AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Projekt Dyplomowy

Modeling and Solving the Bin Packing Capacitated Vehicle Routing Problem

Modelowanie oraz rozwiązywanie łączonego problemu pakowania i marszrutyzacji

Autor:	*Natalia Kwiecień*
Kierunek studiów:	*Informatyka i Systemy Inteligentne*
Opiekun pracy:	*dr inż. Mateusz Ślażyński*

Kraków, 2026

Pragnę z całego serca podziękować moim Rodzicom, Joannie i Kamilowi Kwiecień, za nieustanne wsparcie oraz wiarę we mnie. Słowa wdzięczności kieruję również do mojej Babci, Ireny Kwiecień, która zaszczepiła we mnie zamiłowanie do matematyki i wspierała mnie w jego rozwijaniu.

Serdeczne podziękowania składam również dr inż. Mateuszowi Ślażyńskiemu za wsparcie merytoryczne, cenne wskazówki oraz motywację podczas realizacji pracy inżynierskiej.

Contents

1. Introduction

Packing and routing problems belong to the core of operations research. Many classical optimisation models are refinements or extensions of two fundamental problems: the Knapsack Problem, which captures decisions about limited resources, and the Travelling Salesman Problem, which captures the structure of efficient routing. The **Bin Packing Problem (BPP)** and the **Vehicle Routing Problem (VRP)** can be seen as natural generalisations of these two foundations, enriched with additional structures that make them directly applicable to practical settings.

Because of their close relationship to these fundamental problems, packing and routing models appear in a wide range of applications and continue to attract attention in the literature. At the same time, even small extensions to their basic formulations can lead to a rapid increase in computational difficulty. This makes them a useful testing ground for studying how different modelling choices influence both a model's expressiveness and its tractability.

In this thesis, these problems are approached from a constraint programming perspective. Rather than focussing on the design of specialised algorithms, the emphasis is placed on modelling: on expressing problem structure through decision variables and constraints, and on observing how this structure interacts with the solver's search process. This perspective provides an alternative way of studying classical optimisation problems and offers a flexible framework for experimenting with increasingly rich formulations.

1.1. Background of the study

Packing and routing problems have a longstanding history in operations research. The BPP and the VRP have been thoroughly studied, resulting in a rich body of exact methods, heuristics, and approximation methods. Many extensions have been developed to incorporate real-world constraints, such as capacity limits, time windows, and loading rules. As a result, the literature on these problems is vast, and their algorithmic properties are well understood.

In contrast, models that explicitly combine packing and routing decisions remain comparatively rare. While some studies integrate detailed loading constraints directly into routing, the explicit modelling of palletisation as an intermediate decision layer has received much less attention. Apart from a small number of contributions—most notably by Liu and co-authors, Moura, and Zachariadis—this two-level structure is largely absent from the literature. Yet, it introduces modelling challenges that are distinct from those of classical routing or packing problems considered in isolation.

This gap makes pallet-based integrated models a particularly interesting object of study. They sit at the intersection of two well-understood problem classes, but their interaction raises new questions about feasibility, demand representation, and computational complexity. These questions provide the context for the exploration and experimental evaluation of the models developed in this thesis.

1.2. Research problem and objectives

This thesis explores how classical packing and routing problems can be formulated using a **constraint programming** approach. Instead of developing specialised algorithms, the focus is on declaratively formulating these problems, identifying appropriate decision variables and constraints, and analysing how different modelling choices affect solver behaviour. This includes both standalone formulations of the BPP and the VRP, as well as increasingly integrated models that combine the two.

In particular, the thesis aims to address four key questions: First, how does the computational cost change when moving from basic routing formulations to fully integrated packing-and-routing models? Second, within a fixed time limit, how do these models compare in terms of the quality of the solutions they produce and their ability to prove optimality? Third, which instances remain solvable under restrictive assumptions, such as single visits, and at what point do split deliveries become necessary for restoring feasibility? Finally, how does a sequential heuristic compare to a fully integrated model in terms of solution quality, and how much computational effort does it save?

Together, these objectives serve a broader purpose. By studying a problem that is computationally demanding even at moderate scales, the thesis aims to evaluate whether integrated models are worth investigating in practice and under what conditions simpler or sequential approaches may provide a reasonable compromise. In this sense, the work can be viewed as a proof of concept that explores the limits of constraint programming in modelling integrated packing and routing problems and the insights gained from this exploration.

1.3. Thesis structure

The remainder of the thesis is organised as follows. Chapter 1 reviews the relevant literature on the BPP, the VRP, and integrated routing-and-packing models, with an emphasis on two-level pallet-based formulations. Chapter 3 presents the constraint programming formulations used in the thesis, beginning with baseline models and progressing to fully integrated variants. Chapter 4 describes the experimental methodology and presents the results. Finally, Chapter 5 concludes the thesis and outlines directions for further work.

2. Literature Review

The **Bin Packing–Capacitated Vehicle Routing Problem (BP–CVRP)** combines two well-known combinatorial optimisation problems: the Bin Packing Problem and the Vehicle Routing Problem. Before introducing the integrated model in later chapters, it is important to review the classical versions of these components, the main algorithmic techniques developed for them, and the ways in which packing and routing have been brought together in the literature.

This chapter is organised as follows. Section 2.1 recalls the definition of the BPP and summarises the main exact, approximation, and heuristic approaches for solving it. Section 2.2 does the same for the VRP, with an emphasis on heuristic and metaheuristic methods. Section 2.3 then reviews models that integrate routing and packing decisions. The chapter ends with a short summary that highlights the ideas from the literature that are most relevant to the models developed later in the thesis.

Although this thesis adopts a constraint-programming modelling perspective, the literature on the BPP and the VRP is dominated by exact and heuristic algorithmic approaches. Reviewing these approaches is useful for understanding which structures are typically exploited (bounds, symmetries, decompositions), which later inform modelling choices and simplifications.

2.1. Bin Packing Problem

The **Bin Packing Problem** represents the first component of the BP–CVRP. Originating from early studies on resource allocation and cutting processes in the mid-twentieth century [1], the problem has since found numerous applications across operations research, computer science, and engineering, including manufacturing, container loading, stock cutting, scheduling, and data storage. This section introduces classical BPP, outlines the main approaches for solving it, and discusses its major variants.

2.1.1. Definition of the Bin Packing Problem

The BPP concerns arranging a collection of items of different sizes into a limited number of bins, each with the same fixed capacity, in such a way that the total number of bins used is minimised.

Formally, given a bin capacity C and a list of items $L = (p_1, p_2, \ldots, p_n)$, where each item p_i has a size $s(p_i)$ satisfying $0 \leq s(p_i) < C$, find the smallest integer m such that L can be partitioned into m disjoint subsets $B_1, B_2, \ldots, B_m$, where the total size of the items in each subset does not exceed C. Each subset B_j corresponds to the contents of a single bin with a capacity of C [2].

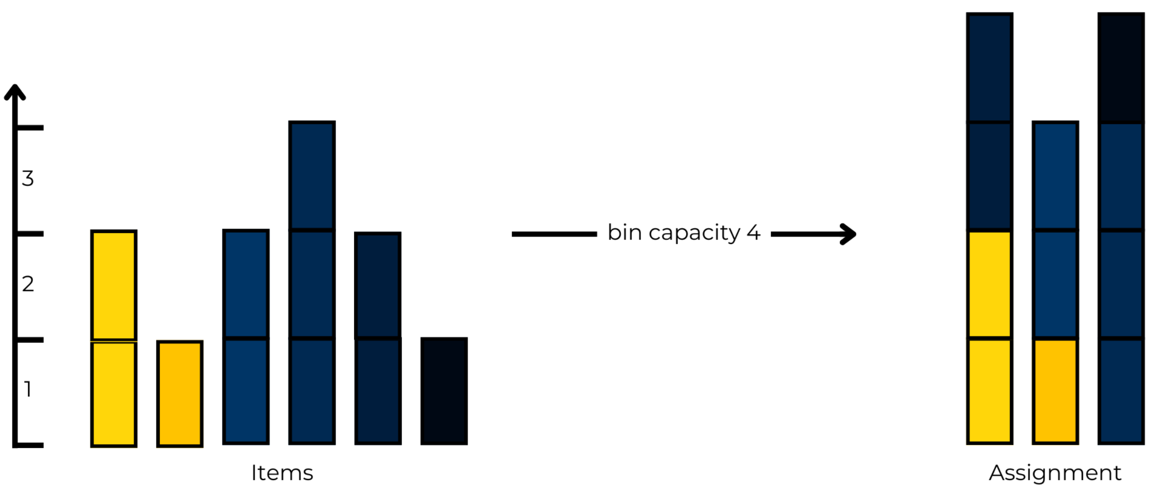


Figure 2.1. This figure shows the procedure of the BPP, which assigns a list of items to the smallest number of bins with a given capacity.

Source: own work.

According to [3], the BPP is strongly NP-complete, which motivates the development of heuristic and approximation algorithms capable of producing near-optimal solutions within a reasonable computational time.

2.1.2. Solution Approaches for the Bin Packing Problem

Solution approaches for almost any optimisation problem can be divided into three main categories: finding an exact solution, finding an approximation (often used to estimate a lower bound that guides the search), or using a heuristic, which sacrifices guaranteed optimality for faster computation.

A solution approximation, following the definition in [4], is a strategy for finding a solution for which the worst-case performance ratio is known. Some optimisation methods benefit from having an initial feasible solution, while others exploit good lower bounds to reduce the search space; hence, approximation algorithms are important.

One of the simplest classes of algorithms for the BPP is online algorithms. These methods process items sequentially, considering only the current state of the packing and the next item to be placed, without knowledge of future items. Popular examples include the Next-Fit (NF), First-Fit (FF), and Best-Fit (BF) algorithms. In all of them, some bins are considered open or closed, and new items may only be placed into open bins. Better solutions are typically achieved with FF and BF, where all bins remain open and a new item is placed either into the first bin it fits (FF) or into the bin that will have the least remaining space after placement (BF) [4].

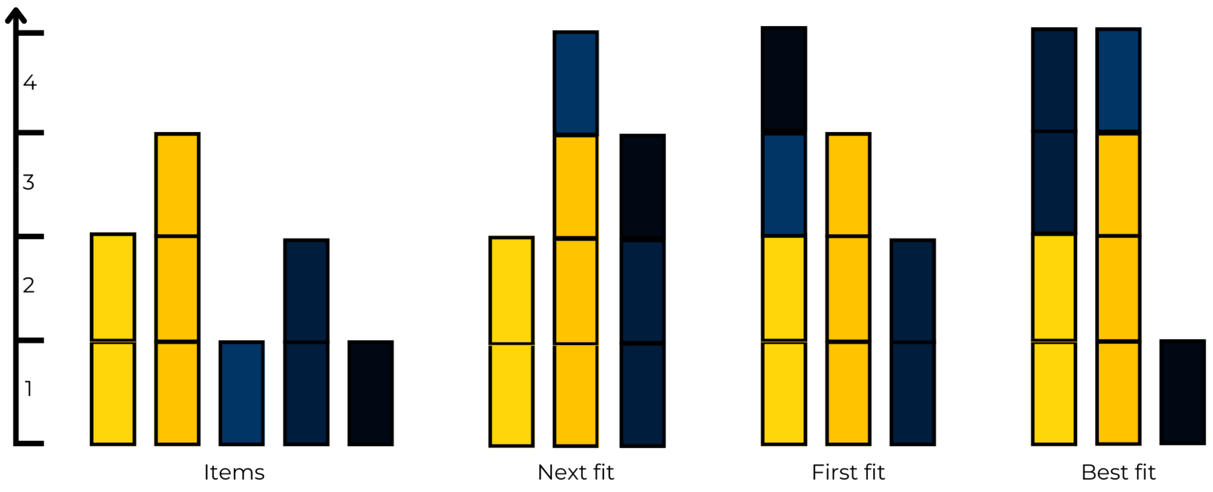


Figure 2.2. Example illustrating how Next-Fit, First-Fit, and Best-Fit place the same sequence of items into bins. Next-Fit considers only the current bin, while First-Fit and Best-Fit check all open bins, leading to more compact packings.

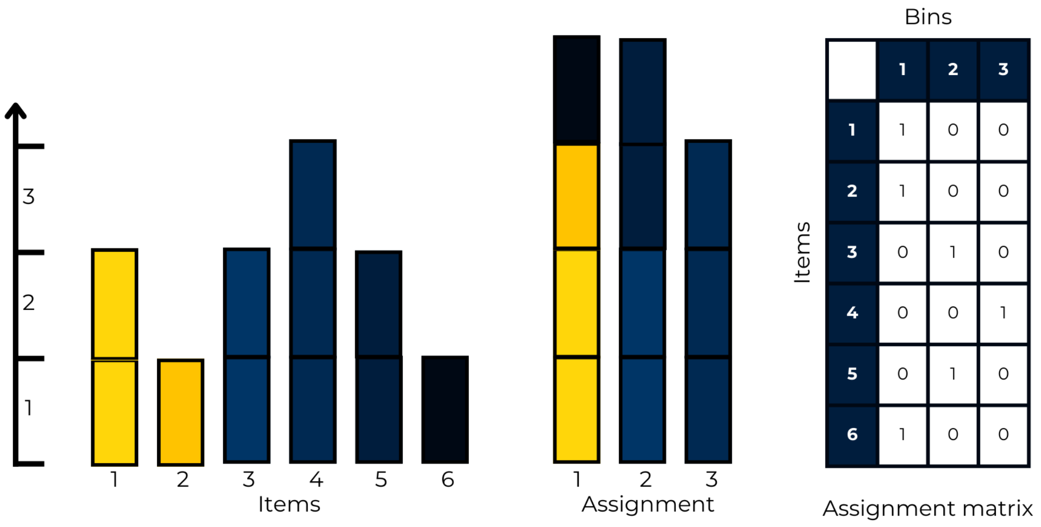
Source: own work.

