fwd}} = C\big[u, \texttt{Enter}(i, \texttt{true})\big], \qquad c_i^{\text{rev}} = C\big[u, \texttt{Enter}(i, \texttt{false})\big]$$

and chooses the orientation with the smaller finite value. This "nearest feasible entrance" rule immediately reduces non-working motion without committing to a global tour structure. If only one orientation is feasible, the choice is forced; if neither is feasible, the track is not a candidate.

Among all candidates, the initializer selects the track with the *largest* demand $d_i$. The rationale is twofold: (i) packing capacity tightly tends to reduce the number of depot returns (hence fewer non-working legs), and (ii) it leaves more slack for SA to rearrange small-demand items later without triggering unnecessary additional tours. When two candidates have the same demand, ties are broken by smaller entry cost $\min\{c_i^{\text{fwd}}, c_i^{\text{rev}}\}$, then by smaller index $i$ for determinism and reproducibility.

After picking $(i, \texttt{dir}(i))$: we append it to $R$ (so its working traversal becomes fixed in the flattened sequence), mark $i$ as used (enforcing single coverage), decrease the remaining capacity by $d_i$, and update the current endpoint $u \leftarrow \texttt{Exit}(i, \texttt{dir}(i))$ (headland position after finishing the track). These updates keep the model's "physical cursor" aligned with the machine's location on the headland network.

If, at some step, no candidate satisfies *all* three criteria (fit, reachability, closability), the initializer closes the current implicit tour and restarts from the depot by resetting $u \leftarrow \texttt{depot}$ and $cap \leftarrow Q$. This restart is not a failure; it is a deliberate safeguard that avoids painting

the route into a corner on irregular fields. It also aligns the initializer with the evaluation-time split: both prefer opening a new tour over taking an infeasible or overly expensive continuation.

The initializer is *locally* greedy in headland distance: it minimizes *entry* cost at every insertion subject to capacity, reachability, and depot-closability. It *does not* attempt global optimization of tour boundaries or long-range sequencing across the entire field. Those global improvements are delegated to SA's neighborhoods (which reorder, reverse, and re-orient blocks) and to the split routine (which places tour cuts to respect $Q$ and penalizes only non-working motion).

**Consistency with the split/evaluation policy -** During evaluation, the split operator scans the flattened sequence left-to-right and opens a new tour when: (i) adding the next track would exceed $Q$, (ii) the transition into its entry endpoint is infeasible, or (iii) returning to the depot and restarting yields a cheaper continuation, modulated by a restart bias $\beta \in (0, 1]$. Because initialization already enforces "enterable now and closable later", the split rarely encounters pathological segments, and both stages apply the same feasibility logic via the threshold $\tau$. This consistency is crucial: changes that improve the initializer cannot degrade evaluation, and vice versa.

### Neighborhood Operators

The Simulated Annealing (SA) algorithm randomly chooses one neighborhood operator and uses it on the current flattened route $R$ at each iteration. Each operator changes the order and/or direction of one or more elements of $R = \left[(i_1, \texttt{fwd}_1), \ldots, (i_m, \texttt{fwd}_m)\right]$, which makes a new candidate route $R'$.

The split routine then checks all of the neighbors to see if they can make feasible tours and figure out the total cost. Candidates that make forbidden arcs or go over capacity are not accepted. The following operators offer additional methods for examining both local and global permutations within the flattened representation.

**Swap (intra-/inter-tour exchange) -** The **swap** operator exchanges the positions of two track visits in the flattened sequence, regardless of whether they belong to the same implicit tour or to different ones. It preserves the total number of visits and their orientations but alters their relative order, which can significantly change the non-working transition distances between adjacent entries. Intra-tour swaps focus on local fine-tuning (improving headland entry alignment within one tour), whereas inter-tour swaps can alter tour boundaries after the split, enabling large structural changes. This operator is inexpensive to compute and is often the most frequently selected neighborhood due to its balance between exploration and stability.

**Relocate (Or-opt move) -** The **relocate** operator removes a short block of one to three consecutive elements from the sequence and reinserts it at another position. This corresponds to cutting a short contiguous subsequence $(i_p, \ldots, i_q)$ and placing it between two other elements. Such a move can bring spatially related tracks closer together, repair poor local ordering, or merge short partial tours after splitting. It is particularly effective in eliminating redundant depot returns and reducing long non-working transitions between geometrically close tracks that were separated in the sequence. Because relocation changes both local adjacency and load distribution, it often yields larger cost improvements than simple swaps.

**Reverse (2-opt-like move) -** The **reverse** operator inverts the order of a contiguous subsequence of the flattened route, analogous to the 2-opt move in classical TSP heuristics. By reversing $(i_p, i_{p+1}, \ldots, i_q)$ to $(i_q, \ldots, i_{p+1}, i_p)$, it simultaneously changes the sequencing of visits and the direction of headland connections between them. When combined with the split operator, reversals can eliminate inefficient "zigzag" connections and produce smoother field traversals. This operator explores a larger neighborhood than swaps, allowing the search to escape shallow local minima caused by short loops or directionally inconsistent arcs.

**Segment swap (cross-exchange) -** The **segment swap** (or *cross-exchange*) operator selects two equal-length contiguous segments and exchanges them. It generalizes the simple swap by acting on blocks rather than single elements, thus producing more disruptive structural changes while maintaining sequence length. In the flattened setting, this move effectively exchanges portions of two implicit tours. It is useful for rebalancing tour workloads and uncovering alternate depot partitions that the greedy initialization or local swaps cannot reach. Although computationally heavier, it provides global diversification essential to SA's ability to escape deep basins in the search space.

**Flip (direction toggle) -** The **flip** operator changes the traversal direction of a selected visit, switching its orientation flag $\texttt{fwd}_i \leftarrow \neg\texttt{fwd}_i$. It preserves the sequence order but replaces the entry and exit endpoints of that track in subsequent evaluations. This operator is critical for problems where the geometric entry costs differ substantially between directions (for instance, when one headland side connects more efficiently to the depot or neighboring tracks). By allowing direction changes without reordering, the flip move complements the other operators and expands the search space to include both sequencing and orientation dimensions.

Together, these operators balance *exploitation* and *exploration*:

- **Swap** and **Reverse** exploit local structure by optimizing adjacency relations.
- **Relocate** and **Segment swap** provide medium- and large-scale exploration by moving or exchanging blocks.
- **Flip** allows orientation flexibility, correcting infeasible or suboptimal entry directions.

Because all are applied on the flattened representation, they operate independently of explicit tour boundaries, allowing the subsequent split operator to reconstruct feasible capacity-limited tours dynamically. This design keeps the search simple, consistent, and capable of exploring a wide range of feasible route configurations.

### Annealing Schedule and Acceptance

Let $C$ denote the total non-working distance of the current solution and $C'$ that of a neighbor. The cost change is $\Delta = C' - C$, and the move is accepted with probability

$$P(\text{accept}) = \begin{cases} 1, & \Delta \leq 0, \\ \exp(-\Delta/T), & \Delta > 0, \end{cases}$$

where $T$ is the current *temperature*. Thus, improving moves are always accepted, while worsening ones are accepted with exponentially decreasing probability. This mechanism enables the algorithm to escape local minima early and gradually focus on intensification as $T$ cools [30].

**Initial temperature.** Before the main loop, a short neighbor-sampling phase estimates the initial temperature, $T_0$. A small set of random neighbors is generated from the greedy seed, and the average positive cost increase $\overline{\Delta}$ is measured. We then choose $T_0$ so that such uphill moves would be accepted with probability 0.8:

$$T_0 = -\frac{\overline{\Delta}}{\ln(0.8)}.$$

This data-driven initialization scales the schedule automatically to the instance's cost range and avoids manual tuning.

**Cooling and inner-loop length.** Temperature decreases geometrically as

$$T_{t+1} = \alpha T_t, \qquad \alpha = 0.999,$$

a slow decay that balances exploration and convergence. At each level, $L = \max(100, 40n)$ neighbors are evaluated, ensuring sufficient sampling for both small and large fields, where $n$ is the number of tracks.

**Stopping conditions.** Annealing stops when either (i) the temperature drops below $T_{\min} = \max(10^{-12}, 10^{-4}T_0)$, or (ii) the best cost has not improved over a fixed stagnation window. These criteria prevent unnecessary computation once convergence is evident.

**Interpretation.** The temperature schedule acts as a control over search randomness: high $T$ values favor diversification by occasionally accepting worse moves, while decreasing $T$ gradually enforces greedier behavior. For the CPP, this behavior is especially beneficial since feasible high-quality routes often lie across narrow ridges of infeasibility; probabilistic acceptance allows the search to cross these barriers and reach better depot partitions and orientations later. The geometric schedule thus provides a predictable, smooth transition from exploration to exploitation and performs robustly across all tested field instances.

