A variable neighborhood search approach for solving a real-world hierarchical multi-echelon vehicle routing problem involving HCT Vehicles

Marduch Tadaros[a,b], Angelo Sifaleras[c,*], Athanasios Migdalas[a]

[a] *Quality Technology and Logistics, Lulea University of Technology, Lulea, 97187, Sweden*
[b] *Communications and Transport Systems Department of Science and Technology 60174 Norrkping Sweden*
[c] *Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Str., Thessaloniki 54636, Greece*

Abstract

This paper studies the Hierarchical Multi-Switch Multi-Echelon VRP (HMSME-VRP), a newly introduced VRP variant based on a real-world case involving High Capacity Vehicles (HCV). The problem originates from the policies of a distribution company in the Nordic countries where HCVs of up to 34.5 meters and up to 76 tons are allowed. The HMSME-VRP offer a new way to model distribution problems to cover large geographical areas without substantial costs in infrastructure. Furthermore, it adds complexity to the standard VRP and, as such, remains $\mathcal{NP}$-hard and difficult to solve to optimality. Indeed, it has been demonstrated that only very small instances can be solved to optimality by a commercial solver. Thus, in order to handle instances of real-world size, we propose two General Variable Neighborhood Search (GVNS) procedures, the second of which is adaptive, utilizing an intelligent reordering mechanism. In order to evaluate the proposed procedures, 48 benchmark instances of various sizes and characteristics are generated and made publicly available, comprising of clustered, random, and semi-clustered customers. The computational results show that both GVNS procedures outperform the exact solver. Additionally, the adaptive version outperforms the conventional version based on both average and best solutions. Furthermore, we present a statistical analysis to verify the superiority of the adaptive version.

Keywords: Variable Neighborhood Search, Adaptive search, High

*Corresponding author

1. Introduction

In today's society, transports are essential to secure the necessary products and services to maintain our standard of living. However, the transportation sector is one of the most polluting industries and one of the most significant contributors to greenhouse gas emissions. Following a decline in emissions from the transportation sector in the EU between 2019 and 2020, there was an increase in 2021, establishing it as the sector with the highest greenhouse gas emissions in the EU (European Commission, 2023). In Nordic countries, geographically characterized by long distances between major cities, the quest for efficient, cost-effective, and environmentally friendly transportation of goods necessitated the adaptation of longer vehicles. Finland, in particular, recognized this need early on and became the first country in the EU to allow longer and heavier vehicles, that is, vehicles up to 34.5 meters long and up to 76 tons are now permitted. Such transports and vehicles are referred to as High Capacity Transports (HCT) and High Capacity Vehicles (HCV) respectively. Sweden, where the length is currently limited to 25.25 meters, which is, however, longer than in most European countries, has recently proposed a change to the EU Commission to allow vehicles up to 34.5 meters in length, reflecting thus the growing recognition of the importance of longer vehicles in this region for efficient and sustainable transportation.

Compared to conventional vehicles, HCVs consume more fuel and energy and release more emissions. However, since their loading capacity is larger, fewer vehicles are needed to deliver the same amount of goods. Therefore, the cost of transportation and the emissions caused per transported unit is decreased. Previous studies have shown that substantial savings in cost and emissions can be obtained by HCVs (Leach and Savage, 2012). The use of HCVs has been estimated to reduce the total mileage of trucks in Finland by 4% only in 2017 alone (Liimatainen et al., 2020), which represents savings of approximately 126 million € and at least 0.1 Mt of CO_2 emissions.

In this paper, we study the Hierarchical Multi-Switch Multi-Echelon VRP (HMSME-VRP), a newly introduced variant of the Vehicle Routing Problem (VRP) based on a real-world case involving HCVs. The problem originates from the policies of a distribution company in the Nordic countries that has to cover large geographical areas and where the cost of establishing facilities is high. An advantage of the problem is its combined use of different

fleets of vehicles. In terms of loading capacity, larger vehicles are primarily used for longer distances, whereas smaller vehicles are only used for shorter distances and distribution routes. The HMSME-VRP adds complexity to the standard VRP and, as such, it remains $\mathcal{NP}$-hard and difficult to solve to optimality. Indeed, only very small instances have been shown to be solved to optimality by a commercial solver (Tadaros et al., 2023).