A different approach involves pseudo-polynomial formulations, which, somewhat counter-intuitively, introduce additional variables to strengthen the model. This results in tighter linear relaxations and helps eliminate symmetries that often weaken the classic integer linear programming formulation originally proposed by Kantorovich [1].

While the methods described above focus on generating good solutions, reducing the search space, or modelling the problem in a way that eliminates symmetric solutions, a further challenge is to explore the search space and identify an optimal solution, in other words, to model the search strategy itself. Such methods include Branch-and-Bound and the more advanced Branch-and-Price.

In the Branch-and-Bound approach, the solution space is recursively divided into smaller subproblems (branches), and each subproblem is evaluated using upper and lower bounds to discard those that cannot contain an optimal solution. In this way, it is possible to avoid conducting an exhaustive search.

In addition to exact and pseudo-polynomial approaches, a wide range of heuristic and metaheuristic methods has been developed to solve large-scale or highly constrained instances of the BPP. These methods trade guaranteed optimality for scalability and shorter computation times, making them suitable for real-world industrial applications. Population-based metaheuristics, such as Genetic Algorithms, Memetic Algorithms, and Ant Colony Optimization, explore many solutions simultaneously through evolutionary or cooperative mechanisms. Trajectory-based approaches, including Tabu Search, Simulated Annealing, and the Greedy Randomised Adaptive Search Procedure (GRASP), focus on iteratively improving a single solution using local search and diversification strategies. Although these methods do not guarantee an optimal result, they can efficiently produce high-quality solutions for large or complex problem instances and are therefore widely used in practice [5, 4].



	1	2	3
1	1	0	0
2	1	0	0
3	0	1	0
4	0	0	1
5	0	1	0
6	1	0	0

Figure 2.3. Because bins are interchangeable, the same packing can be represented by many different assignment matrices. Sorting bins by their total load reduces this symmetry, but a stronger option is to sort the columns of the assignment matrix lexicographically. This leads to a more compact search space in pseudo-polynomial formulations.

Source: own work.

2.2. Vehicle Routing Problem

The **Vehicle Routing Problem (VRP)** consists of designing a set of routes for a fleet of vehicles that start and end at a depot and serve a set of geographically distributed customers. Each customer must be visited in order to satisfy its demand, while operational constraints such as vehicle capacity must be respected. The objective is typically to minimise the total travel cost or distance.

2.2.1. Definition of the Vehicle Routing Problem

The aim of the classical VRP is to find a set of routes, all starting and ending at a depot, for a fleet of vehicles, such that each customer is visited exactly once and the total routing cost is minimised.

Formally, let us consider an undirected graph $G = (V, E)$, where $V = \{v_0, v_1, \ldots, v_n\}$ is the set of vertices and $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is the set of edges. The vertex v_0 represents the depot, and the remaining vertices correspond to customer locations. Each edge (v_i, v_j) has an associated non-negative cost c_{ij}, typically representing distance or travel time. It is assumed that the cost matrix is symmetric. Each customer v_i has a demand d_i, and each vehicle has a capacity limit Q. The task is to construct at most m routes such that: (1) each route starts and ends at the depot; (2) each customer is visited exactly once by one vehicle; (3) the total demand on each route does not exceed the vehicle capacity; (4) the total cost of all routes is minimised [7].

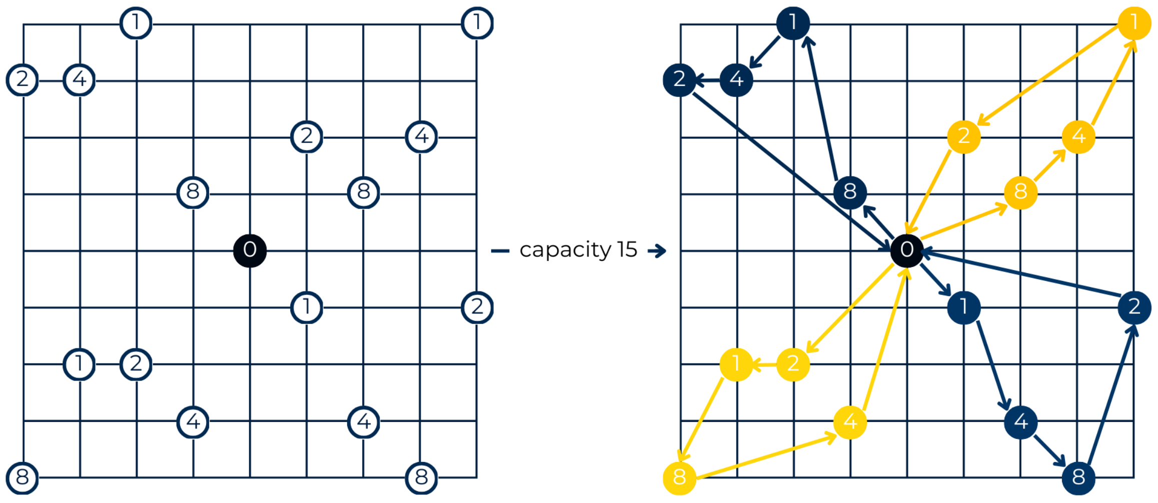


Figure 2.4. Example of a VRP instance: a depot and a set of customers that must be served by capacity-limited vehicles. The goal is to design routes that start and end at the depot and minimise total travel cost.

Source: own work (adapted from Google OR-Tools [6]).

VRP generalises the classical Travelling Salesman Problem and is therefore NP-hard [3]. Even relatively small instances can be computationally challenging, which has motivated the development of a wide range of exact, heuristic, and metaheuristic solution approaches.

2.2.2. Solution Approaches for the Vehicle Routing Problem

The VRP generalises the Travelling Salesman Problem, and it becomes difficult very quickly as the number of customers grows. Exact methods based on integer programming and branch-and-cut are therefore mostly used for smaller instances or to compute strong lower bounds [7]. For larger and more realistic variants, the literature focusses mainly on heuristic approaches.

Classical heuristics often build an initial set of routes and then improve it. The Sweep Algorithm is representative of planar instances: customers are sorted by polar angle around the depot and assigned in that order, starting a new route whenever capacity would be violated [8]. Closely related are cluster-first, route-second methods, which first partition customers into clusters and then solve a TSP-like routing problem within each cluster.

Compared to classical heuristics, metaheuristics is a more general-purpose framework designed to explore the solution space more thoroughly. They often incorporate randomisation, memory, or adaptive strategies, which allow them to escape local optima and perform well across a wide range of problem instances.

One of the most common strategies used within metaheuristics is local search. These methods start from an initial solution and iteratively move to a neighbouring one by applying small changes, such as

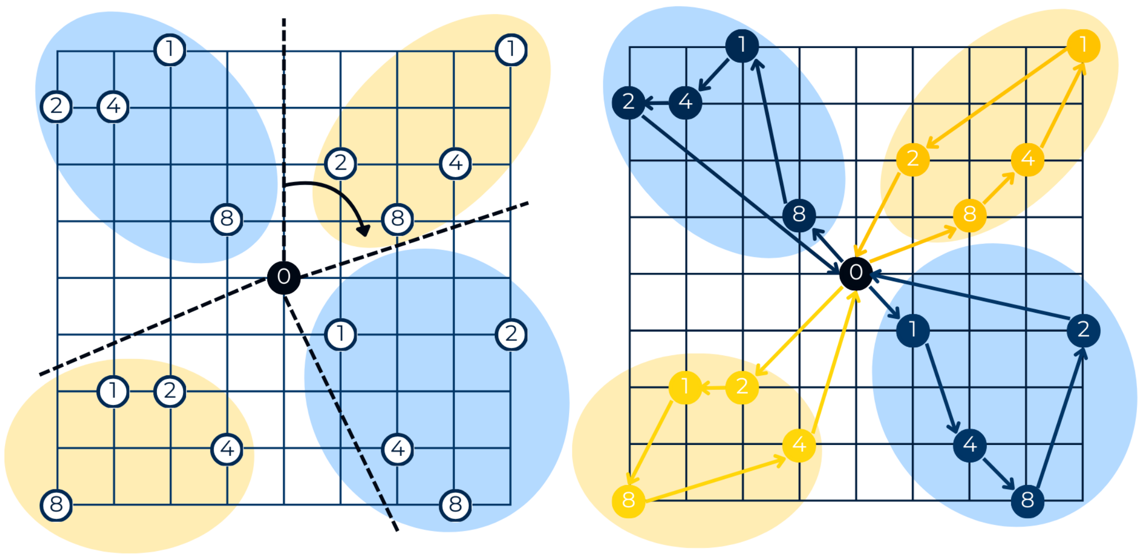


Figure 2.5. Illustration of the sweep algorithm for the VRP. Customers are sorted by their polar angle around the depot and assigned to vehicles while respecting vehicle capacity limits. Each colour represents a capacity-feasible cluster, and a route is then constructed within each cluster.

Source: own work (adapted from Google OR-Tools [6]).

swapping, inserting, or removing customers from routes. The process continues as long as improvements can be found. Local search is often used as a building block within more advanced metaheuristics, and can also be applied as a post-optimisation step to refine solutions obtained by constructive heuristics.

To avoid getting stuck in poor local optima, local search is often embedded into higher-level strategies that introduce memory or controlled randomisation. Tabu Search uses a tabu list to discourage cycling and has been particularly influential for routing problems [9, 10].

Several other metaheuristics have also been successfully applied to the VRP. Genetic Algorithms (GA) and Ant Colony Optimisation (ACO) are population-based methods that construct or evolve solutions through repeated iterations. GA applies operators inspired by natural selection—such as crossover and mutation—while ACO mimics the behaviour of ants finding paths by laying and following pheromone trails [11]. Both methods require careful parameter tuning and are often combined with local search to improve solution quality.

Another group of methods focusses on combining greedy construction with local improvement. GRASP (Greedy Randomised Adaptive Search Procedure) builds multiple solutions using a randomised greedy heuristic, followed by local search [12]. Variable Neighbourhood Search (VNS) explores different neighbourhood structures in a systematic way, changing the scope of the search as needed [13]. Both GRASP and VNS are relatively easy to implement and have been shown to perform well on a variety of VRP instances.

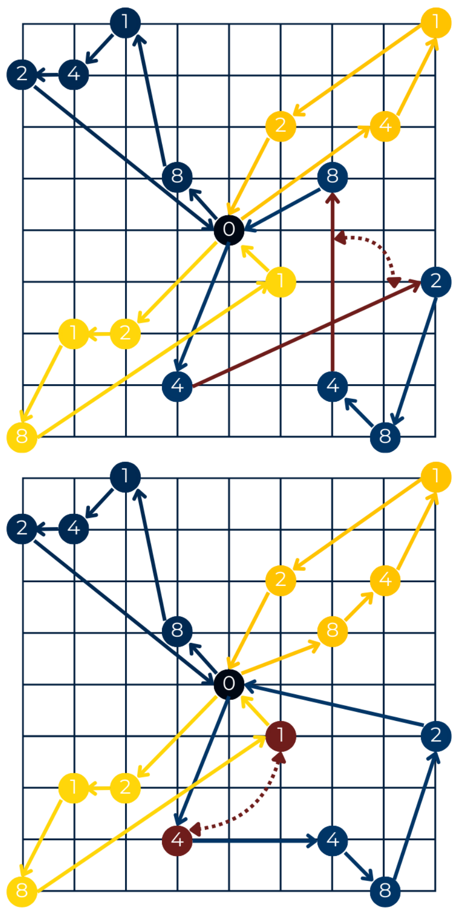


Figure 2.6. Illustration of local search moves applied to a vehicle routing solution. After a small change is applied, the affected tours are re-evaluated. If the resulting configuration improves the objective (e.g., by reducing total travel distance), the new routes are accepted; otherwise, the original solution is retained. Top left: 2-opt move on a route. Top right: relocate a customer between routes. Bottom left: swap customers between routes. Bottom right: resulting improved solution.

Source: own work (adapted from Google OR-Tools [6]).

Finally, many practical approaches are based on hybrid metaheuristics that combine features from multiple strategies. For example, a genetic algorithm might include a tabu-based local search as a refinement step, or a GRASP method may be used to initialise an ACO algorithm. These hybrids aim to balance intensification (searching near good solutions) and diversification (exploring new areas of the space) and are often among the most effective approaches in benchmark studies [10].

2.3. Integrated Routing and Packing

Although the VRP has been studied for over sixty years, it continues to be extended in new directions. Many of these extensions introduce additional constraints or objectives, aiming to make the model more practical and applicable to modern logistics settings.