### Route Splitting and Evaluation

For a candidate permutation $R$, the split routine reconstructs explicit depot-bounded tours by scanning the flattened sequence from left to right. Each track in $R$ is appended to the current tour until one of the following conditions triggers a restart at the depot:

(a) **Capacity limit:** This is when adding the next track would cause the accumulated demand to exceed the vehicle capacity $Q$;

(b) **Feasibility failure:** This is when the transition to the entry node of the next track is forbidden or assigned a very large (infeasible) headland cost, i.e., $C[p, \texttt{Enter}(i)] \geq \tau$;

(c) **Restart preference:** This is when returning to the depot and restarting a new tour is sufficiently cheaper than directly continuing to the next track, according to a bias parameter $\beta \in (0, 1]$.

The third rule introduces controlled flexibility in tour partitioning. Formally, when considering a continuation from the current endpoint $p$ to the next entry node $a$, the split

routine compares two options:

$$c_{\text{cont}} = C[p, a], \qquad c_{\text{re}} = C[p, \texttt{depot}] + C[\texttt{depot}, a].$$

If either $c_{\text{cont}}$ is infeasible ($\geq \tau$) or $\beta c_{\text{re}} < c_{\text{cont}}$, a new tour is started at the depot before visiting $a$. The bias parameter $\beta$ thus controls the algorithm's tendency to restart early:

- When $\beta = 1$, the split behaves greedily where restarts occur only if returning to the depot is *strictly* cheaper than continuing.
- When $\beta < 1$, the routine becomes more restart-prone, slightly favoring depot returns that may lead to shorter overall headland distances after reordering.

In this thesis, $\beta$ is treated as a lightweight heuristic control that trades off early restarts (more depot returns) against continuing a partially built tour. We set $\beta = 0.9$ as a conservative alue close to the greedy case ($\beta \approx 1$) to avoid excessive fragmentation, while still allowing occasional early restarts when they are near-competitive in cost. In preliminary runs, values substantially below 0.9 tended to produce many short tours, whereas $\beta$ close to 1 reduced restarts but could miss beneficial depot returns. A systematic sensitivity analysis of $\beta$ is, thus, a natural extension and is left for future work.

Empirically, $\beta = 0.9$ provides a good balance between continuity and exploration, avoiding excessively fragmented tours while allowing occasional beneficial restarts.

The cost of a single tour is computed as

$$C(\text{tour}) = C[\texttt{depot}, \texttt{Enter}(v_1)] + \sum_{t=1}^{m-1} C[\texttt{Exit}(v_t), \texttt{Enter}(v_{t+1})] + C[\texttt{Exit}(v_m), \texttt{depot}],$$

where the sum includes only non-working (headland) transitions; working-track movements incur no penalty. The total solution cost is the sum of all tour costs after splitting.

This "giant-tour + split" evaluation pattern decouples sequencing from capacity enforcement: the Simulated Annealing neighborhoods operate on the flattened sequence $R$,

while the split function ensures that each evaluated neighbor respects capacity, depot connectivity, and headland feasibility. This design allows SA to explore large permutation spaces efficiently while maintaining consistent operational feasibility during cost evaluation.

### Simulated Annealing Algorithm

The pseudocode below illustrates the algorithm of the workflow.

Algorithm 1: Simulated Annealing for Capacitated Path Planning (SA–CPP)

**Input:** distance matrix $C \in \mathbb{R}_{\geq 0}^{(2n+1)\times(2n+1)}$; per-track demand $d[0..n-1]$; capacity $Q$; depot index `depot`; large penalty threshold $\tau$.

**Output:** best flattened route $R^{\star}$, tours, and total non-working cost $Z^{\star}$.

Define endpoint indices: $L(i) \leftarrow 2i-1$, $R(i) \leftarrow 2i$.

Define: $\texttt{Enter}(i,\texttt{true}) = L(i)$, $\texttt{Exit}(i,\texttt{true}) = R(i)$, $\texttt{Enter}(i,\texttt{false}) = R(i)$, $\texttt{Exit}(i,\texttt{false}) = L(i)$.

Feasibility: $\texttt{Feasible}(u,v) \iff C[u,v] < \tau$.

**procedure** BUILDINITIALROUTE

    $R \leftarrow [\,]$, $\texttt{used}[0..n-1] \leftarrow \texttt{false}$, $u \leftarrow \texttt{depot}$, $cap \leftarrow Q$

    **while** there exists $i$ with $\neg\texttt{used}[i]$ **do**

        $\mathcal{C} \leftarrow \{\, i : \neg\texttt{used}[i],\ d[i] \leq cap,\ \exists \texttt{dir} \in \{\texttt{true},\texttt{false}\} : \texttt{Feasible}(u,\texttt{Enter}(i,\texttt{dir}))$ and $\texttt{Feasible}(\texttt{Exit}(i,\texttt{dir}),\texttt{depot}) \,\}$

        **if** $\mathcal{C} = \emptyset$ **then**

            $u \leftarrow \texttt{depot}$; $cap \leftarrow Q$; **continue**

        **for** each $i \in \mathcal{C}$ **do**

            $c_{\text{fwd}} \leftarrow C[u,\texttt{Enter}(i,\texttt{true})]$

            $c_{\text{rev}} \leftarrow C[u,\texttt{Enter}(i,\texttt{false})]$

            $\texttt{dir}(i) \leftarrow \arg\min\{c_{\text{fwd}}, c_{\text{rev}}\}$ // ignore infeasible values $\geq \tau$

        $i^{\star} \leftarrow \arg\max_{i\in\mathcal{C}} \big(d[i], -\min\{c_{\text{fwd}}, c_{\text{rev}}\}, -i\big)$

        Append $(i^{\star}, \texttt{dir}(i^{\star}))$ to $R$

```
18:          used[i*] ← true; cap ← cap − d[i*]
19:          u ← Exit(i*, dir(i*))
20:      return R
21:  procedure SplitIntoTours(R, β)
22:      Tours ← [], T ← [], load ← 0, p ← depot
23:      for each (i, fwd) in R do
24:          a ← Enter(i, fwd), b ← Exit(i, fwd)
25:          start ← (load + d[i] > Q)
26:          if not start then
27:              c_cont ← Feasible(p, a)?C[p, a] : ∞
28:              c_re ← (Feasible(p, depot) ∧ Feasible(depot, a))?C[p, depot]+C[depot, a] : ∞
29:              start ← (c_cont = ∞) or (β c_re < c_cont)
30:          if start and T ≠ [] then
31:              Append T to Tours; T ← []; load ← 0; p ← depot
32:          Append (i, fwd) to T; load ← load + d[i]; p ← b
33:      if T ≠ [] then
34:          Append T to Tours
35:      return Tours
36:  procedure TourCost(T)
37:      u ← depot; z ← 0
38:      for each (i, fwd) in T do
39:          a ← Enter(i, fwd); b ← Exit(i, fwd)
40:          if not Feasible(u, a) then
41:              return ∞
42:          z ← z + C[u, a]; u ← b
43:      if not Feasible(u, depot) then
44:          return ∞
45:      return z + C[u, depot]
```

**procedure** TOTALCOST($R$)

    $S \leftarrow$ SPLITINTOTOURS($R, 0.9$)

    **return** $\sum_{T \in S}$ TOURCOST($T$)

**procedure** NEIGHBOR($R$)

    *Return a random neighbor by one of:*

        (1) swap two items; (2) relocate a short block; (3) reverse a short segment;

        (4) swap equal-length segments; (5) flip one item's direction; (6) flip a short block's directions.

**procedure** SIMULATEDANNEALING($C, d, Q,$ `depot`$, \tau$)

    $R \leftarrow$ BUILDINITIALROUTE; $Z \leftarrow$ TOTALCOST($R$)

    $(R^\star, Z^\star) \leftarrow (R, Z)$

    Estimate $T_0$ by sampling neighbors (target acceptance 0.8)

    $T_{\min} \leftarrow \max(10^{-12}, 10^{-4} T_0)$; $\alpha \leftarrow 0.999$

    **while** $T_0 > T_{\min}$ **do**

        **for** $\ell \leftarrow 1$ **to** $\max(100, 40n)$ **do**

            $R' \leftarrow$ NEIGHBOR($R$); $Z' \leftarrow$ TOTALCOST($R'$)

            **if** $Z' = \infty$ **then**

                **continue**

            $\Delta \leftarrow Z' - Z$; draw $u \sim \mathcal{U}[0, 1]$

            **if** $\Delta < 0$ **or** $e^{-\Delta / T_0} > u$ **then**

                $(R, Z) \leftarrow (R', Z')$

                **if** $Z < Z^\star$ **then**

                    $(R^\star, Z^\star) \leftarrow (R, Z)$

        $T_0 \leftarrow \alpha T_0$

    **return** $R^\star$

### Illustrative Instance: 8-Track Field Example

To better illustrate the workflow, we reference the canonical 8-track benchmark introduced by Khosravani Moghadam *et al.* (2020) [29], also employed in our early validation exper-

iments. The instance consists of one on-field depot and $n = 8$ parallel tracks, producing a $(2n+1)\times(2n+1)$ symmetric distance matrix where large or missing entries represent infeasible headland transitions. Each track $i$ has a specific demand $d_i$, and a vehicle capacity of $Q = 46{,}000$ L is enforced. Both the ILP and SA solvers consume the same data schema: a distance matrix, a per-track demand vector, and a single-depot identifier.