In the HMSME-VRP, vehicles carry up to three interchangeable containers called swap bodies through a non-predetermined network hierarchy, and deliveries to customers are managed either by routing directly from a central depot or through intermediate facilities called switch points. At the switch points, swap bodies can be transferred to other vehicles. The problem includes two types of vehicles; original vehicles, which originate at a central depot and can carry up to three swap bodies, and local vehicles, which originate at a switch point and can only carry one swap body. At each active switch point, an original vehicle can detach swap bodies which are then attached to a local vehicle. Each local vehicle starting at a switch point performs a route serving assigned customers and then returns to the same switch point. When an original vehicle has detached a swap body at a switch point, it either performs a customer route similar to a local vehicle or continues to another switch point to unload further swap bodies. Before returning to the central depot, the original vehicle visits all the previous switch points to re-attach previously detached swap bodies. Customers are geographically dispersed and have individual demands that must be met by one vehicle carrying only one swap body at a time. The HMSME-VRP considers fixed and variable routing costs for vehicles and swap bodies. Each vehicle, original or local, can only perform one customer route, which in turn is constrained by the vehicle and swap body capacity.

The purpose of this paper is to propose an efficient metaheuristic that can solve larger and more realistic instances. We utilize two General Variable Neighborhood Search (GVNS) procedures and employ several neighborhood structures in the local search's intensification and diversification phase. Both algorithms have been evaluated against Gurobi v9.5 on a set of small instances, and the result shows that the algorithms obtain good quality solutions in a short time. In almost all instances where Gurobi could provide bounds, the same result or better than Gurobi is found. The VRP and VRP variants have previously been solved by Variable Neighborhood Search with good results (Kalatzantonakis et al., 2020, 2023). This paper is the first attempt to solve real-life-sized instances of the HMSME-VRP using metaheuristics. Furthermore, we propose 48 realistic-size benchmark instances on which the proposed GVNS algorithms are tested and evaluated.

The size of the instances spans from 31 to 284 customers and from 2 to 29 switch points.

The remainder of this paper is organized as follows: In Section 3 the problem is described and discussed. Section 2 reviews related problems and outlines the difference between these problems and HMSME-VRP. The solution approach and the proposed GVNS algorithms are described in Section 4, followed by the computational results and statistical tests in Section 5. We conclude the article in Section 6 with conclusions and suggestions for future research.

2. Literature Review

As a variant of the VRP, the HMSME-VRP resembles and combines the features of other extensions and variants of the basic VRP. In this section, we discuss such variants and contributions related to them. We also discuss how the HMSME-VRP differs from other variants.

2.1. Truck and Trailer Routing Problem

The Truck and Trailer Routing Problem (TTRP) was formally introduced by Chao (2002) to address real-life constraints, such as limited maneuver space or inaccessible locations. In the TTRP, a limited number of trucks and trailers serve customers from a central depot. Furthermore, customers can be characterized as truck or vehicle customers. Truck customers require truck-only service, while vehicle customers can be served by a truck or a complete vehicle, where a complete vehicle is a truck with an additional trailer. Complete vehicles can detach their trailers at temporary parking locations, forming three route types: pure truck routes, pure vehicle routes, and complete vehicle routes. The goal of TTRPs is generally to minimize total travel costs while ensuring customers are served by suitable vehicles.

Chao (2002) developed a construction method and tabu search heuristic for solving the TTRP on benchmark instances. Lin et al. (2011) extended the TTRP to include time windows and employed a simulated annealing-based heuristic on 54 instances extended from the Solomon (1987) VRPTW benchmark, and Mirmohammadsadeghi and Ahmed (2015) considered stochastic demand and utilized a memetic algorithm with various procedures to enhance solutions based on modified instances.

Further TTRP extensions include the Single TTRP with Satellite Depots by Villegas et al. (2010), which considers transshipment points independent of customer locations en route. Drexl (2011) introduced the Generalized TTRP, which incorporates time windows. Additionally, Accorsi and Vigo

(2020) considered truck and vehicle customers with and without parking facilities and parking only locations.