One of the most natural ways to extend the VRP is to add a capacity constraint that limits the demand served by each vehicle, resulting in the Capacitated Vehicle Routing Problem (CVRP). In the literature, the terms VRP and CVRP are often used interchangeably [14], since the classical version of the problem almost always includes a capacity constraint; without it, there is little reason to use multiple vehicles.

This capacity is typically modelled as a single value, such as maximum weight or volume. A more realistic approach is to account for the actual dimensions of deliveries, where the additional task is to find a physical arrangement of items inside vehicles using a two- or three-dimensional packing model.

An exact method that builds on this idea is proposed by Iori et al. [15], who study the VRP with two-dimensional loading constraints. In their model, the demand of each customer is given as a list of items that must be packed directly into the vehicle. The packing must respect geometric feasibility: items cannot overlap, and orientation rules or loading constraints must be satisfied. Other extensions focus on route balancing (ensuring that workload is fairly distributed between vehicles) or enforce rules on the order in which customers are visited. These additions make the model more realistic but also more difficult to solve.

2.3.1. Two-Level Vehicle Routing and Loading Problem

For distribution settings with pallets or roll-containers, the packing process is naturally two-level. Items are first packed into intermediate units (pallets), and only then are these units assigned to vehicles and routed. The literature often refers to such models as pallet-based or two-level routing problems. Following the taxonomy of Liu et al. [16], this class is commonly described as the Two-Level Vehicle Routing and Loading Problem (2L–VRLP).

In a 2L–VRLP, the link between packing and routing can be described informally as follows: (i) each customer order is decomposed into individual items, (ii) items are packed into pallets, and (iii) pallets are loaded onto vehicles whose routes must visit all customers. The modelling difficulty is that the packing plan affects vehicle demands and feasibility, while routing choices determine which pallets travel together and, therefore, which loading constraints apply.

A number of influential studies fall into this two-level perspective. Zachariadis et al. introduce pallet-based routing models in which customer boxes are first packed onto pallets, and pallets are then assigned and loaded into vehicles under routing constraints [17]. In parallel, Moura and co-authors study integrated packing and routing in industrial contexts, combining detailed loading rules with routing side constraints such as time windows or pickup-and-delivery requirements [18, 19, 20].

2.3.2. Solution approaches for integrated packing and routing problems

Solution approaches for integrated routing and packing combine ideas from both the BPP and the VRP. As in the previous sections, it is useful to distinguish between exact methods, which aim for provable optimality, and heuristic or matheuristic methods, which trade guarantees for scalability. In the integrated setting, however, the search space is shaped not only by routing decisions but also by the structure of the packing, and these two components are strongly coupled.

Exact integrated formulations are typically based on mixed-integer programming, and they can provide reference solutions for small instances. In practice, their role is often indirect: a detailed model is solved on restricted neighbourhoods or subproblems, while the global search is guided by heuristics or decomposition [18, 20]. This reflects a common theme in integrated problems: modelling power is valuable, but it must be applied selectively to remain computationally feasible.

A common strategy is to treat routing as the main search object and to embed packing as a fast feasibility (and cost) evaluation. Moura and Oliveira propose an influential example by integrating a VRP with time windows and three-dimensional loading constraints [18]. In their sequential variants, routes are constructed (for example, using GRASP-style procedures), and each candidate route is checked by a dedicated loading heuristic. If the loading step fails, the method either repairs the route or adjusts the packing. Their hierarchical variants couple the two layers more tightly by letting loading quality influence route evaluation more directly.

A second line of work by Moura and co-authors moves closer to full integration by using mixed-integer formulations as optimisation engines inside a larger heuristic [19, 20]. The core idea is to alternate between heuristic route construction or improvement and solving restricted versions of an integrated MIP to refine parts of the current solution. This approach captures detailed loading rules when needed, while keeping the overall computation manageable by limiting where the exact model is applied.

Finally, Liu et al. propose a dedicated two-level framework and study constructive heuristics for the 2L–VRLP [16]. Their heuristics are organised into volume-driven methods, which prioritise space utilisation using bin-packing-inspired rules, and destination-driven methods, which attempt to form pallet groups that anticipate the routing structure. Both families are typically followed by local search to refine the solution. From a modelling perspective, this distinction is useful: it highlights two competing ways of linking packing to routing, either by focussing on utilisation first or by incorporating destination structure already during palletisation.

Overall, the integrated literature suggests a practical trade-off. Modelling packing and routing simultaneously increases realism, but it also expands the search space considerably. This motivates the range of formulations developed later in this thesis: from a baseline model that embeds palletisation into a classical CVRP, through a split-delivery variant that removes the single-visit assumption, to a sequential grouped heuristic included for comparison.

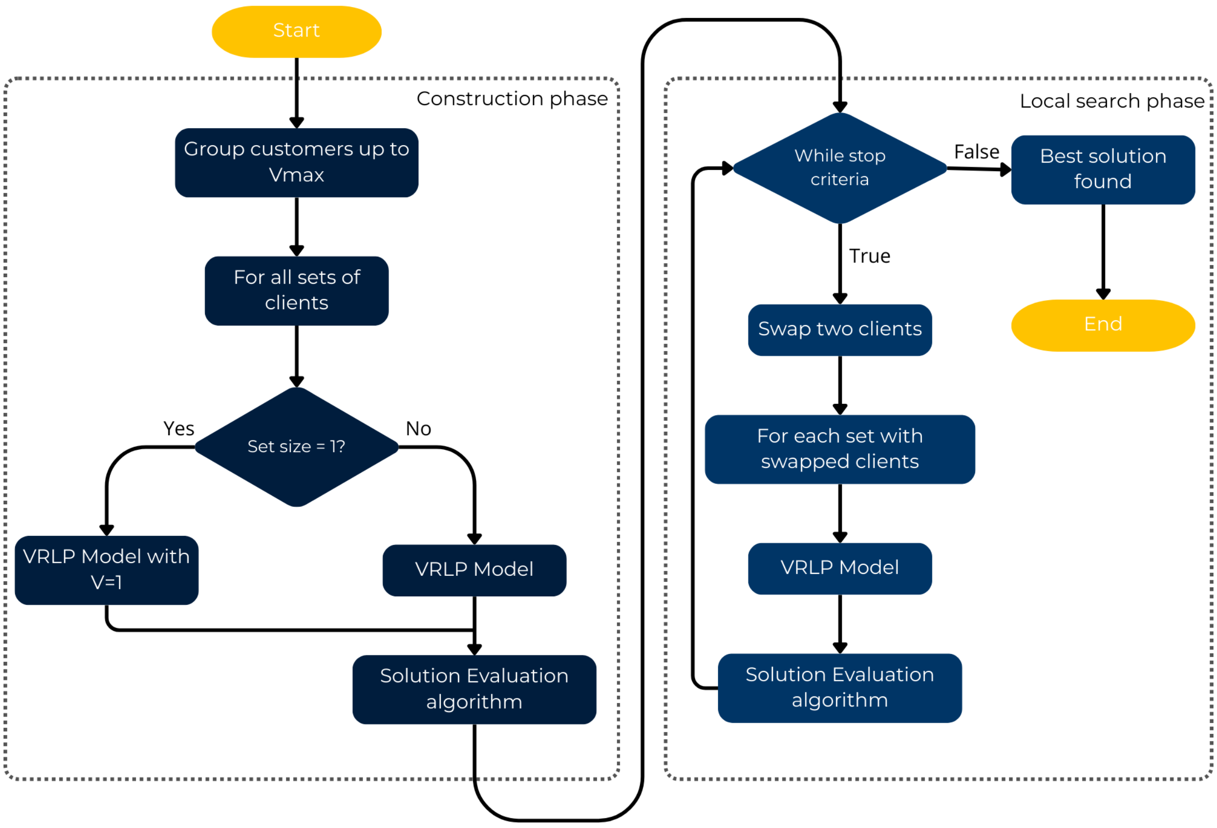


Figure 2.7. Figure illustrates the model-based heuristic proposed by Moura (2018) [19], structured into a construction phase and a local search phase. In the construction phase, customers are grouped into sets up to the maximum vehicle capacity $V_{\max}$. For each set, a VRLP model is solved (with the special case $V = 1$ when the set contains a single customer), and the resulting solution is evaluated. In the local search phase, two customers are swapped between sets, the VRLP model is re-solved for the affected sets, and the solution is re-evaluated iteratively until the best solution is found.

Source: own work (adapted from Moura (2018) [19]).

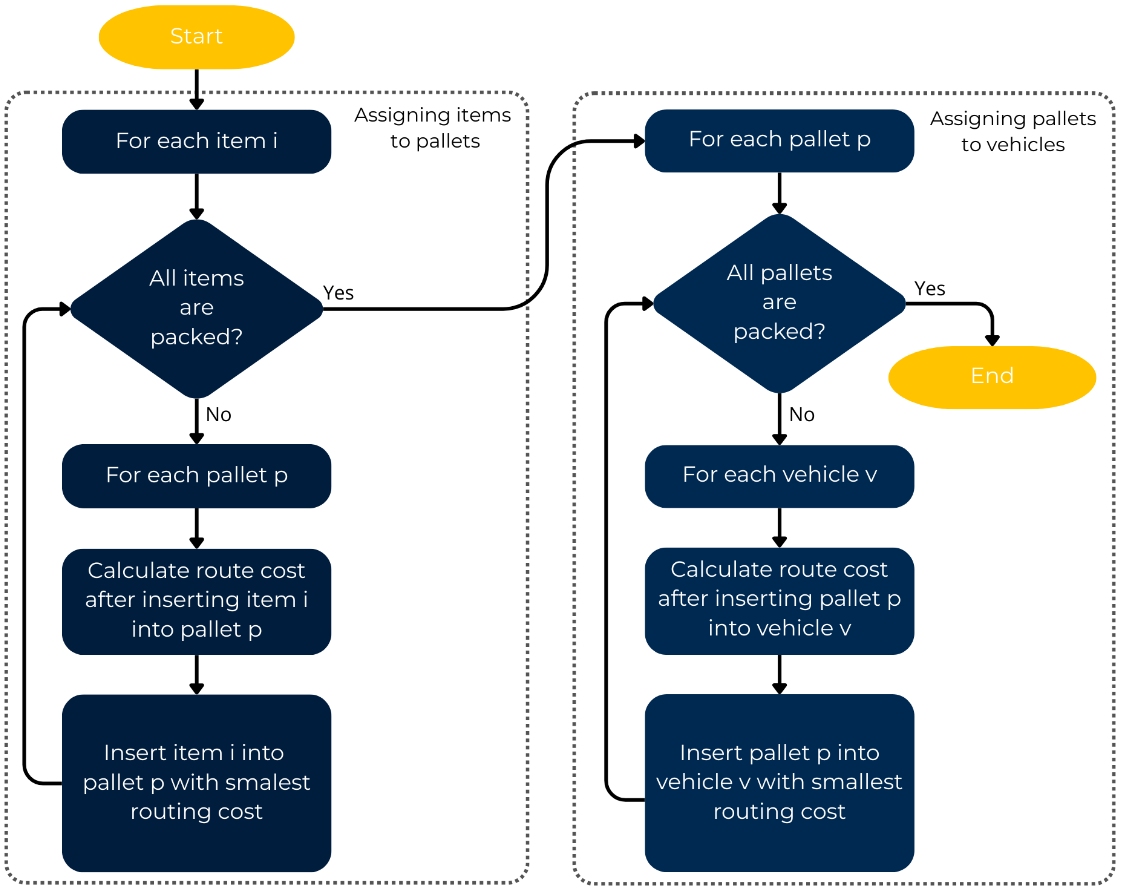

Start
Assigning items to pallets
For each item i
All items are packed?
Yes
No
For each pallet p
Calculate route cost after inserting item i into pallet p
Insert item i into pallet p with smalest routing cost
Assigning pallets to vehicles
For each pallet p
All pallets are packed?
Yes
End
No
For each vehicle v
Calculate route cost after inserting pallet p into vehicle v
Insert pallet p into vehicle v with smallest routing cost

Figure 2.8. Flowchart of the Adjusted Best Fit (ABF) heuristic proposed by Liu et al. The method proceeds in two stages. In the first stage, boxes are assigned to pallets by evaluating the marginal routing cost of inserting each box into each pallet and selecting the best candidate. In the second stage, pallets are assigned to vehicles using the same adjusted-cost principle. At both levels, routes are dynamically updated as assignments are made, yielding feasible routing and packing decisions simultaneously.

Source: own work (adapted from Liu (2024) [16]).

2.4. Summary of the literature

This chapter reviewed the main research streams that lead to the Bin Packing–Capacitated Vehicle Routing Problem. The aim was not to exhaust the very large BPP and VRP literatures, but to highlight the modelling ideas and algorithmic structures that are most relevant for the formulations developed later in this thesis.

The chapter began with the Bin Packing Problem. After recalling the classical definition and its computational hardness, the review outlined representative families of approaches, including simple constructive rules, approximation ideas, and formulations that strengthen relaxations and reduce symmetry. A recurring theme was that, even for the same objective, the way a problem is represented (and how symmetry is handled) can have a major impact on tractability.

Next, the chapter considered the Vehicle Routing Problem and its capacitated form. The key takeaway is that routing quickly becomes computationally demanding as the number of customers grows, which explains why practical work relies heavily on heuristics and metaheuristics. At the same time, even heuristic approaches are typically built around a small set of shared principles: constructing a feasible starting solution and then improving it through neighbourhood-based changes and diversification mechanisms.