In the ILP formulation, the solver enumerates all feasible binary arcs $(i, j)$ between track endpoints and imposes depot, flow-balance, coverage, and capacity constraints (Eqs. 3.1–3.7). Subtour elimination is handled dynamically by Gurobi through a lazy-constraint callback, which detects disconnected subtours during branch-and-cut and injects valid inequalities on demand [5][37].

In contrast, the SA heuristic operates on a flattened permutation of track visits with associated traversal directions. Its capacity-aware *split* operator partitions this sequence into depot-bounded tours, automatically ensuring feasibility under $Q$. Each neighborhood move (`swap`, `relocate`, `reverse`, `flip`) is evaluated through this split routine; only non-working (headland) arcs contribute to the cost, consistent with the objective defined in Chapter 3 and the SA design in Section 4.2 [30][29].


Figure 4.1: Layout of the 8-track benchmark field with on-field depot and working tracks.

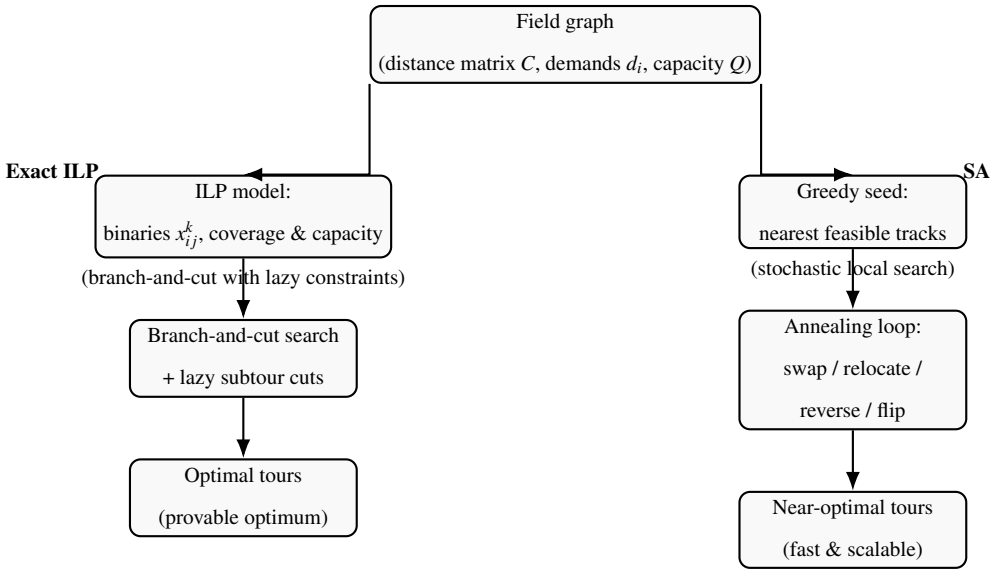



Figure 4.2: Comparison of ILP and SA solver workflows.

Figure 4.1 presents the physical configuration of the benchmark field, showing the on-field depot and working tracks used to generate the distance matrix. Figure 4.2 contrasts the computational approaches: the ILP follows an exact branch-and-cut process with lazy constraint generation, while the SA heuristic performs iterative neighborhood exploration under an annealing schedule. Together, these figures illustrate how the two methods operate on identical data to produce capacity-feasible, depot-bounded tours that minimize non-working distance. Quantitative performance on larger, irregular fields (Fields A–C) is discussed in Chapter 5, where SA continues to match or outperform time-limited ILP in solution quality for medium and large fields.

We include here the tour outputs from both ILP (Gurobi) and Simulated Annealing (SA). Arrows labeled with $\rightarrow$ represent transitions: • track arcs are labeled with demand in liters, • non-working arcs are labeled with travel distance in meters.

**Gurobi Best Cost:** 7085.49

Tour 1: Depot(0) $\overset{241.77m}{\rightarrow}$ 2 $\overset{11237L}{\rightarrow}$ 1 $\overset{45.57m}{\rightarrow}$ 5 $\overset{16334L}{\rightarrow}$ 6 $\overset{59.93m}{\rightarrow}$ 12 $\overset{18384L}{\rightarrow}$ 11 $\overset{83.57m}{\rightarrow}$ Depot(0)

Tour 2: Depot(0) $\overset{260.22m}{\rightarrow}$ 4 $\overset{16950L}{\rightarrow}$ 3 $\overset{45.57m}{\rightarrow}$ 7 $\overset{15670L}{\rightarrow}$ 8 $\overset{304.80m}{\rightarrow}$ 10 $\overset{14590L}{\rightarrow}$ 9 $\overset{63.41m}{\rightarrow}$ Depot(0)

Tour 3: Depot(0) $\overset{126.05m}{\rightarrow}$ 15 $\overset{19861L}{\rightarrow}$ 16 $\overset{22.11m}{\rightarrow}$ 14 $\overset{19154L}{\rightarrow}$ 13 $\overset{107.48m}{\rightarrow}$ Depot(0)

*Presolved: 193 rows, 816 columns, 2496 nonzeros. Explored 22,935 nodes (555,401 simplex iterations) in 2.85 seconds (5.54 work units). Thread count was 14 (of 14 available processors).*

**SA Best Cost:** 7085.49

Tour 1: Depot(0) $\overset{83.57m}{\rightarrow}$ 11 $\overset{18384L}{\rightarrow}$ 12 $\overset{59.93m}{\rightarrow}$ 6 $\overset{16334L}{\rightarrow}$ 5 $\overset{45.57m}{\rightarrow}$ 1 $\overset{11237L}{\rightarrow}$ 2 $\overset{241.77m}{\rightarrow}$ Depot(0)

Tour 2: Depot(0) $\overset{260.22m}{\rightarrow}$ 4 $\overset{16950L}{\rightarrow}$ 3 $\overset{45.57m}{\rightarrow}$ 7 $\overset{15670L}{\rightarrow}$ 8 $\overset{304.80m}{\rightarrow}$ 10 $\overset{14590L}{\rightarrow}$ 9 $\overset{63.41m}{\rightarrow}$ Depot(0)

Tour 3: Depot(0) $\overset{126.05m}{\rightarrow}$ 15 $\overset{19861L}{\rightarrow}$ 16 $\overset{22.11m}{\rightarrow}$ 14 $\overset{19154L}{\rightarrow}$ 13 $\overset{107.48m}{\rightarrow}$ Depot(0)

Figure 4.3: Tour outputs from Gurobi and Simulated Annealing, showing only the output of one run of SA. Distances (m) shown on non-working arcs; demands (L) shown on track arcs.

On this instance, Gurobi returns a *verified optimal* three-tour solution with objective value 7085.49 (Figure 4.3). Over five SA runs, the heuristic matches the optimum twice and otherwise returns near-optimal solutions (e.g., 7089.48, 7109.31, 7123.77), illustrating that SA can closely track the exact benchmark on small instances while scaling to larger fields (Chapter 5).

### Strengths and Limitations

**Strengths.**

- SA consistently yields high-quality, near-optimal routes within seconds, making it well suited for real-time or large-field routing where exact methods become intractable.
- Its modular structure—combining multiple neighborhood operators, a tunable annealing schedule, and a flattened route representation—allows straightforward exten-

sions to multi-objective or parallel variants.

- The method requires no commercial solver or specialized libraries, enabling easy integration into on-field decision-support tools.

**Limitations.**

- As a metaheuristic, SA provides high-quality but not provably optimal solutions, with performance depending on parameter calibration and neighborhood selection.
- Due to probabilistic acceptance, minor run-to-run differences in route quality may occur, though calibrated settings mitigate this effect in practice.

# Chapter 5

# Experimental Setup and Results

This chapter presents the computational setup, test instances, and comparative results of the Integer Linear Programming (ILP), Simulated Annealing (SA), and Ant Colony Optimization (ACO) approaches introduced in Chapter 4. The analyses focus on solution quality, computational efficiency, and scalability across real agricultural fields of increasing complexity.

## 5.1 Test Instances

Three real-world field datasets, anonymized as **Field A**, **Field B**, and **Field C**, were used for evaluation. Table 5.1 displays the summary of the metrics of said fields while Figure 5.1 shows the geometric layout of the fields.