2.2. Swap-Body Vehicle Routing Problem

The VeRoLog solver challenge in 2014 introduced a variant of the TTRP called the swap-body VRP (SB-VRP), which is a generalization of the TTRP. It applies to an asymmetric network comprising a depot, swap locations, customers, and two vehicle configurations: trucks (with one swap body) and trains (with two swap bodies). Some customers can only be served by trucks (truck customers), while others can be served by either a truck or a train (train customers). Additionally, some customers' demands exceed the capacity of a single swap-body, making it necessary to serve them with a train (mandatory train customers). All routes start and end at the depot, and vehicles leaving the depot must return with the same swap-bodies. However, swap bodies cannot change vehicles during a tour. The objective is to minimize fixed and variable vehicle costs within a specific time frame, ensuring every customer is served by an appropriate vehicle. Trains have the flexibility to visit any swap location within the network, where swap-bodies can be parked, picked up, swapped, or exchanged, each action taking a certain amount of time.

The SB-VRP has received little attention in the literature, and most of the papers derive from the VeRoLog challenge. Most works adopt the cluster-first route-second approach. Huber and Geiger (2014) combined this approach with an iterated variable neighborhood search, comparing both sequential and parallel versions. Results showed competitive performance, deviating by only 0.36% from the best-known VeRoLog solution, albeit with the constraint of a single swap location per vehicle. Miranda-Bront et al. (2017) explored various construction heuristics using a cluster-first route-second approach, employing a greedy randomized adaptive search procedure. Absi et al. (2017) introduced a relax-and-repair approach involving a parallel memetic algorithm for heterogeneous fleet VRP. Todosijević et al. (2017) presented a mixed-integer programming formulation alongside the cluster-first route-second approach. Toffolo et al. (2018) devised a stochastic local search algorithm, which outperformed previous methods and improved best-known solutions in the VeRoLog Solver Challenge for most instances.

The HMSME-VRP bears resemblance to the SB-VRP in that both problems involve the use of swap bodies and vehicles departing from a central depot, with the option to visit intermediate facilities or switch points. However, the HMSME-VRP encompasses a single customer set, only accessible by trucks carrying one swap body at a time. In contrast, the SB-VRP deals

with three distinct customer sets, each demanding specific vehicle configurations. Moreover, in the SB-VRP, swap bodies remain with their assigned vehicles throughout service. In the HMSME-VRP, two vehicle fleets are employed, and swap bodies can change vehicle assignments, resulting in a non-pre-determined hierarchical structure.

2.3. Two-Echelon Vehicle Routing Problem

In the two-echelon vehicle routing problem (2E-VRP) and its extensions, a two-level network is considered. Deliveries to customers are routed via intermediate facilities, known as satellites. The goal is to optimize primary and secondary routes while fulfilling customer demands. At the depot, a fleet of primary vehicles is stationed, while secondary vehicles originate from the satellites. If each satellite has a designated capacity, the problem becomes the 2E-CVRP, as introduced by Perboli et al. (2011). Additionally, a handling cost is incurred whenever a satellite is used for loading or unloading. Unlike the HMSME-VRP, vehicles are restricted from moving between the echelons. The first echelon, linking the depot and the satellites, allows for split deliveries, where vehicles can visit one or more satellites before returning to the depot. However, split deliveries are not permitted at the second echelon, connecting satellites to customers, meaning secondary vehicles load at satellites and then visit one or more customers before returning to the same satellite, ensuring each customer is visited just once.

The 2E-VRP shares a hierarchical structure and multiple vehicle fleets with the HMSME-VRP, but it does not allow vehicles to move between echelons in its standard version. In the 2E-VRP, the structure is predetermined, with dedicated vehicle fleets for each echelon, while the HMSME-VRP has a hierarchical but non-predefined structure.