Finally, the chapter reviewed integrated routing and packing models, with particular attention to two-level settings where items are first packed into pallets and only then routed. The literature shows a consistent trade-off: tighter integration increases realism, but it also enlarges the decision space and often requires decomposition, sequential evaluation of feasibility, or matheuristic frameworks. These observations directly motivate the modelling path taken in the remainder of the thesis: starting from a baseline pallet-demand embedding, extending the routing layer to allow split deliveries, and introducing a sequential grouped variant as a comparison point when full integration becomes computationally expensive.

3. Problem Formulation

The aim of this chapter is to introduce the optimisation models that will be used in the remainder of the thesis. These models form the core of the work.

The formulations adopt a declarative rather than an imperative perspective. Instead of writing an algorithm that instructs *how* to solve a problem, they specify *what* the problem is–its domain, which is made up of decision variables and relations between them expressed by constraints. This approach is characteristic of constraint programming (CP). A good model does not only reflect the problem correctly; it should also improve performance. For example, eliminating symmetries (such as treating identical bins or equivalent routes as indistinguishable) prevents the solver from exploring many redundant solutions. Because of this, CP is a natural choice for the kind of combinatorial optimisation problems considered in this thesis.

To express these declarative models, the chapter uses **MiniZinc** [21], a high-level modelling language designed specifically for constraint programming. In MiniZinc, models are described in terms of parameters, decision variables, and constraints without committing to a particular underlying solver. It provides a rich library of global constraints–such as `bin_packing` for packing items into bins and `circuit` for enforcing a single Hamiltonian cycle. These global constraints come with specialised propagation algorithms, so using them often leads to stronger pruning and improved performance. The same model can be run with different solvers, and extensions or variants of the problem can often be obtained by modifying or adding constraints rather than rewriting large parts of the formulation.

Within this setting, the chapter proceeds through three formulations. It begins with a model of the BPP, which later serves as the packing layer in the integrated problem. Next, it presents a CVRP model that captures the routing of vehicles under capacity constraints. Finally, it introduces an integrated BP-CVRP model that combines these two layers: items are packed into pallets, pallets are loaded onto vehicles, and vehicles are routed to customers.

3.1. Bin Packing Problem Formulation

The BPP consists of assigning a set of items to identical bins of fixed capacity. Each item must be placed in exactly one bin; bin capacities must not be exceeded, and the objective is to minimise the number of used bins. The packing is modelled with the global constraint `bin_packing` and adds symmetry-breaking constraints (based on lexicographic ordering, introduced in Chapter 2) to reduce the search space explored by the solver.

3.1.1. Parameters

The parameters defining a BPP instance are summarised in Table 3.1. The model assumes that the input items are sorted in decreasing order of size.

Table 3.1. Parameters used in the BPP formulation.

Core input parameters	
$\texttt{n} \in \mathbb{Z}_{>0}$	Number of items to be packed.
$\texttt{capacity} \in \mathbb{Z}_{>0}$	Capacity of each bin (identical for all bins).
$\texttt{size[i]} \in \mathbb{Z}_{>0}$	Size of item i, for $i \in \texttt{I}$, where $\forall\, i, j \in \texttt{I},\ i < j \Rightarrow \texttt{size[i]} \geq \texttt{size[j]}$
Index sets	
$\texttt{I} = \{1, \ldots, \texttt{n}\}$	Index set of items.
$\texttt{B} = \{1, \ldots, \texttt{m}\}$	Index set of bins, where $\texttt{m} = \texttt{n}$ is an upper bound on the number of bins (in the worst case, each item occupies its own bin).

3.1.2. Decision variables

The decision variables used in the BPP formulation are summarised in Table 3.2. They describe the assignment of items to bins and the number of bins used in the solution.

Table 3.2. Decision variables used in the BPP formulation.

Decision variables	
$\texttt{b[i]} \in \texttt{B}$	Index of the bin to which item $i \in \texttt{I}$ is assigned.
$\texttt{x[j,i]} \in \{0, 1\}$	Binary variable equal to 1 if item $i \in \texttt{I}$ is assigned to bin $j \in \texttt{B}$, and 0 otherwise.
$\texttt{nBins} \in \{0, \ldots, \texttt{m}\}$	Number of bins used in the solution (with $\texttt{m}$ denoting the upper bound on the number of bins).

The arrays `b` and `x` offer two complementary representations of the same packing: `b[i]` stores the index of the bin chosen for item i while `x[j,i]` indicates whether item i is placed in bin j.

3.1.3. Constraints

The constraints define the feasible solutions of the BPP. They are grouped according to their modelling role.

Packing and assignment constraints

(1) `bin_packing(capacity, b, size);`

(2) $\forall i \in \texttt{I},\ \forall j \in \texttt{B}:\ \texttt{x[j,i]} = 1 \iff \texttt{b[i]} = j$

Constraints (1)–(2) can be interpreted as follows:

- Constraint (1) uses the global `bin_packing` [22] constraint to ensure that the total size of items assigned to each bin does not exceed the bin capacity.
- Constraint (2) links the item–bin assignment array `b` with the binary matrix `x`, ensuring that both representations describe the same packing decision.

Symmetry-breaking constraints

(3) $\forall j \in \{1,\ldots,\texttt{m}-1\}:\ (\texttt{x[j,1]},\ldots,\texttt{x[j,n]}) \succeq_{\text{lex}} (\texttt{x[j+1,1]},\ldots,\texttt{x[j+1,n]})$

(4) $\forall i \in \texttt{I}:\ \texttt{b[i]} \leq i$

(5) $\forall i,k \in \texttt{I}\ i < k\ \wedge\ \texttt{size[i]} = \texttt{size[k]}:\ \texttt{b[i]} \leq \texttt{b[k]}$

Constraints (3)–(5) can be interpreted as follows:

- Constraint (3) imposes a lexicographic order on the rows of the assignment matrix `x`, which reduces symmetry by enforcing an ordering on bins that are otherwise interchangeable, as illustrated in Figure 2.3.
- Constraint (4) restricts item i to be assigned only to bins with index at most i, which breaks symmetry by preventing unnecessary use of higher-indexed bins.
- Constraint (5) enforces an ordering for items of equal size by requiring that an item with a smaller index is not placed in a bin with a higher index than a later item of the same size, further reducing symmetric solutions.

3.1.4. Objective function

The objective of the BPP is to minimise the number of bins used. This is modelled by minimising the variable `nBins`, which represents the highest bin index assigned to any item:

$$\texttt{nBins} = \max_{i \in \texttt{I}} \texttt{b[i]}.$$

Under the lexicographic ordering constraint, unused (empty) bins are pushed to the end of the index range, so used bins appear without gaps. Therefore, minimising the maximum assigned bin index is equivalent to minimising the number of used bins.

3.2. Capacitated Vehicle Routing Problem Formulation

The CVRP consists of determining routes for a fixed fleet of identical vehicles serving a set of customers. Each route starts and ends at a single depot, every customer is visited exactly once, and the total delivered demand on each route must not exceed the vehicle capacity. The objective is to minimise total travel cost (distance or time). The formulation in this section follows the benchmark MiniZinc implementation from the official repository.

To model the routing decisions, the formulation uses the giant tour representation. All customer visits and depot copies are placed into one cycle–a Hamiltonian circuit–defined by a successor relation. By inserting multiple depot copies, the single cycle decomposes into individual vehicle routes: each subpath from a start-depot node to the corresponding end-depot node represents one vehicle tour.

3.2.1. Parameters

The parameters defining a CVRP instance are summarised in Table 3.3. They are grouped into core input parameters, auxiliary bounds, domain sets, and quantities derived for the giant-tour representation.

Table 3.3. Parameters used in the CVRP formulation.

Core input parameters	
$\texttt{N} \in \mathbb{Z}_{>0}$	Number of customers in the original input instance.
$\texttt{Capacity} \in \mathbb{Z}_{>0}$	Maximum capacity of each vehicle.
$\texttt{Demand[i]} \in \mathbb{Z}_{\geq 0}$	Demand associated with customer i, for $i \in \texttt{CUSTOMER}$.
$\texttt{Distance[i,j]} \in \mathbb{Z}_{\geq 0}$	Distance between nodes i and j in the original input representation.
Auxiliary bounds	
$\texttt{nbVehicles} \in \mathbb{Z}_{>0}$	Upper bound on the number of vehicles; set to N to allow one vehicle per customer in the worst case.
$\texttt{nbCustomers} \in \mathbb{Z}_{>0}$	Number of customers; in this implementation equal to N.
$\texttt{timeBudget} \in \mathbb{Z}_{\geq 0}$	Upper bound on route duration, computed from the distance matrix and used to define the time domain.
Domain sets	
$\texttt{VEHICLE} = \{1, \ldots, \texttt{nbVehicles}\}$	Index set of vehicles.
$\texttt{CUSTOMER} = \{1, \ldots, \texttt{nbCustomers}\}$	Index set of customers.
$\texttt{LOAD} = \{0, \ldots, \texttt{Capacity}\}$	Domain of cumulative vehicle load.
$\texttt{TIME} = \{0, \ldots, \texttt{timeBudget}\}$	Domain of arrival-time variables.

Giant-tour representation	
$\texttt{NODES} = \{1,\ldots,\texttt{nbCustomers} + 2\cdot\texttt{nbVehicles}\}$	Node set of the giant-tour representation, consisting of customer nodes and start/end depot nodes for each vehicle.
`DEPOT_NODES`	Set of all depot nodes in the giant-tour representation.
`START_DEPOT_NODES`	Subset of depot nodes representing route starts.
`END_DEPOT_NODES`	Subset of depot nodes representing route ends.
$\texttt{demand[i]} \in \mathbb{Z}_{\geq 0}$	Demand in the giant-tour representation; equal to `Demand[i]` for customer nodes and 0 for depot nodes.
$\texttt{distance[i,j]} \in \mathbb{Z}_{\geq 0}$	Distance matrix adapted to the giant-tour representation, combining customer–customer, customer–depot, and depot–depot distances.

3.2.2. Decision variables

The decision variables used in the CVRP formulation are summarised in Table 3.4. Together, they describe the routing structure of the giant tour.

Table 3.4. Decision variables used in the CVRP formulation.

Decision variables	
$\texttt{successor[i]} \in \texttt{NODES}$	Node visited immediately after node $i \in \texttt{NODES}$ in the giant tour, defining the forward routing structure.
$\texttt{predecessor[i]} \in \texttt{NODES}$	Node visited immediately before node $i \in \texttt{NODES}$ in the giant tour; this variable is redundant, as it can be derived from the successor relation, but may simplify constraint definitions and improve propagation.
$\texttt{vehicle[i]} \in \texttt{VEHICLE}$	Index of the vehicle assigned to visit node $i \in \texttt{NODES}$.
$\texttt{load[i]} \in \texttt{LOAD}$	Cumulative load carried by the assigned vehicle upon arrival at node i, used to enforce vehicle-capacity constraints along the tour.
$\texttt{arrivalTime[i]} \in \texttt{TIME}$	Time at which the vehicle assigned to node i arrives at that node.

Together, the arrays `successor` and `predecessor` describe the Hamiltonian cycle over `NODES`, while `vehicle`, `load`, and `arrivalTime` track which vehicle traverses each arc and how capacity and time evolve along the tour.

3.2.3. Constraints

The constraints define how the decision variables interact to form feasible vehicle routes. They are grouped according to their role in the model, and each group is followed by a short explanation.

Throughout this section, we use N_C for `nbCustomers` and N_V for `nbVehicles`. The giant-tour node set is partitioned into customers, start-depot copies, and end-depot copies: $\mathcal{C} = \{1, \ldots, N_C\}$, $\mathcal{S} = \{N_C+1, \ldots, N_C+N_V\}, \mathcal{E} = \{N_C+N_V+1, \ldots, N_C+2N_V\}, \mathcal{N} = \mathcal{C} \cup \mathcal{S} \cup \mathcal{E}$.

Initialisation constraints

(1) $\forall n \in \{N_C+2, \ldots, N_C+N_V\}: \quad \texttt{predecessor[n]} = n + N_V - 1$

(2) $\texttt{predecessor[}N_C\texttt{+1]} = N_C + 2N_V$

(3) $\forall n \in \{N_C+N_V+1, \ldots, N_C+2N_V-1\}: \quad \texttt{successor[n]} = n - N_V + 1$

(4) $\texttt{successor[}N_C \texttt{ + } 2N_V\texttt{]} = N_C + 1$

(5) $\forall n \in \mathcal{S}: \quad \texttt{vehicle[n]} = n - N_C$

(6) $\forall n \in \mathcal{E}: \quad \texttt{vehicle[n]} = n - N_C - N_V$

(7) $\forall n \in \mathcal{S}: \quad \texttt{arrivalTime[n]} = 0$

(8) $\forall n \in \mathcal{S}: \quad \texttt{load[n]} = 0$

Constraints (1)–(8) can be interpreted as follows:

- Constraints (1)–(2) initialise the `predecessor` array for depot nodes by pairing each start-depot node with its corresponding end-depot node, and by linking the first start-depot node to the last end-depot node to close the cycle.
- Constraints (3)–(4) perform the analogous initialisation for the `successor` array, ensuring that each end-depot node points to the appropriate start-depot node and that the last end-depot node points back to the first start-depot node.
- Constraints (5)–(6) associate each depot node with a vehicle index, so that every full loop from start to end depot is assigned to a unique vehicle.
- Constraint (7) fixes the departure time at all start-depot nodes to zero, meaning that all vehicles leave the depot at time 0.
- Constraint (8) sets the load at all start-depot nodes to zero. Alternatively, the load could be initialised to the maximum vehicle capacity, since loading and unloading problems are symmetric; in this formulation, the cumulative load is modelled as increasing along the route.