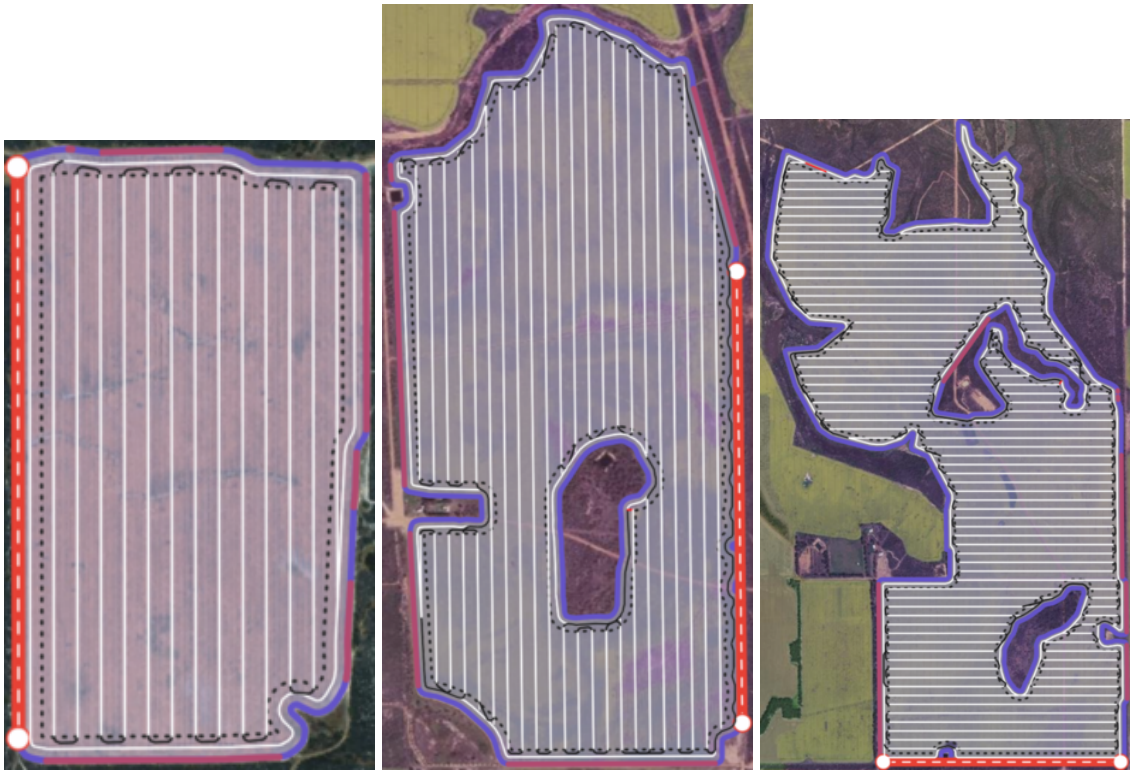
Each field was discretized into working tracks and headland transition arcs. Track demands were computed from swath length and the prescribed application rate, while headland arcs model non-productive travel. All depots were located on-field, consistent with typical machinery operations.

The inter-endpoint distance matrix $C[i, j]$ was provided by Verge Ag as part of the field data package. The company's own routing engine figures out the shortest physically possible maneuver between track endpoints in the headland network for each entry. Transitions that were not possible or were not connected were given a large penalty cost ($C[i, j] \geq \tau$) to keep them from being chosen during optimization. This made a symmetric matrix that shows how the field geometry and turning feasibility work in different directions. It is the same

input for both the ILP and SA solvers.

Table 5.1: Summary of real-field instances used for evaluation.

| Field | Perimeter (km) | Area (ha) | Tracks | Depot | Geometry Notes |
|---|---|---|---|---|---|
| A | 3.70 | 78.6 | 13 | On-field | Moderately irregular boundary |
| B | 7.26 | 256.6 | 35 | On-field | Irregular boundary, one internal island |
| C | 12.64 | 409.4 | 96 | On-field | Complex boundary with multiple islands |



(a) Field A (b) Field B (c) Field C

Figure 5.1: Geometric layouts of the three test fields with discretized tracks (white) and headland boundaries (red/purple). © Verge Ag

**Equipment and operational parameters.** Each instance assumes a 48 m working swath, 20 km/h working speed, 15 km/h turning speed, 18 m minimum turn radius, an 8,000 L tank capacity, and an application rate of 125 L/ha. These parameters approximate those of a self-propelled sprayer or liquid fertilizer applicator operating under Controlled Traffic Farming

(CTF) conditions.

## 5.2 Methods and Implementations

All algorithms share identical input data: a distance matrix and per-track demand list. The objective for all solvers is the minimization of *non-working distance*, i.e., travel along headland and transfer arcs that do not contribute to field coverage.

**ILP (Python / Gurobi 12.0.0).** The exact method implements the capacitated coverage ILP described in Section 4.1. Subtour elimination is enforced dynamically via Gurobi's lazy-constraint callback. Default solver parameters were used unless stated otherwise, with time limits applied for large instances.

**Simulated Annealing (C++).** The metaheuristic implementation follows the SA–CPP design of Chapter 4.2, using five independent runs per field. Neighborhood operators include `swap`, `2-opt/reverse`, `relocate`, orientation `flip`, and `segment-swap` operators. The temperature schedule uses a geometric cooling rate of 0.999 and a restart bias of 0.9 in the split procedure.

**Ant Colony Optimization (C#).** An Ant Colony Optimization (ACO) implementation, provided by Verge Ag, was used for comparative benchmarking. Since the solver operates within the company's proprietary optimization framework and is implemented in C#, the internal parameterization and update rules are confidential. However, the method is based on the general ACO paradigm that Dorigo [21] introduced, in which a group of artificial ants work together to build routes over the field graph. Each ant builds a solution bit by bit by choosing arcs at random based on pheromone intensity and heuristic desirability. Global pheromone updates strengthen high-quality routes over time. The algorithm focuses on quick convergence and the ability to reproduce results with fixed random seeds. The same distance matrices and demand data were used in the ILP and SA experiments to keep things

consistent.

## 5.3 Evaluation Metrics

Algorithmic performance is compared using four metrics:

- **Objective value:** total non-working distance ($Z$).
- **Number of tours:** number of refill trips or capacity-bounded tours.
- **Runtime:** total computational time in seconds.
- **Optimality gap (ILP only):** The relative optimality gap quantifies how close the best integer solution ($Z_{\text{inc}}$) is to the current best bound ($Z_{\text{bd}}$) reported by the solver:

$$\text{Gap} = 100 \times \frac{Z_{\text{inc}} - Z_{\text{bd}}}{|Z_{\text{inc}}|} \,\%.$$

  Here, $Z_{\text{inc}}$ denotes the objective value of the best feasible (incumbent) solution found, while $Z_{\text{bd}}$ represents the best known lower bound on the optimal objective at termination. A smaller gap therefore indicates tighter convergence toward the global optimum.

For the stochastic Simulated Annealing (SA) heuristic, each instance was executed five independent times using identical parameters to assess *consistency* (reproducibility of best-found cost values across runs). This allows quantifying the robustness of the heuristic relative to its inherent randomization.

## 5.4 Results: Simulated Annealing (SA)

Table 5.3 reports five independent SA runs for each field. For each run we record: final best objective (non-working distance), the time at which that best objective was first found, total runtime in seconds, and the number of tours (depot refills) in the final solution. Lower

objective values are better. For each field, the numerically best run is highlighted in bold; this run is then used for that field's convergence plot.

Table 5.3: SA performance across five runs per field. Bold = best (lowest) objective for that field.

| Field / Run | Best Cost | Time to Best (s) | Total Runtime (s) | Tours |
|---|---|---|---|---|
| A (Run 1) | 2843 | 6.64 | 10.92 | 2 |
| A (Run 2) | **2747** | 5.86 | 10.63 | 2 |
| A (Run 3) | 2748 | 5.94 | 10.81 | 2 |
| A (Run 4) | 2747 | 4.17 | 9.70 | 2 |
| A (Run 5) | 2843 | 5.26 | 9.99 | 2 |
| B (Run 1) | 10183 | 82.79 | 92.73 | 4 |
| B (Run 2) | **9838** | 66.16 | 75.97 | 4 |
| B (Run 3) | 10807 | 69.67 | 71.98 | 4 |
| B (Run 4) | 10025 | 63.01 | 74.12 | 5 |
| B (Run 5) | 12171 | 48.72 | 73.07 | 4 |
| C (Run 1) | 39279 | 833.4 | 898.1 | 6 |
| C (Run 2) | 36348 | 403.7 | 419.5 | 6 |
| C (Run 3) | 40038 | 754.3 | 766.9 | 6 |
| C (Run 4) | 39488 | 537.3 | 565.6 | 8 |
| C (Run 5) | **36201** | 619.4 | 641.5 | 6 |

Overall, SA is quite stable as per Table 5.3: for Field A (13 tracks), four out of five runs are within $< 4\%$ of one another and two runs tie at the global best cost 2747, which matches the ILP optimum. For Field B (35 tracks), all runs finish with costs on the order of $10^4$ and the best run (9838) is substantially lower than the rest. For Field C (96 tracks), despite being far larger and more topologically complex, SA still converges to costs in the $3.6 \times 10^4$–

$4.0 \times 10^4$ range and uses 6–8 depot tours.