There are, however, some examples in the 2E-VRP literature where customers can be served directly from the central depot by a first-echelon vehicle. In the context of city logistics, Anderluh et al. (2017) and Anderluh et al. (2021) address scenarios without swap bodies, involving unloading and loading operations at intermediate facilities. Deliveries in the second echelon are handled by bikes originating from a city center depot. Customers are divided into van or bike customers, each with its own predetermined vehicle fleet, where no such distinction is present in the HMSME-VRP. Furthermore, in the HMSME-VRP, when a customer is visited directly from the depot, the vehicle cannot proceed to a switch point to detach or attach any swap bodies. In contrast, Anderluh et al. (2017) and Anderluh et al. (2021) allow alternating visits to van customers and satellites. Satellites effectively serve as demand nodes for the first-echelon vehicles. Moreover,

a second-echelon vehicle can complete multiple routes in contrast to the HMSME-VRP, where one swap body corresponds to one customer route. In Anderluh et al. (2021), a third set of customers, referred to as the grey zone, is introduced. Unlike the other sets, vehicle types for this set are not predefined; the optimization process determines suitable vehicles for these customers.

Further examples include Song et al. (2017) which allows direct service from the depot by first-echelon vehicles, but a vehicle reaching a satellite cannot continue to serve customers, unlike the HMSME-VRP. For the crowd shipping problem with transshipment points, Macrina et al. (2020) only permits occasional drivers to pick up a single parcel at a transshipment point, resulting in direct deliveries in the second echelon. The field of 2E-VRP and its variants is well studied in the literature and the interested reader is referred to the survey by Cuda et al. (2015) and more recently Sluijk et al. (2022).

Even if the problems discussed share some characteristics with the HMSME-VRP, they differ in several distinct features of the problem, such as customers, vehicles, and the network. Table 1 further illustrates these differences.

Table 1: Characteristics of the HMSME-VRP and basic variants of the 2E-VRP, TTRP, and SB-VRP

Characteristics	**HMSME-VRP**	**2E-VRP**	**TTRP**	**SB-VRP**
Customers	One type of customers	One type of customers	Two types of customers	Three types of customers
Vehicles	Two types of vehicles	Two types of vehicles	One type of vehicles	One type of vehicles
	Same swap-body capacity	-	Differentiated truck/trailer capacity	Same swap-body capacity
	All vehicles can serve any customer	Only secondary vehicles can serve customers	Predetermined applicable vehicle for each customer	Predetermined applicable vehicle for each customer
Network	Non-predetermined hierarchy	Predetermined hierarchy	Non-hierarchical	Non-hierarchical
Intermediate-	Switch point	Satelite	Specific customer locations	Pre-determined swap-locations
facilities	Swap-body transfer	Loading and unloading	Park and pick-up	Park, pickup, swap and exchange

3. Problem Description

In the HMSME-VRP, in addition to a central depot, there is a non-predetermined hierarchy of intermediate facilities called switch points. Commodities are

loaded on swap-bodies which are carried by vehicles originating at the central depot. A swap-body is a carrier that can be detached from a vehicle and stand on fold-out support legs at the same height as the vehicle or be loaded on a railway carriage. Switch points can be described as enhanced parking lots and are predetermined locations with enough space for a swap body to be detached from one vehicle and attached to another without any additional equipment.

The HMSME-VRP can be defined on a directed graph $G = (N, A)$, where N represents the set of nodes and A represents the set of arcs. The set of nodes includes the central depot (O), switch points (S), and the customers (C). Thus, $N = O \cup S \cup C$. We further define $K = S \cup C$ and $W = O \cup S$. Each customer $c \in C$ has a corresponding demand d_c. A customer can only be visited by one vehicle with only one attached swap body, and each arc $(i, j) \in A$ has a corresponding cost c_{ij}.

A fleet of homogeneous vehicles in terms of capacity belongs to the central depot. These vehicles, referred to as original vehicles, can carry up to three swap bodies, and all swap bodies have the same loading capacity. At the same time, the set of switch points shares another fleet of homogeneous vehicles, local vehicles. Unlike original vehicles, local vehicles can only carry one swap body at a time. Each type of vehicle has a corresponding fixed cost if used. Therefore, the total vehicle fleet is heterogeneous. For a local vehicle to perform a customer route, an original vehicle must transfer one of the swap bodies it carries to the local vehicle at the switch-point from which it departs. All customer routes must end at their starting point, and all swap bodies must be returned to the central depot. This means that an original vehicle departing from the central depot can visit customers directly if and only if one swap body is attached; otherwise, if two or more swap bodies are attached to it, it must visit at least one switch point. An original vehicle can detach at most two swap bodies, so a second switch point can be visited from a previous switch point. Therefore, possible delivery arcs are $(o, s) \cup (s, c) \quad o \in O, s \in S, c \in C$, or $(o, s) \cup (s, s') \cup (s', c) \quad o \in O, s, s' \in S, c \in C$ and, $(o, c) \quad o \in O, c \in C$.