Successor–predecessor constraints

(9) $\forall n \in \mathcal{N}:$ `successor`$\big[$`predecessor[n]`$\big] =$ `n`

(10) $\forall n \in \mathcal{N}:$ `predecessor`$\big[$`successor[n]`$\big] =$ `n`

(11) `circuit(successor)`

(12) `circuit(predecessor)`

(13) $\forall n \in \mathcal{C}:$ `vehicle`$\big[$`predecessor[n]`$\big] =$ `vehicle[n]`

(14) $\forall n \in \mathcal{C}:$ `vehicle`$\big[$`successor[n]`$\big] =$ `vehicle[n]`

Constraints (9)–(14) can be interpreted as follows:

- Constraints (9)–(10) ensure that the `successor` and `predecessor` arrays are mutually consistent: for every node, following the predecessor and then the successor (or vice versa) returns to the original node.
- Constraints (11)–(12) use the global constraint `circuit` [23] to enforce that both `successor` and `predecessor` define a single Hamiltonian cycle over `NODES`. This guarantees that every node has exactly one predecessor and one successor and that all nodes lie on the same cycle.
- Constraints (13)–(14) propagate vehicle assignments along the tour. For every customer node, the vehicle assigned to its predecessor and successor must be the same as the vehicle assigned to the customer itself. Since depot nodes are already associated with vehicles by constraints (5)–(6), this ensures that each loop in the giant tour corresponds to exactly one vehicle.

Note that `predecessor` is redundant given `successor`; nevertheless, we keep both arrays together with inverse constraints to simplify the presentation of some constraints and to strengthen propagation in constraint programming.

Time and load propagation constraints

(15) $\forall n \in \mathcal{C}:$
`arrivalTime[n]`$+$`distance`$\big[n,$ `successor[n]`$\big] \leq$ `arrivalTime`$\big[$`successor[n]`$\big]$

(16) $\forall n \in \mathcal{S}:$
`arrivalTime[n]`$+$`distance`$\big[n,$ `successor[n]`$\big] \leq$ `arrivalTime`$\big[$`successor[n]`$\big]$

(17) $\forall n \in \mathcal{C}:$
`load[n]` $+$ `demand[n]` $=$ `load`$\big[$`successor[n]`$\big]$

(18) $\forall n \in \mathcal{S}:$
`load[n]` $=$ `load`$\big[$`successor[n]`$\big]$

Constraints (15)–(18) can be interpreted as follows:

- Constraints (15)–(16) propagate arrival times along the arcs of the tour. For each traversed arc `(n,successor[n])`, the arrival time at the successor must be at least the arrival time at node `n` plus the travel time on that arc. This enforces a minimum travel time while still allowing waiting or other delays.
- Constraint (17) updates the cumulative load when moving from a customer node to its successor by adding the demand of the customer, thereby modelling the increase in vehicle load along the route.
- Constraint (18) propagates the load from each start-depot node to its successor without adding any demand, reflecting the fact that depot nodes do not contribute to the vehicle load.

We intentionally do not propagate time or load from end-depot nodes to subsequent start-depot nodes, because each start-depot node represents an independent route and is re-initialised by constraints (7)–(8).

3.2.4. Objective function

The objective of the CVRP is to minimise the total time required to complete all vehicle routes. In this formulation, the distance matrix is interpreted as travel time or arc cost, and the variable `arrivalTime` represents the cumulative cost incurred along a route. In the giant-tour representation, the completion time of vehicle v is the arrival time at its end-depot node. Therefore, the objective is

$$\min \sum_{n \in \mathcal{E}} \texttt{arrivalTime[n]}.$$

Minimising the sum of arrival times at end-depot nodes is therefore equivalent to minimising the total route cost across all vehicles.

3.3. Bin Packing Capacitated Vehicle Routing Problem Formulation

This section introduces the Bin Packing–Capacitated Vehicle Routing Problem as the integrated problem studied in this thesis. The key link between the two layers is palletisation: each customer order is given as a list of items, these items are packed into pallets, and the resulting number of pallets becomes the demand used by the routing layer.

The section presents three related formulations. First, a baseline model embeds palletisation into a classical CVRP by deriving `Demand[c]` from packing while keeping the single-visit assumption. Next, the routing layer is extended with split deliveries to remove this restriction. Finally, a sequential grouped-pallet variant is introduced as a comparison model that reduces routing complexity by fixing part of the decision process before solving the routing subproblem.

3.3.1. Baseline Routing Model with embedded Packing Layer

This first integrated model is obtained by directly composing the BPP and the CVRP within a single MiniZinc model. The CVRP formulation from Section 3.2 is extended with additional parameters obtained from solving the BPP (Section 3.1) for each customer's order.

Contrary to the classical CVRP, the demand of each customer is not a single parameter but a calculated variable derived from each order, understood as a list of individual items with given sizes. The palletisation process, which consists of solving one BPP for each customer order, then becomes the link between those two models. Specifically, the solution from BPP is injected into `Demand[c]`, originating from VRP. Table 3.5 summarises this change.

Table 3.5. How customer demand is defined in the directly composed model.

Quantity	In CVRP	In the combined model
`Demand[c]`	Input parameter: customer demand is given in the instance.	Derived value: number of pallets used to pack each customer's items.
`ItemsPerCustomer[c]`	Not present.	Input parameter: number of items ordered by customer c.
`SizesOfItems[c,i]`	Not present.	Input parameter: size of item i belonging to customer c.
`binCapacity`	Not present.	Capacity of one bin (pallet) used in packing.
`Capacity`	Maximum capacity of each vehicle.	Capacity interpreted as the maximum number of pallets per vehicle.

The remainder of the CVRP model, including the giant-tour representation and routing constraints, remains unchanged; it simply operates on demands that are no longer fixed input data. In particular, the giant-tour representation and all routing constraints are identical to those presented in Section 3.2.

Since the routing model allows only a single visit per customer, the derived demand `Demand[c]` must not exceed the carrying capacity of a single vehicle. As `Demand[c]` represents the number of pallets obtained from packing, the model requires an upper bound on how many pallets a single customer order may generate. The worst-case packing scenario occurs when each item occupies its own bin. In practice, this means the model may reject feasible instances simply because the worst-case pallet count would not fit into one vehicle, even if the actual optimal packing would. Hence, the maximum number of items per order must be constrained to be less than or equal to the maximum number of pallets that a single vehicle can carry, which quickly becomes overly restrictive.

3.3.2. Extending the Routing Layer: VRP with Split–Deliveries

The baseline formulation above inherits a standard CVRP assumption: each customer is represented by a single aggregated demand and must be served by exactly one vehicle. To remove this restriction, the routing layer can be extended to allow split–deliveries, meaning that a single customer order may be delivered by multiple vehicles (or visited multiple times), as long as the total delivered quantity matches the customer demand. The key idea is to extend each customer node with a small number of customer-copy nodes. Each copy represents a potential visit to the same physical location. The same routing formulation from Section 3.2 remains the same, but each customer node is replaced with several optional copies and let the model decide how many of them are active. Table 3.6 summarises what changes with respect to the classical CVRP model.

Table 3.6. Summary of changes when moving from the CVRP routing layer to the split-delivery variant.

Quantity	In CVRP (Section 3.2)	In split-delivery routing
`CUSTOMER`	One node per customer; each customer is visited exactly once.	Customers are represented by `CUSTOMER_COPIES`; multiple copy nodes may be visited.
`maxVisitsPerCustomer`	Not present.	Input parameter: maximum number of copy nodes (visits) allowed per customer.
`NODES`	Customer nodes plus start/end depot nodes.	Customer-copy nodes plus start/end depot nodes; unused copies are permitted.
Demand representation	`Demand[c]` is delivered at the single customer node c.	Each visited copy node n carries `delivered[n]`; deliveries for the same customer sum to `Demand[c]`.
Routing global constraint	`circuit(successor)` enforces one Hamiltonian cycle over all nodes.	`subcircuit(successor)` allows unused copy nodes via self-loops, while depot nodes remain active.
Distance adaptation	Standard customer–customer and depot arcs.	Distances between copies of the same customer are set to 0 (same physical location).

The split-delivery variant modifies the routing layer from Section 3.2 so that a customer may be visited multiple times. This is achieved by duplicating customer nodes and introducing constraints that control which copies are active and how much is delivered at each visit.

Activation of customer copies

(S1) $\texttt{subcircuit}(\texttt{successor})$

(S2) $\forall n \in \texttt{DEPOT\_NODES}: \texttt{successor[n]} \neq n$

Constraints (S1)–(S2) have the following meaning:

- Constraint (S1) replaces the Hamiltonian-cycle requirement with `subcircuit` [24], which permits inactive customer-copy nodes by allowing self-loops $\texttt{successor[n]} = n$.
- Constraint (S2) keeps all depot nodes active, so that the routing structure still decomposes into vehicle routes separated by depot nodes.

Delivered quantity and demand satisfaction

(S3) $\forall c \in \texttt{CUSTOMER}: \sum_{\substack{n \in \texttt{CUSTOMER\_COPIES} \\ \texttt{origCustomer}(n)=c}} \texttt{delivered[n]} = \texttt{Demand[c]}$

(S4) $\forall n \in \texttt{CUSTOMER\_COPIES}: \big(\texttt{successor[n]} = n\big) \Rightarrow \texttt{delivered[n]} = 0$

(S5) $\forall n \in \texttt{CUSTOMER\_COPIES}: \big(\texttt{successor[n]} \neq n\big) \Rightarrow \texttt{delivered[n]} \geq 1$

Constraints (S3)–(S5) have the following meaning:

- Constraint (S3) enforces split deliveries by distributing the total customer demand across all copies of that customer.
- Constraints (S4)–(S5) link activity to delivery: an inactive copy delivers nothing, while an active copy must deliver a positive amount, which avoids meaningless visits.

Load propagation with split deliveries

(S6) $\forall n \in \texttt{CUSTOMER\_COPIES}: \texttt{load[n]} + \texttt{delivered[n]} = \texttt{load[successor[n]]}$

(S7) $\forall n \in \texttt{START\_DEPOT\_NODES}: \texttt{load[n]} = \texttt{load[successor[n]]}$

Constraints (S6)–(S7) have the following meaning:

- Constraint (S6) updates the cumulative load along the route by adding the delivered quantity at each active customer-copy node.
- Constraint (S7) propagates the load from each start depot without adding any demand, preserving the interpretation of `load` as the cumulative load upon arrival.

Allowing split-deliveries through customer-copy nodes makes the routing layer significantly larger. In the giant-tour representation, the number of customer nodes grows from `nbCustomers` to approximately `nbCustomers` · `maxVisitsPerCustomer`, and this expansion propagates to all routing variables and constraints (`successor`, `vehicle`, `load`, `arrivalTime`). While this extension is modelling-wise well justified, it can quickly lead to instances that are computationally demanding. This motivates a more structured solution strategy that keeps the pallet-based interpretation but reduces the size of the routing subproblem.

3.3.3. Sequential Formulation with Pallet-grouping

The split-delivery formulation introduced in the previous section removes the restrictive single-visit assumption, but it does so by significantly enlarging the routing layer. In particular, modelling multiple visits through customer-copy nodes increases the number of routing variables and constraints, which can quickly make the problem computationally challenging. The formulation presented here follows a different strategy. Instead of solving packing and routing simultaneously, the problem is decomposed into a sequential formulation consisting of a sequence of simpler steps. This model is included as a trade-off variant, positioned between the baseline integrated formulation and the full split-delivery model.

The sequential approach consists of three stages. The BPP and the CVRP formulations described earlier are reused directly; the key difference lies in how pallet demand is processed before routing.

Stage 1: Palletisation

In the first stage, each customer order is treated independently. The BPP formulation from Section 3.1 is applied to the list of items ordered by customer c, producing the number of pallets p_c required to satisfy that order. At this point, all packing decisions are fixed and are not influenced by routing considerations. As in the baseline integrated model, pallet counts replace the original notion of customer demand.

Stage 2: Grouping of pallet demand

In the second stage, pallet demands are grouped according to the vehicle capacity $Q =$ `Capacity`, expressed as the maximum number of pallets that a vehicle can carry. The pallet demands are processed using a simple grouping rule that is deliberately restrictive. If a customer requires at most one vehicle load ($p_c \leq Q$), then the entire order is assigned to a single vehicle and no split delivery is introduced. If the pallet demand exceeds the vehicle capacity ($p_c > Q$), then split deliveries are allowed, but only to the extent that they are unavoidable.

Concretely, the pallet demand of customer c is decomposed into a number of full-vehicle trips and at most one remaining partial load. Each full-vehicle trip corresponds to a direct depot–customer–depot route and is therefore converted into a fixed travel cost. Only the remaining pallets, if any, are passed to the routing solver. This rule enforces the modelling assumption that split deliveries should occur only when a single vehicle is insufficient to serve the order.