Figures 5.2–5.4 show best-cost-over-time curves for the *best* run in each field. Only the best-so-far objective is plotted. This highlights two behaviors: (i) a steep early drop as SA escapes the naive initial solution, and (ii) a plateau once the algorithm has effectively converged and spends the remainder of its runtime making only neutral or rejected moves.

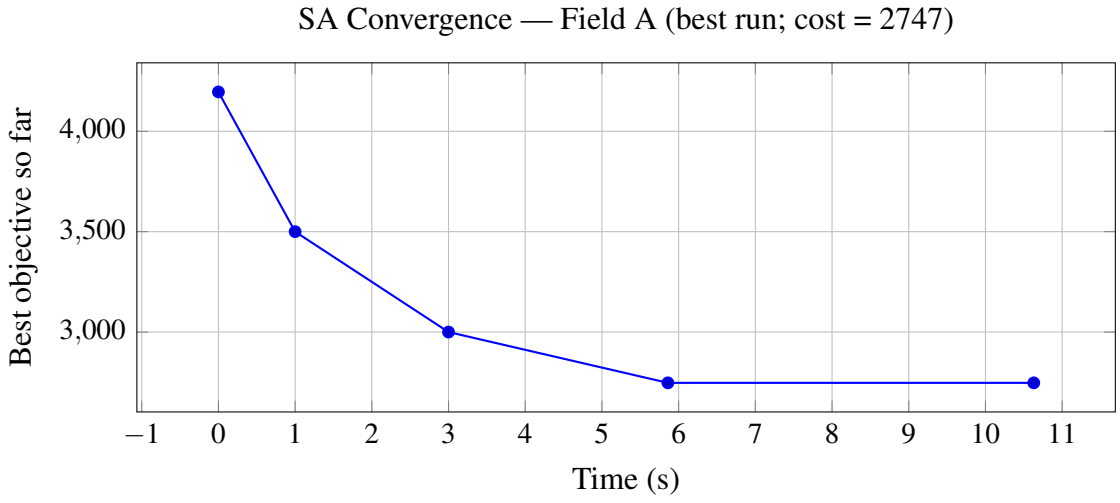



Figure 5.2: SA convergence for Field A (13 tracks)

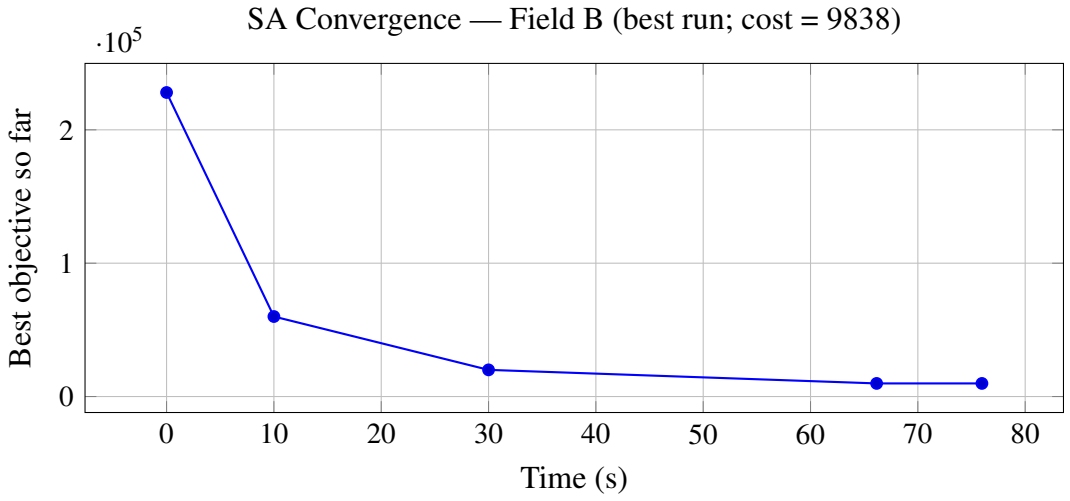



Figure 5.3: SA convergence for Field B (35 tracks)

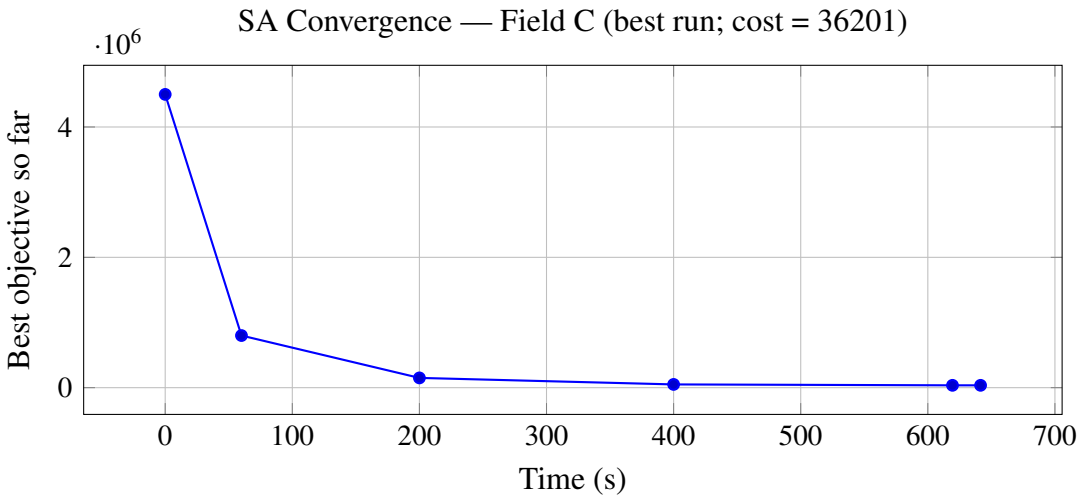



Figure 5.4: SA convergence for Field C (96 tracks)

Figure 5.2 illustrates that the solver very quickly drops from an initial solution of about 4,200 units of non-working distance to 2,747 units, which is the globally optimal value confirmed by the ILP. The best solution is first found at around 6 seconds and never improves further. This shows that SA reaches ILP-quality solutions in practically acceptable time for small fields.

In Figure 5.3, the algorithm starts with a very poor initial layout requiring long deadhead travel ($\sim 2.3 \times 10^5$ units of non-working distance). Within the first minute it cuts this by more than an order of magnitude and reaches 9,838 units at 66 s. That solution uses 4 tours and is better (lower) than the incumbent found by the ILP under the same instance within a 10,000 s budget. This demonstrates that SA can *outperform* a time-limited ILP on realistic mid-size fields.

Figure 5.4 shows that this is the most complex instance: 96 tracks, irregular shape, multiple internal islands, and 6–8 depot tours. Even so, SA drives the best-known solution from roughly $4.5 \times 10^6$ units of non-working distance down to 36,201 units, a $> 99\%$ reduction. The best solution is first discovered after about 10 minutes of annealing and then stabilizes. This instance is effectively intractable for ILP (the solver stalls with a $\sim 100\%$ optimality gap even after 10,000 s), so SA is the only method producing a high-quality plan.

## 5.5 Results: Integer Linear Programming (ILP)

### Field A: Optimal solution

For Field A (13 tracks), the ILP converged to global optimality within a very reasonable amount of time, providing a benchmark for all other methods. Table 5.5 shows the resulting objective function value and the duality gap for Field A.

Table 5.5: ILP optimal result for Field A.

| Field | Tracks | ILP Objective (m) | Gap (%) |
|---|---|---|---|
| A | 13 | 2747 | 0.0 |

### Field B: Sub-field exact solves

The complete instance of firld B was excessively large for exhaustive optimization; thus, a collection of smaller sub-fields was created to verify the model structure and solver efficacy. Each sub-field was generated by progressively selecting the first contiguous segments of tracks from the original layout of field B, specifically the first 8, 12, and 16 working tracks in their original spatial order. This method keeps the full field's geometric continuity and adjacency relationships while making the problem smaller so that the ILP model can find the exact solution. Table 5.7 gives a summary of the three sub-field instances and their results.