The aim of the problem is to design distribution routes for original vehicles originating at the central depot and for local vehicles originating from any switch point, determine the number of vehicles, original and local, and swap bodies required to satisfy demand, and determine which switch points should be used. We further make the following assumptions:

- The demand of each customer is known and deterministic
- Vehicles and swap-bodies are capacitated

- Only one trip can be performed by each vehicle
- The depot and switch points are uncapacitated
- Locations of the depot and switch points are fixed

The objective is to minimize fixed and variable routing costs for vehicles and swap bodies, which is described mathematically in Equation 1.

$$\begin{aligned} \min TC = & \sum_{i \in N} \sum_{j \in N} \sum_{v \in V_o} c_{ij} \cdot q_{ij}^{v} + \sum_{i \in K} \sum_{j \in K} \sum_{v \in V_s} \sum_{t \in T} c_{ij} \cdot z_{ij}^{vt} \\ & + \sum_{v \in V_o} f_v \cdot c_v^1 + \sum_{v \in V_s} f_v \cdot c_v^2 + \sum_{t \in T} b_t \cdot c_t^3. \end{aligned} \tag{1}$$

Where q_{ij}^{v} is a binary variable equal to 1 if an original vehicle $v \in V_o$ traverses arc $(i, j) \in A$, z_{ij}^{vt} is a binary variable equal to 1 if local a vehicle $v \in V_s$ with swap-body $t \in T$ traverses arc $(i, j) \in K$. Furthermore, f_v is a binary variable indicating if a certain vehicle $v \in V$ is used, whereas binary variable b_t indicates if swap-body $t \in T$ is used. For the complete mathematical model, the interested reader is referred to Tadaros et al. (2023). A graphical representation of a feasible solution to the HMSME-VRP is depicted in Figure 1.

Table 2: Node coordinates for graphical representation in Figure 1

D	$(5, 5)$
c_1	$(0, 0)$
c_2	$(3.5, 8.25)$
c_3	$(5, 8.25)$
c_4	$(7, 9.25)$
c_5	$(8.25, 10.25)$
c_6	$(9.5, 9.5)$
c_7	$(9, 7)$
c_8	$(10.25, 7.75)$
c_9	$(7.75, 5)$
c_{10}	$(7.75, 3)$
c_{11}	$(6.5, 3.25)$
c_{12}	$(4.5, 3)$
c_{13}	$(3, 2.75)$
c_{14}	$(0.5, 3.25)$
c_{15}	$(0.75, 2.25)$
c_{16}	$(1.75, 3)$
c_{17}	$(1.75, 6.5)$
c_{18}	$(0.75, 4.75)$
c_{19}	$(0.25, 6)$
s_1	$(3.5, 4.5)$
s_2	$(5.5, 6.75)$
s_3	$(7.5, 7.75)$