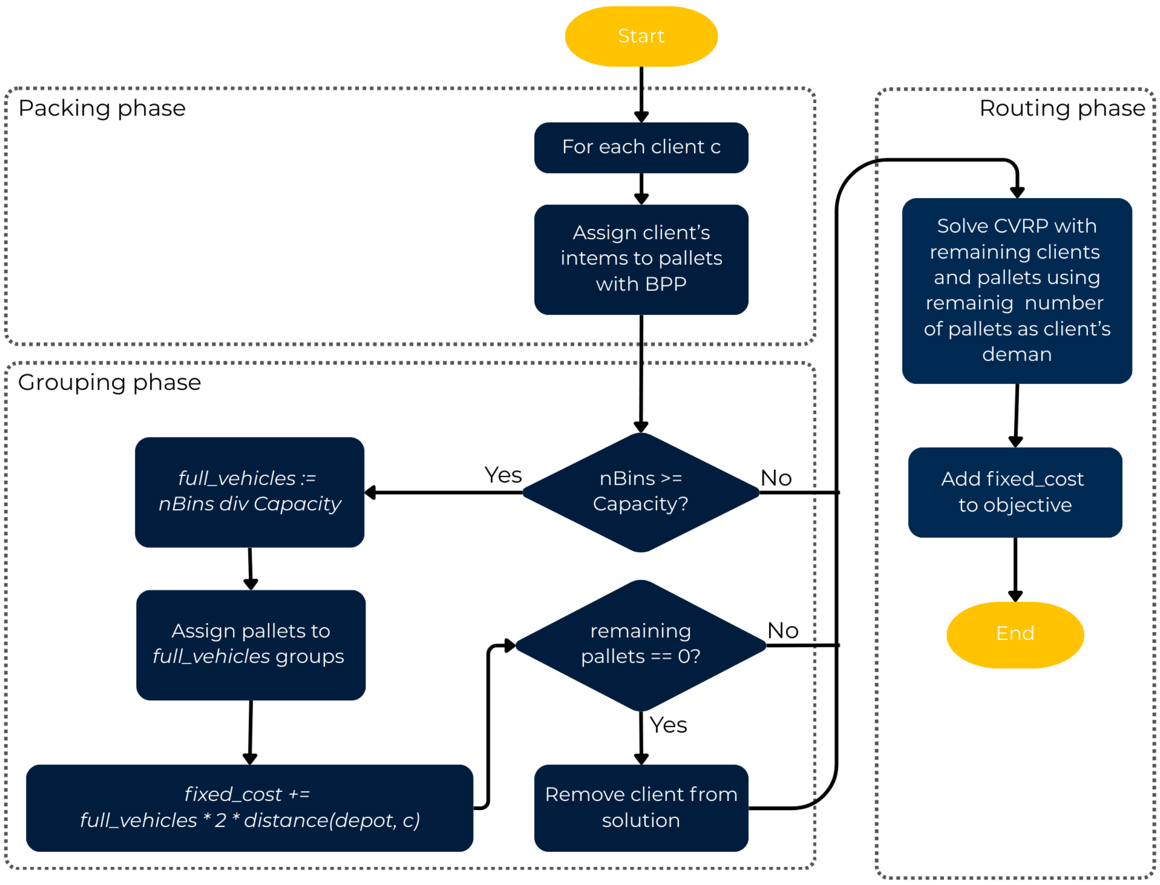

Start
Packing phase
For each client c
Assign client's intems to pallets with BPP
Routing phase
Solve CVRP with remaining clients and pallets using remainig number of pallets as client's deman
Grouping phase
full_vehicles := nBins div Capacity
Yes
nBins >= Capacity?
No
Add fixed_cost to objective
Assign pallets to full_vehicles groups
remaining pallets == 0?
No
End
Yes
fixed_cost += full_vehicles * 2 * distance(depot, c)
Remove client from solution

Figure 3.1. Sequential solution architecture. Each customer is first palletised by solving a BPP instance. Full-vehicle loads are then extracted for direct depot–customer–depot trips. Finally, a reduced CVRP is solved for the remaining pallets/customers.

Source: own work.

Stage 3: Reduced routing problem

In the final stage, routing decisions are made only for customers with a positive remaining pallet demand. Each such customer appears exactly once in the routing instance, with demand strictly smaller than the vehicle capacity Q. As a result, the routing layer is a reduced routing problem that takes the form of a classical CVRP without split deliveries or customer-copy nodes. The formulation from Section 3.2 can therefore be reused without modification, after restricting the instance to the remaining customers.

The overall objective value is obtained by combining two components: the fixed cost associated with all full-vehicle trips and the routing cost returned by the reduced CVRP model. In this way, the sequential approach preserves the pallet-based interpretation of demand while substantially reducing the size of the routing problem.

This formulation deliberately restricts the solver's freedom. Packing decisions are fixed before routing, and split deliveries are permitted only when a customer's pallet demand exceeds the capacity of a single vehicle. As a consequence, some feasible solutions of the full split-delivery model are no longer representable, particularly in cases where splitting a customer across multiple vehicles could be beneficial even when it is not strictly necessary. In return, the routing instance becomes much smaller and easier to solve. For this reason, the sequential grouped-pallet model is included as a comparison variant in this thesis, allowing an assessment of whether fully simultaneous optimisation of packing and routing is necessary in practice, or whether a carefully designed sequential approach is sufficient.

3.4. Summary of the model formulation

This chapter introduced the optimisation models that form the core of the thesis. Following a constraint programming viewpoint, the focus was placed on describing problems through decision variables and constraints, rather than designing dedicated algorithms. The models were expressed in MiniZinc, which makes it possible to state the structure of each problem in a solver-independent way while benefiting from specialised global constraints.

The chapter began with a BPP formulation, used later as the palletisation layer of the integrated problem. The model combines the global constraint `bin_packing` with symmetry-breaking constraints (including lexicographic ordering) to reduce redundant searches and improve solver performance. Next, a CVRP formulation was presented using the giant-tour representation, where routing decisions are captured by a successor relation constrained by `circuit`, and vehicle feasibility is enforced through load and time propagation constraints.

Building on these two components, the chapter then introduces a sequence of integrated BP–CVRP formulations. The baseline model directly composes packing and routing by deriving `Demand[c]` from palletisation, but it still inherits the standard CVRP assumption that each customer must be served exactly once. To remove this restriction, the routing layer was extended to allow split deliveries through optional customer-copy nodes and a `subcircuit`-based routing structure, at the cost of a substantially larger routing model. Finally, a sequential grouped-pallet variant was included as a comparison model: packing is solved first, split deliveries are permitted only when they are unavoidable, and the remaining routing decisions are handled by a reduced classical CVRP along with a fixed-cost component for full-vehicle trips.

Taken together, these formulations provide a modelling framework for studying the BP–CVRP under different assumptions and levels of integration. They also prepare the ground for the experimental part of the thesis, where the models can be compared in terms of feasibility, solution quality, and computational effort.

4. Computational Experiments and Discussion

The aim of this chapter is to evaluate the behaviour of the optimisation models introduced in the previous chapter. The experiments are used to observe how different formulations perform in practice when solved using a constraint programming approach, and how their behaviour changes as problem size increases.

The experiments are designed to highlight the impact of modelling choices, such as integrating packing and routing decisions or allowing split deliveries, on computational effort and solution quality within a fixed time limit. This allows for a clearer understanding of the balance between modelling expressiveness and tractability, which motivates the various variants discussed in the thesis.

All experiments are conducted on synthetically generated instances. For each model, instances of increasing size are solved under the same solver configuration and time limit. The results are then recorded and analysed primarily in terms of runtime, feasibility, and the quality of the best solution found.

The chapter is organised as follows. Section 4.1 describes the experimental methodology, including instance generation, solver configuration, and evaluation criteria. Section 4.2 then presents and discusses the results obtained for each model and the position of the results in the current literature.

4.1. Experimental methodology

The experimental methodology is designed to support a comparative study of the modelling choices introduced in Chapter 3. Rather than aiming for absolute performance evaluation, the experiments are structured to answer a small number of focused questions about how different formulations behave when solved under comparable conditions.

In particular, the experiments are used to explore three aspects of the proposed models. The first concerns the modelling trade-off: how computational effort changes when moving from baseline routing models to richer formulations that allow split deliveries or fully integrate packing and routing decisions. The second aspect concerns solution quality under a fixed time limit, comparing the best objective values found and the ability of each model to prove optimality within the available time. Finally, the experiments examine the role of the sequential heuristic by comparing it to the fully integrated model. This makes it possible to assess how much solution quality is lost—and how much computational effort is saved—when packing and routing are solved in a less restricted, sequential manner.

4.1.1. Models and experimental scope

The following models are evaluated in the remainder of this chapter:

– **BPP**: the standalone Bin Packing Problem formulation.

– **CVRP**: the baseline Capacitated Vehicle Routing Problem formulation.

– **SD–CVRP**: the split-delivery variant of the CVRP.

– **BP–CVRP**: the baseline integrated model in which customer demand is derived from palletisation.

– **BP–SD–CVRP**: the integrated model that combines pallet-derived demand with split deliveries.

– **Sequential BP–CVRP**: the trade-off variant in which palletisation and grouping are performed before solving a reduced routing problem.

As these models do not share a natural notion of instance size, the following values were chosen. For BPP, the size parameter is the number of items n. For routing models (CVRP, SD–CVRP), size is measured by the number of customers N. For integrated models, N remains the primary size parameter, as their difficulty additionally depends on the number of items per customer and the pallet capacity; their influence on the runtime is not as significant as that of the number of clients. Throughout the experiments, sizes are gradually increased until the solver begins to consistently reach the time limit, which is 2 hours; at that point, the experiment stops increasing size and moves to analysis.

4.1.2. Instance generation

All instances are generated synthetically to provide a controlled and reproducible testing environment. Customer locations are sampled in a bounded two-dimensional region, and a distance matrix is derived from these coordinates. For routing-only models, each customer is assigned a demand value, and vehicle capacity is chosen such that instances are non-trivial while remaining typically feasible.

For integrated models, each customer order is represented as a list of item sizes. Item sizes are generated relative to a pallet capacity `binCapacity`, using specified minimum and maximum size ratios to avoid degenerate cases. In these instances, pallet demand is not an input parameter but a derived quantity obtained by applying the BPP formulation to each customer's item list.

To reduce dependence on individual random instances, experiments are repeated across multiple random seeds for each size. Seeds are chosen deterministically, and both the generated instances and solver outputs are stored. This ensures that the full experimental study can be rerun exactly and that individual runs can be inspected when unexpected solver outcomes occur.

4.1.3. Ensuring comparable instances across models

Whenever possible, models are compared on instances that represent the same underlying input data. This is essential because many observed differences between models can otherwise be attributed to differences in instance structure rather than to modelling choices.

For routing-only comparisons (CVRP vs SD–CVRP), the same customer locations and distance matrix are used, and customer demand is sampled from the same distribution. The split-delivery model is then evaluated on the same geometric instance while relaxing the service restriction.

For the integrated comparison (BP–CVRP vs BP–SD–CVRP vs Sequential), each instance is generated once in the most expressive representation, including customer locations, distances, and per-customer item lists. This instance is then used directly by the split-delivery integrated model. The single-visit integrated model is obtained from the same underlying data by enforcing the single-visit restriction in the routing layer. In this way, any differences in feasibility, runtime, or objective values reflect the consequences of the modelling assumptions rather than differences in the input.

Finally, the sequential model is evaluated on the same integrated instances. In this case, the comparison is still instance-consistent, but the computation differs in structure: the total runtime is measured from the start to the finish of the pipeline, including palletisation, grouping, and the reduced routing solve. The reported objective value combines the fixed cost of full-vehicle trips with the routing cost returned by the reduced CVRP instance.

4.1.4. Solver configuration and run protocol

All models are solved using MiniZinc with the `cp-sat` backend from Google OR-Tools [25]. No search annotations are used; therefore, the experiments reflect the default search behaviour of the solver combined with the propagation strength of the constraints in each formulation. Experiments are executed on a single machine running Linux Mint, equipped with an AMD Ryzen 9 7900X CPU and 64 GB of DDR5 RAM. Unless stated otherwise, all runs use all 24 threads available.

For each model, instances are generated and solved for increasing problem sizes. For a fixed size, multiple instances (5 or 10, depending on problem complexity) are produced using different random seeds, and each instance is solved independently under the same solver settings. Each run is subject to a fixed time limit of 2 hours. If the solver reaches the time limit at a given size, the experiment stops increasing in size and proceeds with the collected results. This stopping rule avoids spending large amounts of time on sizes that are consistently beyond the computational budget.

For every run, the instance is saved in `.dzn` format, and the solver output is stored as a JSON record. The stored fields include the solver status, runtime, whether a solution was found, and the objective value when applicable. This makes it possible to analyse results without rerunning the solver and to inspect individual runs in cases where a model becomes infeasible or unexpectedly slow.

4.2. Computational experiments

This section presents the experimental results organised around the research questions introduced in Section 4.1. Each subsection focusses on one question and discusses the relevant observations based on runtime, feasibility, and solution quality.