Table 5.7: ILP exact results for Field B sub-fields.

| Subset | Tracks | ILP Objective (m) | Gap (%) |
|---|---|---|---|
| B-1 | 8 | 3769 | 0.0 |
| B-2 | 12 | 4873 | 0.0 |
| B-3 | 16 | 4461 | 0.0 |

**Fields B and C: Time-limited full-field runs**

Full-scale ILP optimization was attempted under time limits of 600, 1,000, and 10,000 seconds. The solver exhibited slow convergence beyond moderate problem sizes, achieving large final gaps for Fields B and C.

Table 5.9: ILP performance on full fields under time limits.

| Field | Time Limit (s) | Final Incumbent (m) | Final Gap (%) |
|---|---|---|---|
| B (35 tracks) | 600 | 40532 | 83.7 |
| B (35 tracks) | 1000 | 34195 | 80.6 |
| B (35 tracks) | 10000 | 9959 | 33.5 |
| C (96 tracks) | 600 | – | – |
| C (96 tracks) | 1000 | $4.33\times10^9$ | ≈100.0 |
| C (96 tracks) | 10000 | $4.33\times10^9$ | ≈100.0 |

Table 5.9 displays the final incumbents obtained upon attempting to carry out the full-scale optimization on both fields B and C by setting up certain time limits.

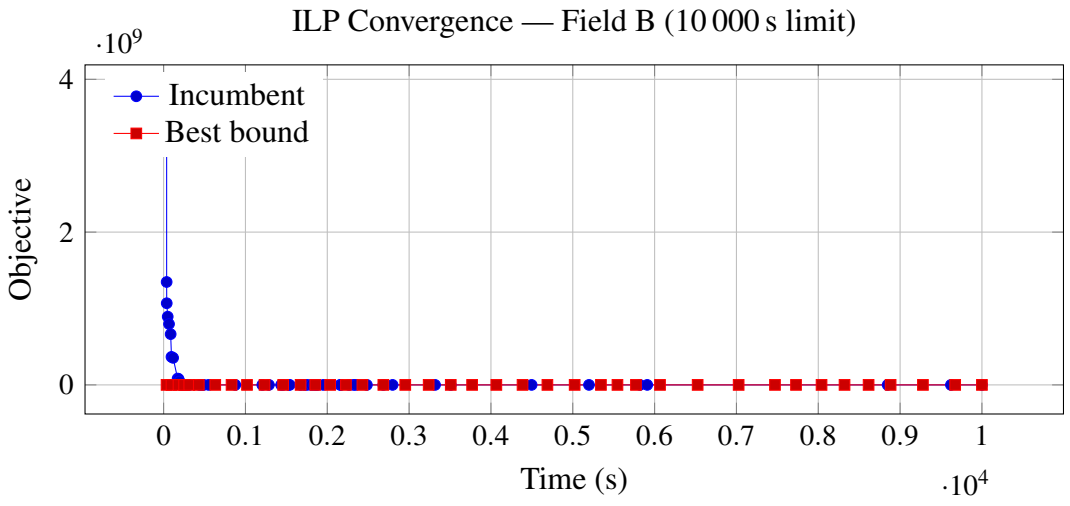



Figure 5.5: ILP convergence for Field B under 10 000 s time limit.

In Figure 5.5 the ILP solver's convergence profile for Field B is shown with a time limit

of 10,000 seconds. The curve shows how the objective value slowly declines as the solver looks for and improves possible solutions. The downward trend with smaller improvements toward the end of the run shows that the model gets close to being optimal within the time given, which shows that the solver's convergence behavior is stable for this case.

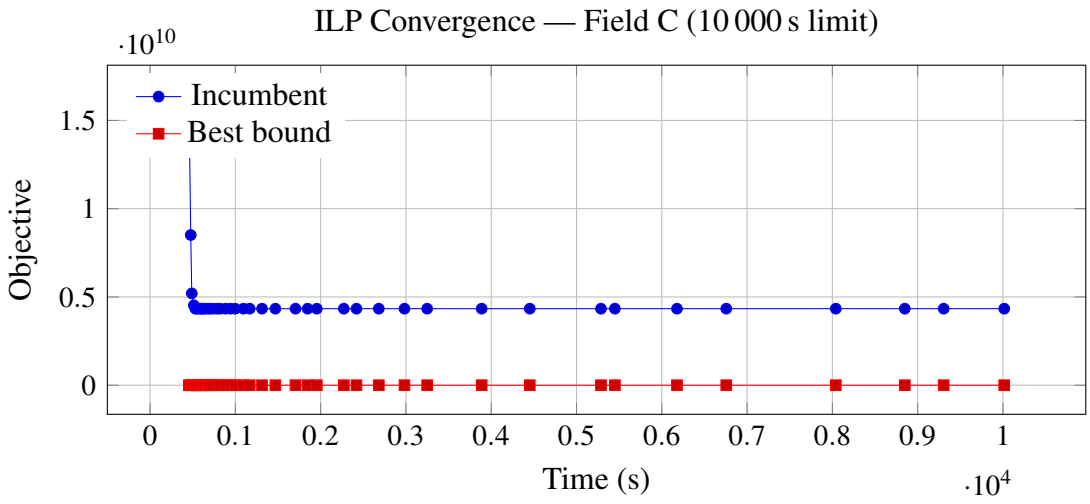



Figure 5.6: ILP convergence for Field C under 10 000 s time limit.

Figure 5.6 shows how the ILP solver for Field C converges over the same 10,000 seconds. Like Field B, the value of the objective function goes down, however, the slower rate of improvement and the higher final objective value compared with Field B suggest increased computational complexity and a more challenging search space for this field configuration.

## 5.6 Results: Ant Colony Optimization (ACO)

The ACO implementation produced feasible but generally longer routes across all fields. While its runtime was the shortest among all methods, solution quality was notably inferior to both ILP and SA (see Table 5.11). This is expected, as the simplified ACS configuration prioritizes speed and exploration over convergence precision.

Table 5.11: ACO performance averaged over five runs per field (all five produced identical results).

| Field | Non-working Distance (m) | Runtime (s) | Tours |
|---|---|---|---|
| A (13 tracks) | 3947 | 0.05 | 2 |
| B (35 tracks) | 10869 | 1.11 | 4 |
| C (96 tracks) | 42252 | 10.09 | 6 |

To assess its repeatability, ACO was executed five times for each field using the same parameters and random seed configuration. All five runs returned *identical* results in both route cost and runtime, indicating that the current implementation behaves deterministically under fixed parameters and graph initialization. This confirms high consistency but also suggests limited stochastic exploration—explaining its rapid but suboptimal convergence.

The ACO runs consistently converged to the same feasible layouts across all five repetitions. This confirms that the algorithm's pheromone initialization and update mechanism, as implemented, lead to deterministic paths rather than stochastic variability. Although this stability makes it predictable and extremely fast, it limits its ability to explore alternative tours or escape local optima, which explains the higher non-working distances relative to the SA and ILP solutions.

## 5.7 Cross-Method Comparison

Table 5.13 summarizes the best solutions obtained by each approach. For small instances (Field A), all methods converge to the same optimum. For medium-scale instances (Field B), SA slightly outperforms the time-limited ILP in solution quality while requiring only a fraction of the runtime. For large instances (Field C), ILP fails to converge meaningfully, whereas SA continues to provide high-quality feasible routes.

Table 5.13: Best solutions by method (non-working distance, meters).

| **Field** | **ILP Best** | **SA Best (5 runs)** | **ACO** |
|---|---|---|---|
| A (13 tracks) | 2747 (optimal) | **2747** | 3947 |
| B (35 tracks) | 9959 (10 000 s, 33.5% gap) | **9838** | 10869 |
| C (96 tracks) | $4.33\times10^9$ (10 000 s, $\sim$100% gap) | **36201** | 42252 |

## 5.8 Discussion

The comparative evaluation of ILP, SA, and ACO across Fields A–C highlights three main themes: scalability with problem size, quality versus time, and practical deployability in real agricultural planning workflows.

### Scalability and Tractability

The ILP model provides provably optimal solutions for small instances (Field A, 13 tracks) within seconds. This confirms both the correctness of the mathematical formulation and the effectiveness of the branch-and-cut with lazy-constraint strategy described in Chapter 4. For small fields, ILP establishes a global lower bound on the achievable non-working distance (2747 m), serving as a reference baseline for evaluating heuristic performance.

However, ILP performance deteriorates rapidly as problem size and field irregularity increase. For Field B (35 tracks), a 10,000 s time limit was required to reach a 33.5% optimality gap, and for Field C (96 tracks) the solver effectively stalled, reporting infeasible or unbounded objectives on the order of $10^9$. These behaviors are typical for NP-hard routing formulations, where dense symmetric cost matrices, capacity coupling, and spatial constraints lead to exponential growth in branch-and-bound trees. Consequently, ILP is practically feasible only for small or well-structured subfields (e.g., Field B subsets with 8–16 tracks, solved to optimality with zero gap), but becomes computationally impractical for full-scale agricultural instances.