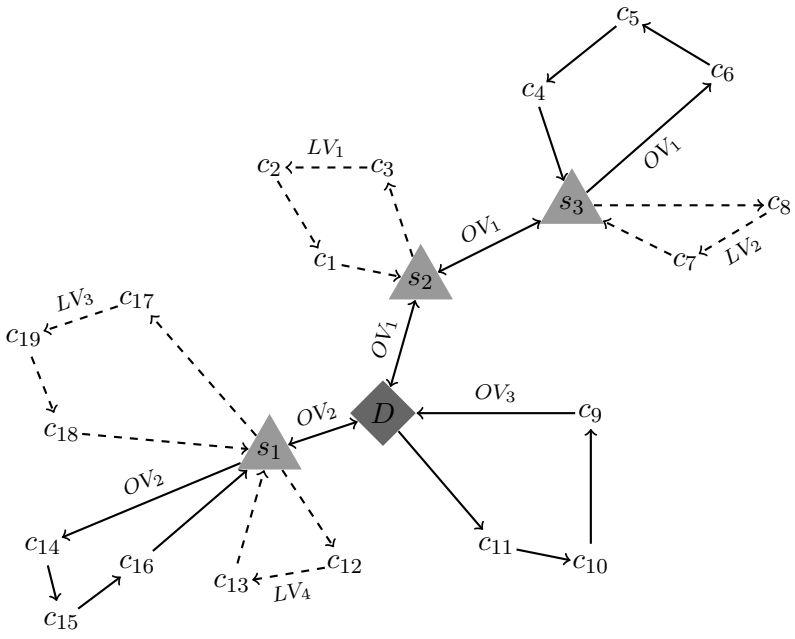

Figure 1: Graphical representation of a feasible solution for the HMSME-VRP.

The instance in Figure 1 includes 19 customers, 3 switch points and requires 3 original vehicles, 4 local vehicles, and 7 swap bodies. The original vehicle, OV_1, departs from the depot with three swap bodies; It first approaches the switch point S_2 to detach one swap body before continuing to the switch point S_3 to detach a second swap body. Once the two swap bodies have been detached, the vehicle continues with a customer tour serving customers c_6, c_5, and c_4. As soon as the swap bodies have been detached from OV_1, local vehicles attach them and perform customer tours on their own; LV_1 starting from S_2 and LV_2 starting from S_3. When OV_1 has finished its customer tour, it returns first to S_3 and then to S_2 to re-attach the swap bodies before returning to the depot. OV_2 follows a similar procedure but only visits S_1 where it detaches two swap bodies at once, which are assigned to local vehicles LV_3 and $LV4$. Finally, OV_3 serves customers directly from the depot with only one swap body attached. Assuming that the instance parameters in Eq. 1 have the values 10, 5, and 1 for c_v^1, c_v^2, and c_t^3, which corresponds to the fixed cost of using an original vehicle, local vehicles, and a swap body, respectively. Furthermore, if c_{ij} corresponds to the Euclidean distance between nodes i and j, the solution depicted in Figure 1 has an

objective value of 116.92. Whereas the total distance traveled by all vehicles equals 59.92, the use of the original vehicles equals 30, and the fixed costs of the local vehicles and swap bodies used are equal to 20 and 7, respectively, and hence 116.92 in total.

4. Proposed solution method

It is well known that the VRP is an $\mathcal{NP}$-hard problem (Toth and Vigo, 2002); therefore, as an extension of the VRP, the HMSME-VRP adds further complexity and makes it computationally difficult to solve. Previous attempts have shown that only small instances can be optimally solved using commercial solvers (Tadaros et al., 2023). Such instances included up to three switch points and 12 customers. However, slightly larger instances, including twenty customers and six switch points, could not be solved in 498,000 seconds. Therefore, it is apparent that heuristics or metaheuristics are needed to solve larger and more realistic-sized instances. To this end, in this paper, two General Variable Neighborhood Search (GVNS) procedures are developed and used to solve several instances of the HMSME-VRP.

4.1. Variable Neighborhood Search

VNS, first introduced by Mladenović and Hansen (1997), is a metaheuristic for solving global and combinatorial optimization problems and has previously been successfully applied to versions of the VRP. Guo et al. (2022) developed a VNS algorithm for the VRP with intercity ride sharing. Liu and Jiang (2022) designed an effective VNS algorithm for the 2E-VRP with simultaneous pickup and delivery demands considering constraint relaxation, expanding the search space to the infeasible region. Smiti et al. (2020) developed a Skewed-GVNS algorithm for the cumulative capacitated VRP, whereas Pei et al. (2020) applied a GVNS algorithm for the traveling repairman problem with profits and obtained new best solutions for most instances.