4.2.1. Computational cost of richer formulations

Figure 4.1 provides an overview of how runtime changes with instance size across all models. As expected, runtime increases with N, but the growth is not uniform across formulations. The baseline CVRP remains relatively fast for small and medium sizes, while the introduction of split deliveries leads to a noticeable increase in runtime, even for small instances. The integrated models follow the same pattern: adding a packing layer does not immediately dominate the runtime, but the combination of integration and split deliveries quickly becomes computationally demanding.

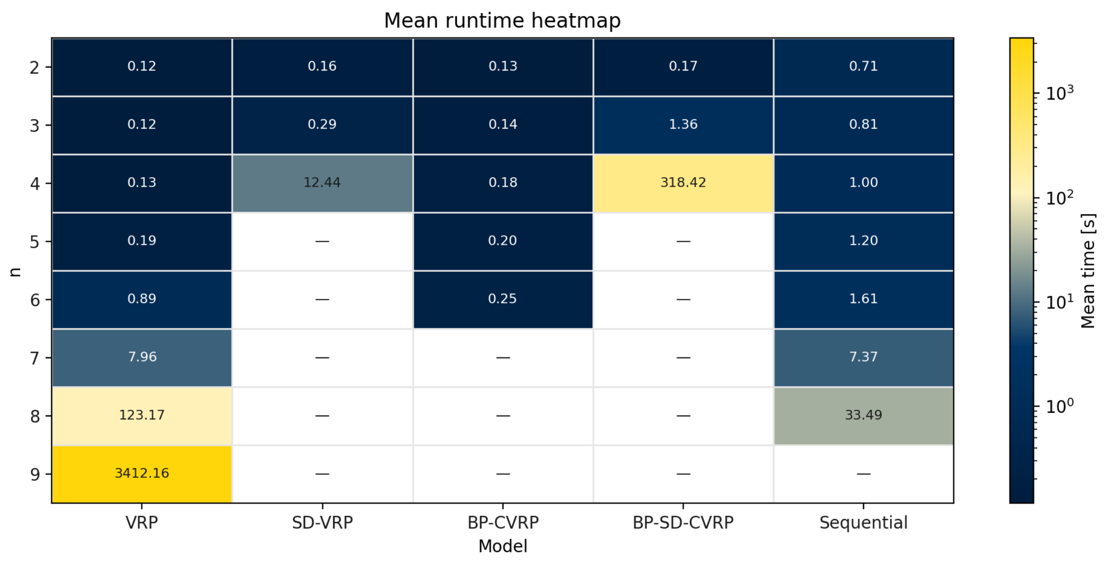


Figure 4.1. Figure illustrating heatmap of average runtime across all analysed models by number of customers (n).

This effect is illustrated more directly in Figure 4.2, which compares CVRP and SD–CVRP. The split-delivery model becomes difficult much earlier: for the tested setting, SD–CVRP reaches the two-hour time limit at substantially smaller N than the baseline CVRP–which is expected as the split-deliveries variant uses customer-copies. In other words, allowing multiple visits increases modelling flexibility, but it also expands the routing search space to the point where the solver spends most of its time exploring alternatives rather than improving the current solution.

At the same time, Figure 4.1 shows that the sequential grouped approach scales further than the fully integrated split-delivery model in terms of solvable instance size. This is consistent with its design: it fixes part of the decision-making (palletisation and the unavoidable full trips) before routing, which reduces the number of routing decisions left for the solver.

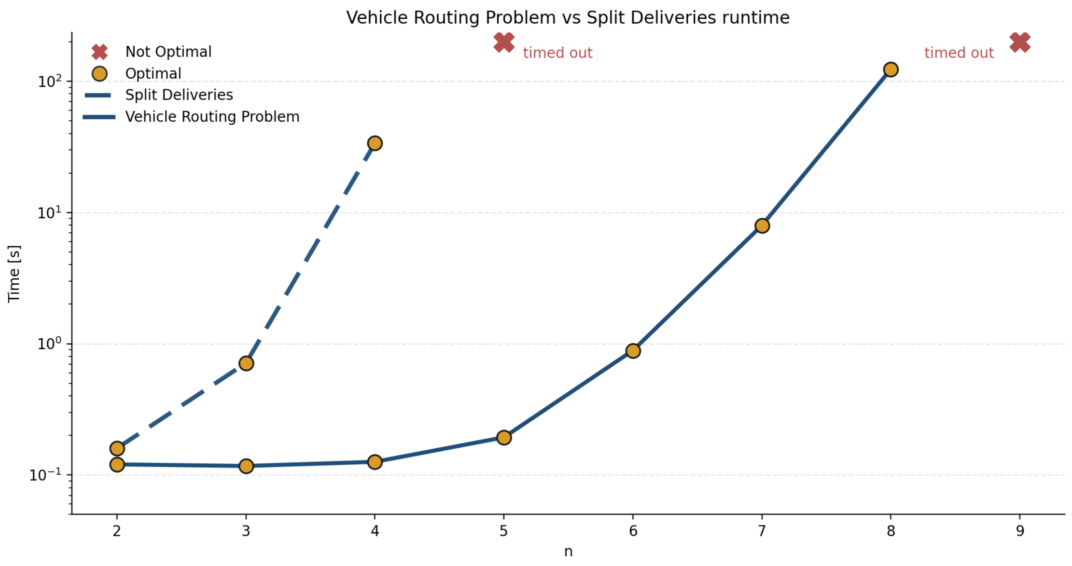


Figure 4.2. Figure illustrating plot comparing average time over customers (n) for the CVRP and SD–CVRP models.

4.2.2. Solution quality under a fixed time limit

Figure 4.3 compares the objective values obtained by CVRP and SD–CVRP on matched instances, where lower values indicate better solutions. For many instances, both models return similar objectives, which suggests that allowing split deliveries does not automatically improve solution quality.

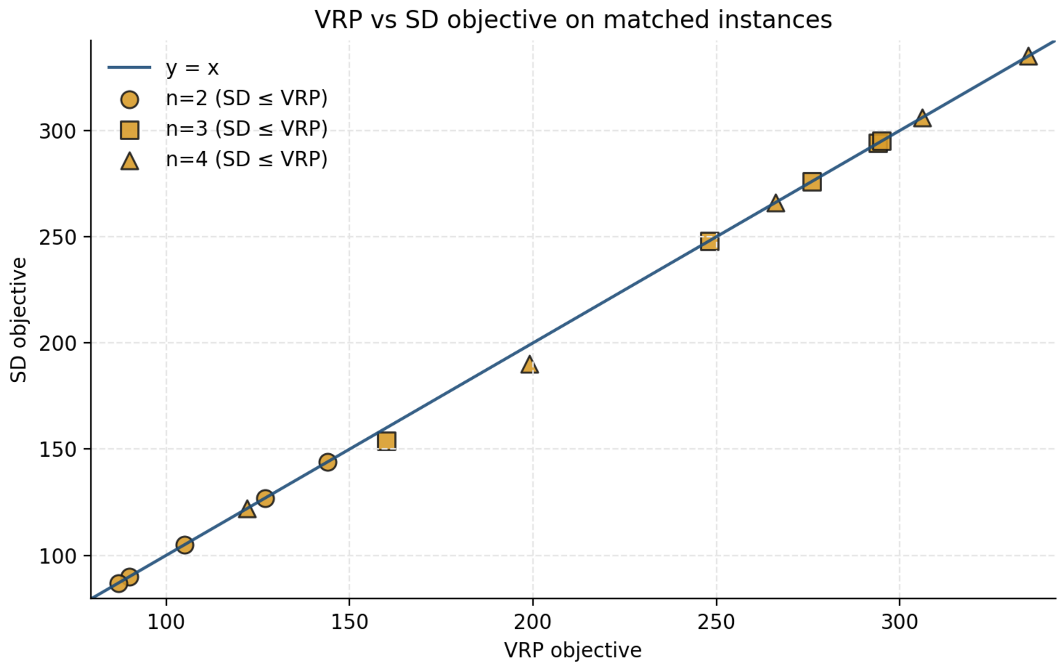


Figure 4.3. Figure illustrating solutions achieved on the same instances by CVRP and SD–CVRP models.

At the same time, the plot shows several cases where SD–CVRP achieves a slightly lower objective than the baseline CVRP. This is particularly interesting because these instances do not necessarily require splitting to remain feasible. The result indicates that the additional modelling freedom can sometimes lead the solver to better routes within the same time budget, even if the best solution ends up using no splits (or uses them only minimally). In this sense, split deliveries can improve solution quality not only by restoring feasibility but also by changing the search landscape in a way that makes good solutions easier to discover.

4.2.3. Feasibility range and the role of split deliveries

Table 4.1 highlights a key practical limitation of the baseline BP–CVRP formulation: as instance size increases, feasibility becomes the dominant issue rather than runtime. The single-visit assumption means that each customer must be served by exactly one vehicle, which implies that the derived pallet demand for that customer must fit into one vehicle load. In the pallet-based setting, this can fail even when the overall instance is "intuitively" serviceable, simply because some customer orders are too large to be delivered in a single visit. As a result, BP–CVRP becomes infeasible for a growing share of instances as N increases.

n	BP–CVRP	BP–SD–CVRP	Sequential
2	0.132	0.166	0.709
3	0.144	1.359	0.813
4	0.179	318.417	1.004
5	0.201	—	1.199
6	0.246	—	1.607
7	—	—	7.366
8	—	—	33.485

(a) Mean runtime (s) by n.

n	BP–CVRP	BP–SD–CVRP	Sequential
2	60.0	60.0	100.0
3	60.0	100.0	100.0
4	20.0	100.0	100.0
5	40.0	—	100.0
6	40.0	—	100.0
7	0.0	—	100.0
8	0.0	—	100.0

(b) Feasibility rate (%) by n.

Table 4.1. Table presenting mean runtime and feasibility rate on analogical instances for BP–CVRP, BP–SD–CVRP and Sequential models.

Importantly, this infeasibility is closely linked to how demand is modelled in the integrated setting. Customer demand is not known in advance; it is derived from the palletisation step, i.e., from solving a BPP instance for each customer's item list. If customers must be visited exactly once, then the model effectively requires an upper bound on the number of items a customer can have. The only safe worst-case bound is to assume that each item occupies its own pallet, which imposes a hard limit on the number of items per customer. This bound is overly restrictive and can render otherwise reasonable instances infeasible, even though a practical solution would simply serve the customer using multiple vehicle loads.

Allowing split deliveries addresses exactly this issue. The BP–SD–CVRP model restores feasibility in instances where the BP–CVRP fails by permitting a customer to be served in more than one visit, so that orders exceeding a single vehicle load can still be delivered. The price is computational cost: in the tested range, runtime increases sharply, and the model reaches the time limit much earlier than the baseline formulations. This confirms the trade-off already observed for CVRP versus SD–CVRP: split deliveries expand the routing layer substantially, and this becomes even more difficult when combined with an integrated packing layer.

4.2.4. Sequential versus full integration models

In designing the sequential grouped formulation, the expectation was that restricting the solver's freedom would sometimes lead to worse objective values than those of the fully integrated model. In the current experiments, this effect does not appear: whenever both models return a solution, the sequential approach achieves the same objective value as the fully integrated formulation. However, matching objectives on the tested instances does not prove that the sequential approach is generally equivalent. It only shows that, within this instance range and time budget, the additional flexibility of full integration was not exploited to produce better solutions.

A natural explanation is that the tested instances may be too small, too structured, or too constrained by the time limit for the integrated model to benefit from the fully integrated packing and routing layers. On larger or more varied instances, this could lead to improved routing decisions and, therefore, a better objective. For this reason, the sequential model is best viewed as a trade-off variant for comparison.

4.2.5. Relation to prior work and implications

The experimental results also help to position the proposed approaches relative to the literature on integrated packing-and-routing and VRP solution methodologies. In particular, using a sequential model is common in practice. For example, in a work by Moura et al., [19], a sequential solution is used; first, a construction phase builds a feasible structured solution, and then an improvement phase refines it with a local search. The same work utilised the single-client routes to simplify the routing problem—similarly to the Sequential approach used in this thesis.

A key difference is that the present work explicitly highlights the role of an initial client-grouping mechanism before the main routing optimisation. Similar ideas appear in the literature (e.g., Liu et al. [16]), where clustering or ordering-based preprocessing reduces the effective combinatorial complexity. Even a simple Sweep-style initialisation could substantially reduce the search space for later decisions and improve scalability.

Finally, the difficulty observed for fully integrated exact models is not accidental: modern VRP research relies heavily on heuristics and metaheuristics precisely because exact methods struggle to scale when routing is coupled with additional combinatorial structures (such as packing). Nevertheless, the CP models developed here remain useful: as previously mentioned, two-phase models require an initial solution that does not have to be optimal yet—CP solvers are good in finding any feasible solution, which can later be enhanced, for example, with local search.

4.3. Summary of findings

The computational study confirms that, in the BP–CVRP variants considered in this thesis, the main source of difficulty is routing rather than palletisation. In the tested instances, the bin packing component remains comparatively stable, while runtime grows quickly once routing decisions dominate the model. This is consistent with the combinatorial nature of route construction: small structural changes create many alternative solutions, and most search effort is spent exploring these alternatives.