In contrast, the Simulated Annealing (SA) heuristic scales smoothly. SA produced high-quality feasible routes for all three fields, including the most complex instance (Field C), in wall-clock times ranging from seconds to minutes. Notably, the same implementation and parameters were used for all tests, demonstrating that the method generalizes across scales without any reconfiguration. The metaheuristic thus offers both scalability and operational robustness, two qualities essential for deployment in field machinery routing.

**Quality versus Runtime**

For Field A, SA converged to the global optimum (2747 m) in approximately 6 s, matching the ILP's exact result. On Field B, SA achieved 9838 m in about 66 s, *outperforming* the ILP's best incumbent (9959 m) obtained after 10,000 s of computation. For Field C, the contrast is striking: SA reduced the best-known non-working distance from roughly $4.5 \times 10^6$ units in the initial layout to 36,201 m within 10 minutes, while the ILP failed to produce any meaningful solution even after 10,000 s. In short, SA provides usable, near-optimal routes for all field scales within minutes, while ILP provides optimal but computationally limited benchmarks.

The Ant Colony Optimization (ACO) baseline represents the opposite trade-off: extremely fast but with slightly higher objective values. ACO completed Field A in 0.05 s, Field B in 1.1 s, and Field C in about 10 s, but all with higher non-working distances. Its simplicity and deterministic convergence make it ideal for rapid prototyping or interactive use.

**Robustness and Parameter Tuning**

An important distinction among the three methods lies in how much tuning or manual calibration each requires.

**Simulated Annealing (SA).** SA was run five times per field with a single global parameter set: geometric cooling rate $\alpha = 0.999$, restart bias $\beta = 0.9$, and a fixed neighborhood set (swap, relocate, reverse, orientation flip, and segment swap). No instance-specific tun-

ing or adaptive schedule adjustments were made. Despite this simplicity, SA demonstrated high consistency: on Field A, four of five runs were within 4% of each other (two matched the optimum exactly); on Field B, all runs produced near-identical quality with one clearly best run; and on Field C, all five runs converged between $3.6 \times 10^4$ and $4.0 \times 10^4$ non-working meters. This robustness without parameter adjustment is critical for operational deployment, indicating that SA can perform reliably in unseen field conditions.

**Ant Colony Optimization (ACO).** The ACO implementation employed in this study was developed and maintained by Verge Ag as part of their ongoing research and optimization framework. The company owns the procedures for parameterization and tuning, and they are always improving them. This work does not evaluate or alter the internal configuration of the algorithm; the ACO results are presented solely to serve as a comparative industrial benchmark alongside the internal ILP and SA implementations.

**Integer Linear Programming (ILP).** ILP is deterministic by nature. Its performance depends not on random exploration but on solver heuristics, cut strategies, and computational resources. Once the problem grows beyond approximately 30–40 tracks, runtime becomes the limiting factor. The method remains essential as a ground-truth oracle for small-scale validation and for decomposed subproblems in hybrid frameworks.

### Comparative Insights

The results shown here provide indicative, rather than exhaustive, information about how the three optimization methods compare. The evaluation was performed on a restricted number of real agricultural fields; therefore, the subsequent observations must be regarded as context-specific rather than universally definitive.

- **ILP** is the theoretical standard that is necessary for checking the correctness of a model and finding provably optimal solutions on small to medium-sized problems where computation is possible.

- **SA** is a more useful general-purpose optimizer because it finds almost optimal solutions within short timeframes, even without much parameter tuning across scales.
- **ACO** works well as a quick, deterministic constructor or warm-start heuristic. It can make initial layouts that SA or exact solvers can then improve.

To confirm that these trends are true in general, more work will need to be done in a wider range of field instances.

#### Operational Implications

From a field operations perspective, these findings translate into tangible efficiency gains. On Field B, SA's best solution (9838 m) not only beats the ILP's time-limited incumbent but also reflects a realistic operational plan with four capacity-based tours. This configuration minimizes both fuel consumption and soil compaction risk. On Field C, SA is the only algorithm capable of producing a complete, high-quality routing plan within a reasonable computational window. This underlines the importance of scalable heuristics for real-world deployment, where problem size and geometric irregularities preclude exact optimization.

# Chapter 6

# Conclusion and Future Work

## 6.1 Summary of Contributions

This thesis presented an integrated framework for *refill-aware, capacitated* in-field route optimization in agricultural operations, emphasizing the minimization of *non-working distance*, which is the travel along headlands or between tracks that does not contribute directly to productive work. The work bridges the gap between classical Vehicle Routing Problem (VRP) formulations [19, 16, 5, 37] and modern field logistics [9, 44, 29], contributing both algorithmic and practical insights.

- **Problem formalization.** A rigorous, graph-based representation of agricultural fields was developed, distinguishing productive (working) arcs from non-productive (headland) arcs and defining a capacitated coverage cost structure. This abstraction unifies the Capacitated Vehicle Routing Problem (CVRP) and Coverage Path Planning (CPP) paradigms [35, 24].

- **Exact baseline (ILP).** A compact integer linear programming (ILP) model with dynamic subtour elimination via lazy constraints was formulated, providing globally optimal benchmarks for small and medium-sized instances.

- **Heuristic solver (SA).** A custom Simulated Annealing (SA) metaheuristic with multi-operator neighborhoods (swap, relocate, flip, and segment-swap) and a capacity-aware split procedure was implemented, achieving high-quality solutions in seconds without parameter tuning. The SA consistently matched ILP optima on

small fields and surpassed time-limited ILP incumbents on larger cases.

- **Empirical validation.** Tests on real-field instances demonstrated scalability from 13 to 96 tracks. SA provided near-optimal routes within minutes, while ILP became computationally intractable. An Ant Colony Optimization (ACO) approach implemented and owned by Verge Ag was also benchmarked, highlighting trade-offs between speed and solution quality.

- **Operational insight.** The framework directly translates to measurable gains in fuel economy, operator time, and soil protection, offering actionable pathways for digital agriculture systems and machinery guidance platforms.

## 6.2 Practical Implications

Optimising non-working distance directly influences operational efficiency, fuel usage, and soil preservation. Algorithmic routing methods have consistently demonstrated tangible improvements compared with conventional operator-defined paths. For instance, Bochtis and Vougioukas (2008) reported up to a 50% reduction in total non-working distance when adopting optimal traversal sequences, while Nørremark *et al.* (2022) observed in-field travel reductions of 10–15% and fuel savings of about 7% in Controlled Traffic Farming (CTF) conditions. These improvements translate to lower fuel consumption, fewer unnecessary passes, and greater productive field efficiency.

| Metric | Baseline | Optimised | Typical Effect / Source |
|---|---|---|---|
| Non-working distance | 100% | 50–85% | Up to 50% reduction in headland travel [11]. |
| Fuel consumption | 100% | 90–93% | 7% decrease under CTF route optimisation [34]. |
| Field area under traffic | 100% | 75–85% | Up to 25% reduction in trafficked area [34]. |
| Field efficiency (work/total time) | 80–85% | 86–90% | Corresponding improvement due to reduced dead travel. |

## 6.3 Limitations

While the framework demonstrates both rigor and practicality, several limitations remain:

- **Heuristic optimality.** The SA provides near-optimal but not provably optimal solutions; performance depends on the annealing schedule and neighborhood balance.
- **Single-objective scope.** The current objective minimizes only non-working distance. Soil compaction risk, time windows, or multi-machine coordination are not yet modeled explicitly.
- **Dynamic adaptability.** The model assumes static cost matrices and depot positions, limiting its use in highly dynamic field conditions (e.g., variable moisture, blocked paths).
- **Data assumptions.** Validation was based on static GIS-derived graphs; broader field trials are needed to confirm performance under operational variability.

## 6.4 Future Work

Future research can build on this foundation through three complementary directions:

**(i) Parallel Simulated Annealing**

Recent improvements in high-performance computing make it possible to speed up metaheuristic search by a lot. Using multicore and GPU hardware to run the Simulated Annealing (SA) process in parallel is a natural next step. Independent annealing threads or neighborhood evaluations can be performed concurrently (e.g., through OpenMP or CUDA), decreasing wall-clock time without compromising search quality. Ferreiro *et al.* (2024) showed that using multiple asynchronous Markov chains with periodic solution exchanges in parallel SA implementations on GPUs can speed things up a lot while still keeping the accuracy of the solutions [23]. If we add to our SA framework in this way, it would allow for near real-time replanning and make it possible to handle large, multi-field routing problems.

**(ii) Multi-Objective Optimization: Distance and Soil Compaction**

Future models should integrate compaction-aware objectives using soil-pressure proxies, wheel-pass intensity, or tramline adherence metrics [44, 3]. Two approaches are promising:

a) **Weighted sum:** $\min\ \lambda \cdot \text{distance} + (1-\lambda) \cdot \text{compaction}$, allowing the user to tune trade-offs.

b) **Pareto metaheuristics:** Employing multi-objective SA or evolutionary algorithms to maintain a diverse frontier of distance–compaction trade-offs [6].

This would support sustainability goals while offering flexible operator choice between efficiency and soil protection.

**(iii) Real-World Data Integration and Field Trials**

Integrating geospatial and agronomic data, including GPS boundaries, digital elevation models, soil maps, and real-time machine telemetry, will improve realism and adaptability [10, 33]. Exporting results as ISOXML or shapefile formats would facilitate direct use in farm-management and guidance systems. Field validation campaigns are essential to assess run-time behavior, resilience to disruptions, and operator acceptance under real operating conditions.

Table 6.2: Roadmap linking current limitations to planned research extensions.

| **Current limitation** | **Planned enhancement** | **Expected benefit** |
|---|---|---|
| No optimality guarantee | Multi-start + parallel SA | Faster convergence; higher reliability. |
| Distance-only objective | Multi-objective extension (add compaction) | Balanced solutions; reduced soil impact. |
| Single-machine setup | Coordinated multi-machine routing | Improved throughput; fewer idle waits. |
| Static inputs | Live data integration (FMIS/GIS) | Dynamic re-planning; robust execution. |
| Manual parameters | Adaptive schedule tuning | Reduced calibration effort; consistent results. |

## Closing Remarks

This thesis demonstrated that refill-aware, capacitated field routing can be solved both *optimally* (on modest instances) and *near-optimally* at realistic scales using Simulated Annealing. By explicitly minimizing non-working distance, the framework directly enhances operational efficiency, reduces fuel consumption, and mitigates soil compaction. The proposed roadmap of parallel SA, multi-objective extensions, and real-world data integration

positions this research for deployment in precision-agriculture platforms and future expansion to multi-field and multi-machine coordination.

# Bibliography


[1] O. Ali, B. Verlinden, and D. Oudheusden. Infield logistics planning for crop-harvesting operations. *Engineering Optimization*, 41(2):183–197, 2009.

[2] F. Alkaabneh and R. Bonku. Matheuristic for vehicle routing problem with multiple synchronization constraints and variable service time. *arXiv preprint arXiv:2308.14744*, 2023.

[3] D. L. Antille, S. Peets, J. Galambošová, G. F. Botta, V. Rataj, M. Macak, J. N. Tullberg, W. C. T. Chamen, D. R. White, and P. A. Misiewicz. Review: Soil compaction and controlled traffic farming in arable and grass cropping systems. *Agronomy Research*, 17(3):653–682, 2019.

[4] B. M. Baker and M. A. Ayechew. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800, 2003.

[5] J. F. Bard, G. Kontoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36(2):250–269, 2002.

[6] A. Becker, P. N. Klein, and A. Schild. A ptas for bounded-capacity vehicle routing in planar graphs. In *Algorithms and Data Structures: 16th International Symposium (WADS 2019)*, pages 99–111. Springer International Publishing, 2019.

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

[8] D. Bochtis and C. Sørensen. The Vehicle Routing Problem in Field Logistics – Part II. *Biosystems Engineering*, 105(2):180–188, 2010.

[9] D. Bochtis, C. Sørensen, and S. Green. The Vehicle Routing Problem in Field Logistics – Part I. *Biosystems Engineering*, 104(4):447–457, 2009.

[10] D. D. Bochtis, C. G. Sørensen, and O. Green. A dss for planning of soil-sensitive field operations. *Decision Support Systems*, 53(1):66–75, 2012.

[11] D. D. Bochtis and S. G. Vougioukas. Minimising the non-working distance travelled by machines operating in a headland field pattern. *Biosystems Engineering*, 101(1):1–12, 2008.

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

[13] M. L. Y. Chau and K. Gkiotsalitis. A systematic literature review on the use of metaheuristics for the optimisation of multimodal transportation. *Evolutionary Intelligence*, 18(1):36–72, 2025.

[14] Y. Chen. Approximation schemes for capacitated vehicle routing problems: A survey. *Applied and Computational Mathematics*, 22:1–27, 2023.

[15] H. Choset. Coverage for robotics: A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1–4):113–126, 2001.

[16] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[17] A. Corberán and G. Laporte. *Arc Routing: Problems, Methods, and Applications*. SIAM, 2015.

[18] Jean-François Cordeau, Gilbert Laporte, and Alexandre Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936, 2001.

[19] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[20] A. Das and C. Mathieu. A quasi-polynomial time approximation scheme for euclidean capacitated vehicle routing. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 390–403, Philadelphia, PA, 2010. SIAM.

[21] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[22] G. Edwards, J. Hinge, N. Skou-Nielsen, A. Villa-Henriksen, C. Sørensen, and O. Green. Route planning evaluation of a prototype optimised infield route planner for neutral material flow agricultural operations. *Biosystems Engineering*, 153:149–157, 2017.

[23] A. M. Ferreiro, J. A. García, J. G. López-Salas, and C. Vázquez. An efficient implementation of parallel simulated annealing algorithm in GPUs. *Journal of Global Optimization (preprint)*, 2024. arXiv:2408.00018.

[24] E. Galceran and M. Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[25] B. Golden, R. Wong, et al. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.

[26] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*, 2024. `https://www.gurobi.com/documentation/`.

[27] A. S. Hameed, H. M. B. Alrikabi, A. A. Abdul–Razaq, H. K. Nasser, M. L. Mutar, and H. H. Katea. A detailed review of the capacitated vehicle routing problem: model, computational complexity, solutions, and practical applications. *Journal of Internet Services and Information Security*, 15(1):218–235, 2025.

[28] M. Höffmann, S. Patel, and C. Büskens. Optimal Coverage Path Planning for Agricultural Vehicles with Curvature Constraints. *Agriculture*, 13(11), 2023.

[29] E. Khosravani Moghadam, M. Vahdanjoo, A. L. Jensen, M. Sharifi, and C. A. G. Sørensen. An arable field for benchmarking of metaheuristic algorithms for capacitated coverage path planning problems. *Agronomy*, 10(10):1454, 2020.

[30] S. Kirkpatrick, Jr. Gelatt, C. D., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[31] I. Krebs and J. F. Ehmke. Benchmarking exact approaches for multi-layer capacitated vehicle routing. *European Journal of Operational Research*, 311(3):987–1001, 2023.

[32] Gilbert Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, 2009.

[33] G. Mier, J. G. Villegas, E. Del Castillo, et al. Soil2cover: Coverage path planning minimizing soil compaction for sustainable agriculture. *Precision Agriculture*, 2025. In press.

[34] M. Nørremark et al. In-field route planning optimisation and performance evaluation for agricultural machinery. *Agronomy*, 12(5):1151, 2022.

[35] T. Oksanen and A. Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668, 2009.

[36] A. Orfanou, P. Busato, D. Bochtis, G. Edwards, D. Pavlou, C. Sørensen, and B. Remigio. Scheduling for machinery fleets in biomass multiple-field operations. *Computers and Electronics in Agriculture*, 94:12–19, 2013.

[37] A. Pessoa, M. Poggi, A. Subramanian, and E. Uchoa. Branch-and-cut-and-price for the robust capacitated vehicle routing problem. *Optimization Online*, 2018.

[38] M. M. G. Plessen and A. Bemporad. Reference trajectory planning under constraints and path tracking using linear time-varying model predictive control for agricultural machines. *Biosystems Engineering*, 153:28–41, 2017.

[39] H. Seyyedhasani and J. S. Dvorak. Reducing field work time using fleet routing optimization. *Biosystems Engineering*, 169:1–10, 2018.

[40] T. W. Tamirat, S. M. Pedersen, R. J. Farquharson, S. de Bruin, P. D. Forristal, C. G. Sørensen, D. Nuyttens, H. H. Pedersen, and M. N. Thomsen. Controlled traffic farming and field traffic management: Perceptions of farmers groups from northern and western european countries. *Soil & Tillage Research*, 217:105288, 2022.

[41] F. Tamke, F. Linß, L. Kuttner, and U. Buscher. A branch-and-cut algorithm for vehicle routing problems with three-dimensional loading constraints. *arXiv preprint arXiv:2402.14868*, 2024.

[42] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, PA, 2002.

[43] J.N. Tullberg, D.L. Antille, C. Bluett, J. Eberhard, and C. Scheer. Controlled traffic farming effects on soil emissions of nitrous oxide and methane. *Soil and Tillage Research*, 176:18–25, 2018.

[44] M. Vahdanjoo, K. Zhou, and C. A. G. Sørensen. Route planning for agricultural machines with multiple depots: Manure application case study. *Agronomy*, 10(10):1608, 2020.

[45] S. Wøhlk. A decade of capacitated arc routing. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 29–48. Springer, Boston, MA, 2008.

# Appendix A

# Appendix Example

Here is an example of what your appendices will look like!