Feasibility is a second key theme. For the baseline BP–CVRP, infeasibility appears early as a practical limitation, largely because derived pallet demand interacts poorly with the single-visit assumption. In this setting, the model must effectively guard against large per-customer pallet counts, which becomes overly restrictive. Allowing split deliveries addresses this directly by permitting large orders to be served across multiple visits, but it also expands the routing layer substantially. As a result, BP–SD–CVRP improves feasibility yet tends to be harder to solve within the same time budget.

Within these limits, increased modelling freedom does not automatically yield better solutions. In this experimental setting, whenever both the sequential grouped pipeline and the fully integrated split-delivery model returned feasible solutions, their objective values matched. This suggests that, under the chosen time limits, the integrated model often cannot exploit its extra flexibility, while the sequential approach benefits from a smaller and more structured search space.

These observations align with the literature reviewed earlier: as routing models become richer, exact approaches become difficult to scale, which helps explain the dominance of heuristics and metaheuristics in state-of-the-art VRP work. From this perspective, the sequential approach is naturally interpreted as a construction-style pipeline, similar in spirit to two-phase methods, while the results also point to a clear enhancement: introducing a principled initial grouping mechanism could reduce later complexity. Overall, the models developed here are best viewed as reusable optimisation components—the bin packing model as a reliable subroutine and the richer routing models as tools that can be applied to find baseline feasible solutions that could be refined later on.

5. Conclusions and Future Work

This thesis studied the Bin Packing–Capacitated Vehicle Routing Problem from a constraint-programming modelling perspective. The main objective was to use this integrated setting to better understand the modelling of the Bin Packing Problem and the Vehicle Routing Problem: how common formulations encode their decisions, where their computational difficulty arises, and what changes when packing and routing constraints are combined in a single model. To this end, the thesis develops a sequence of MiniZinc formulations of increasing expressiveness and compares them experimentally, focussing on feasibility, solution quality under a fixed time budget, and computational effort. This final chapter summarises the thesis, presents the main conclusions, and outlines realistic directions for future work.

5.1. Summary of the thesis

The thesis began by reviewing the classical BPP and VRP, emphasising two themes that later reappear in the experiments: the importance of representation and symmetry handling in packing models, and the observation that routing rapidly becomes computationally demanding, which motivates the widespread use of heuristics and metaheuristics in practice. The literature review then focused on integrated routing-and-packing models, especially two-level pallet-based settings (2L–VRLP), where items are first packed into pallets and only then routed.

Building on this background, Chapter 3 develops three related formulations of the integrated problem. In the baseline BP–CVRP, palletisation is embedded in a classical CVRP by deriving customer demand from per-customer bin packing while retaining the single-visit assumption. The second formulation extends the routing layer to allow split deliveries through customer-copy nodes, resulting in BP–SD–CVRP. Finally, a sequential variant decomposes the problem into palletisation, a deliberately restricted pallet-grouping step in which split deliveries are permitted only when unavoidable, producing a reduced CVRP subproblem combined with a fixed-cost component for full-vehicle trips.

Chapter 4 compared these models empirically on synthetically generated instances under a fixed solver configuration and a two-hour time limit. The purpose of these experiments was comparative: to isolate the consequences of modelling assumptions rather than to optimise performance through solver tuning.

5.2. Conclusions

The first conclusion is that, in the studied formulations, routing is the dominant source of computational difficulty. The bin packing component behaves robustly in the tested range, while runtime grows quickly as the routing layer becomes richer. This is visible already when moving from CVRP to SD–CVRP, and it becomes even more pronounced when split deliveries are combined with an integrated packing layer. From a modelling viewpoint, this is not surprising: customer copies and subcircuit-style activation enlarge the routing decision space substantially, and the solver spends most of its effort exploring alternative route structures.

A second conclusion concerns feasibility and the role of split deliveries in the pallet-based integrated setting. For the baseline BP–CVRP, feasibility becomes a practical limitation earlier than runtime. The underlying reason is structural: demand is derived from palletisation rather than given, and under the single-visit assumption, each customer must be serviceable in one vehicle load. To guarantee this, the model must rely on a conservative bound on the maximum item count per customer, which is ultimately driven by a worst-case packing scenario. As a result, instances can become infeasible in the model even when they would be serviceable in practice – even with the single-visit restriction. Allowing split deliveries addresses exactly this limitation and restores feasibility, but it does so at a significant computational cost.

The third conclusion is a pragmatic one about model integration. Within the tested instance setting and time budget, the fully integrated model did not consistently translate its additional modelling freedom into better objective values. In particular, whenever both BP–SD–CVRP and the sequential pipeline returned feasible solutions, their objective values matched in the experiments. This does not imply that full integration is redundant in general. Rather, it suggests that, under the current computational budget, the integrated model often cannot exploit its larger decision space before the time limit is reached, whereas the sequential approach benefits from a smaller and more structured routing subproblem.

Finally, these findings connect back naturally to the literature. The observed scalability limits of exact integrated formulations align with the broader VRP landscape, where heuristics and metaheuristics remain dominant precisely because enriched routing models are difficult to solve optimally at realistic sizes. At the same time, sequential approaches are common in the VRP literature; however, in those solutions, the initial phase typically clusters customers before any later improvement steps, which is not present in this work as grouping here applies only to pallets belonging to one customer.

5.3. Limitations

The conclusions above should be interpreted with the purpose of the thesis in mind. A main practical limitation is the lack of established benchmarks for the specific pallet-based setting considered here, especially when pallet demand is derived through a one-dimensional packing subproblem; as a result, the experimental framework and instances had to be constructed within the thesis. The computational study is therefore best read as an empirical complement to the modelling work, not as a definitive performance assessment. Moreover, the thesis deliberately focuses on exact constraint-programming formulations implemented in MiniZinc, whereas much of the related VRP literature relies on heuristics and metaheuristics precisely because time budgets become critical as routing models are enriched. This difficulty was also visible in the experiments: even with extended runs (e.g., a 12-hour test for SD–CVRP), solvable instance sizes remained very small, which limits the amount of data available for deeper statistical analysis. For this reason, the experimental results are used mainly to illustrate and support the central modelling conclusions of the thesis: how the BPP and VRP components are represented mathematically, where complexity enters when they are combined, and which modelling assumptions most strongly affect feasibility and tractability.

5.4. Future work

A natural next step is to strengthen the sequential pipeline with a more principled initial client-grouping mechanism. A simple cluster-first step could drastically reduce later complexity, and there is a wide range of options to explore, from classical VRP heuristics such as the Sweep algorithm to well-known clustering methods from machine learning (e.g., k-means [26]).

A second direction is to move from stand-alone solving to hybrid methods in which the CP models are used selectively as optimisation components. In particular, the MiniZinc formulations developed here are well suited both for producing an initial feasible baseline solution that could be refined later on with any local search algorithm and for acting as an exact neighbourhood solver [27] within a neighbourhood-search loop, where the method repeatedly re-optimises a small, restricted part of the current solution.

Third, there is room to explore alternative decompositions and modelling refinements that retain more of the integrated structure without fully expanding the routing layer. Examples include tighter bounds and stronger symmetry handling for customer copies, adaptive limits on the number of allowed visits, or restricted forms of integration in which packing decisions influence routing only through limited cost signals (rather than through full simultaneous optimisation).

Finally, the experimental study itself can be extended. Larger instance regimes, a broader variety of instance structures, and comparisons across different solver backends or search strategies would help to separate effects that are inherent to the modelling choices from effects that are specific to one solver configuration.

5.5. Final remarks

Overall, the thesis supports a balanced conclusion. Fully integrated pallet-based routing models are expressive and conceptually appealing, but they very quickly become computationally demanding. At the same time, developing these formulations is valuable in its own right: it makes precise, in mathematical terms, how two fundamental combinatorial problems—bin packing and vehicle routing—can be constructed. Perhaps most importantly, it suggests there could be a practical role for these models as components inside hybrid solution approaches, where exact optimisation is applied locally.

Bibliography

[1] L. V. Kantorovich. „*Mathematical Methods of Organizing and Planning Production*". In: *Management Science* 6 (July 1960), pp. 366–422. DOI: *10.1287/mnsc.6.4.366*.

[2] „*Encyclopedia of Operations Research and Management Science*". In: (2013). Ed. by Saul I. Gass and Michael C. Fu. DOI: *10.1007/978-1-4419-1153-7*.

[3] Juris Hartmanis. „*Computers and Intractability: A Guide to the Theory of NP-Completeness (Michael R. Garey and David S. Johnson)*". In: *SIAM Review* 24 (Jan. 1982), pp. 90–91. DOI: *10.1137/1024022*.

[4] Maxence Delorme, Manuel Iori, and Silvano Martello. „*Bin packing and cutting stock problems: Mathematical models and exact algorithms*". In: *European Journal of Operational Research* 255 (Nov. 2016), pp. 1–20. DOI: *10.1016/j.ejor.2016.04.030*.

[5] Chanaleä Munien and Absalom E. Ezugwu. „*Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications*". In: *Journal of Intelligent Systems* 30 (Jan. 2021), pp. 636–663. DOI: *10.1515/jisys-2020-0117*.

[6] „*Capacity Constraints | OR-Tools | Google Developers*". Google Developers, 2019.

[7] J-F Cordeau et al. „*A guide to vehicle routing heuristics*". In: *Journal of the Operational Research Society* 53 (May 2002), pp. 512–522. DOI: *10.1057/palgrave/jors/2601319*.

[8] Billy E. Gillett and Leland R. Miller. „*A Heuristic Algorithm for the Vehicle-Dispatch Problem*". In: *Operations Research* 22 (Apr. 1974), pp. 340–349. DOI: *10.1287/opre.22.2.340*.

[9] Fred Glover. „*Tabu Search: A Tutorial*". In: *Interfaces* 20 (Aug. 1990), pp. 74–94. DOI: *10.1287/inte.20.4.74*.

[10] Nacima Labadie, Christian Prins, and Caroline Prodhon. „*Metaheuristics for Vehicle Routing Problems*". In: (Feb. 2016). DOI: *10.1002/9781119136767*.

[11] M. Dorigo and L.M. Gambardella. „*Ant colony system: a cooperative learning approach to the traveling salesman problem*". In: *IEEE Transactions on Evolutionary Computation* 1 (Apr. 1997), pp. 53–66. DOI: *10.1109/4235.585892*.

[12] Fred Glover. „*Handbook of metaheuristics*". Kluwer Academic Publishers, 2003, pp. 219–249.

[13] N. Mladenović and P. Hansen. „*Variable neighborhood search*". In: *Computers Operations Research* 24 (Nov. 1997), pp. 1097–1100. DOI: *10.1016/s0305-0548(97)00031-2*.

[14] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. „*The vehicle routing problem: State of the art classification and review*". In: *Computers Industrial Engineering* 99 (Sept. 2016), pp. 300–313. DOI: *10.1016/j.cie.2015.12.007*.

[15] Manuel Iori, Juan-José Salazar-González, and Daniele Vigo. „*An Exact Approach for the Vehicle Routing Problem with Two-Dimensional Loading Constraints*". In: *Transportation Science* 41 (May 2007), pp. 253–264. DOI: *10.1287/trsc.1060.0165*.

[16] Congzheng Liu, Jing Lyu, and Ke Fang. „*Integrated packing and routing: A model and its solutions*". In: *Computers Operations Research* 172 (Aug. 2024), pp. 106790–106790. DOI: *10.1016/j.cor.2024.106790*.

[17] Emmanouil E. Zachariadis, Christos D. Tarantilis, and Chris T. Kiranoudis. „*The Pallet-Packing Vehicle Routing Problem*". In: *Transportation Science* 46 (Aug. 2012), pp. 341–358. DOI: *10.1287/trsc.1110.0373*.

[18] Ana Moura and José Fernando Oliveira. „*An integrated approach to the vehicle routing and container loading problems*". In: *OR Spectrum* 31 (Mar. 2008), pp. 775–800. DOI: *10.1007/s00291-008-0129-4*.

[19] Ana Moura. „*A model-based heuristic to the vehicle routing and loading problem*". In: *International transactions in operational research* 26 (Aug. 2018), pp. 888–907. DOI: *10.1111/itor.12586*.

[20] Ana Moura et al. „*A Matheuristic Approach to the Integration of Three-Dimensional Bin Packing Problem and Vehicle Routing Problem with Simultaneous Delivery and Pickup*". In: *Mathematics* 11 (Jan. 2023), p. 713. DOI: *10.3390/math11030713*.

[21] „*The MiniZinc Handbook — The MiniZinc Handbook 2.9.4*". Minizinc.dev, 2016.

[22] Nicolas Beldiceanu. „*Global Constraint Catalog: Cbin_packing*". Github.io, 2026.

[23] Nicolas Beldiceanu. „*Global Constraint Catalog: Ccircuit*". Github.io, 2026.

[24] „*4.2.2.11. Graph constraints — The MiniZinc Handbook 2.9.4*". Minizinc.dev, 2016.

[25] „*CP–SAT Solver*". Google Developers, 2019.

[26] scikit-learn. „*sklearn.cluster.KMeans — scikit-learn 0.21.3 documentation*". Scikit-learn.org, 2019.

[27] Jip J Dekker et al. „*Solver-Independent Large Neighbourhood Search*". In: *Lecture notes in computer science* (Jan. 2018), pp. 81–98. DOI: *10.1007/978-3-319-98334-9_6*.