The VNS framework is trajectory-based and flexible for building heuristics, meaning it manages a single solution in each step. The main idea of VNS is a systematic change of neighborhoods in an intensification phase, intending to find a local optimum and a diversification phase to escape the valley of the local optimum (Hansen et al., 2019; Brimberg et al., 2023). Furthermore, it is based on the facts that a local optimum with regard to one neighborhood structure is not necessarily so far from another and that

a global optimum is also a local optimum considering all possible neighborhood structures. Lastly, local optima with respect to one or several neighborhoods are relatively close to each other.

It should also be noted that, VNS is a different metaheuristic method compared to Large Neighborhood Search (LNS). This is due to the fact that, the latter only uses one large neighborhood in its local search step (intensification phase) instead of a variable number of different neighborhoods as it is done by the VNS methodology.

The systematic search to find an optimal or near-optimal solution is repeated until a particular stopping criterion is met and includes three essential elements. These are:

- Shaking procedure, that is, the diversification phase to escape local optimum.
- Neighborhood Change Step, the systematic change of neighborhood, i.e., guiding the search of the solution space.
- Improvement Procedure, i.e., the intensification phase aiming to improve the incumbent solution.

The search can be organized in a deterministic or stochastic way or as a combination. Over the years, several versions of the VNS have been proposed and successfully applied, such as the Basic VNS (BVNS), Variable Neighborhood Descent (VND), and General VNS (GVNS). For example, the BVNS incorporates deterministic and stochastic neighborhood changes, whereas the local search procedure is deterministic, and the shaking procedure is stochastic. The local search procedure can utilize either the best or the first improvement search strategy. VND includes a local search procedure and a deterministic neighborhood change step. Apart from this intensification phase (local search), VND does not include any diversification mechanism to escape from local optima. This is the crucial difference from the (more complete) VNS method, which additionally randomly selects a point from a neighborhood. This latter part of VNS is done in order to avoid cycling, which might occur with a deterministic rule. Thus, this diversification mechanism (also known as shaking operation in the VNS terminology) makes the VNS a stochastic method, contrary to the VND variant. Several neighborhood structures can be examined to improve the incumbent solution, either sequentially or nested (Hansen et al., 2017). However, most VND variants examine the different neighborhood structures sequentially (Duarte et al., 2018) and differ in how the search proceeds when an improvement of

the incumbent solution is found in one neighborhood. The most represented versions in the literature are the Basic VND (bVND), which restarts the search from the first neighborhood, and the Pipe VND, which continues the search from the same neighborhood. In Cyclic VND, the search is resumed by examining the next neighborhood, and in Union VND, where all the predefined neighborhoods are treated as a single one, the search continues in the same large neighborhood. In addition, there are adaptive search strategies. These use one of the strategies described above, but the order of the operators changes after each iteration. Usually, the order of the operators in the subsequent iteration is determined by their performance in the previous iteration. The intensification part of the Basic VNS (BVNS) method consists of a local search in one neighborhood. The diversification/shaking part of BVNS consists of a random selection of a solution from one neighborhood. However, the neighborhoods (used both in the intensification and in the diversification phases) are sequentially changed one after the other. The General VNS (GVNS) method replaces the previous local search step of BVNS with the more powerful VND method; which is also more time consuming since each time it searches in several neighborhoods one after the other. Also, the diversification part consists of a random selection of a solution from one of all the neighborhoods, again in a random fashion. In this paper, two GVNS algorithms are developed and proposed for the HMSME-VRP. The first uses a bVND schema and is denoted $GVNS$, and the second utilizes an adaptive bVND schema and is denoted $GVNS_{adaptive}$ hereafter. The distinct difference between the two algorithms is that in the adaptive version, the order of the operators is changed for each run based on their previous performance. Section 4.6 presents a more detailed description of the algorithms.

4.2. Solution representation

To better explain a solution and the different neighborhood structures, the following notation will be used. Given a solution X, let m be the number of swap bodies used and n be the number of original vehicles used. As a swap body can only be used in one customer route, R_m corresponds to the customer route in which the swap body m serves customers. $P(OV_n)$ denotes which customer routes are assigned to the original vehicle n, while $Q(OV_n)$ is the sequence of visits to the switch points and the number of unloaded swap bodies at each switch point for the original vehicle n. Therefore, a solution of the HMSME-VRP could be represented as in Figure 2, which illustrates an instance consisting of three switch points and 11 customers.

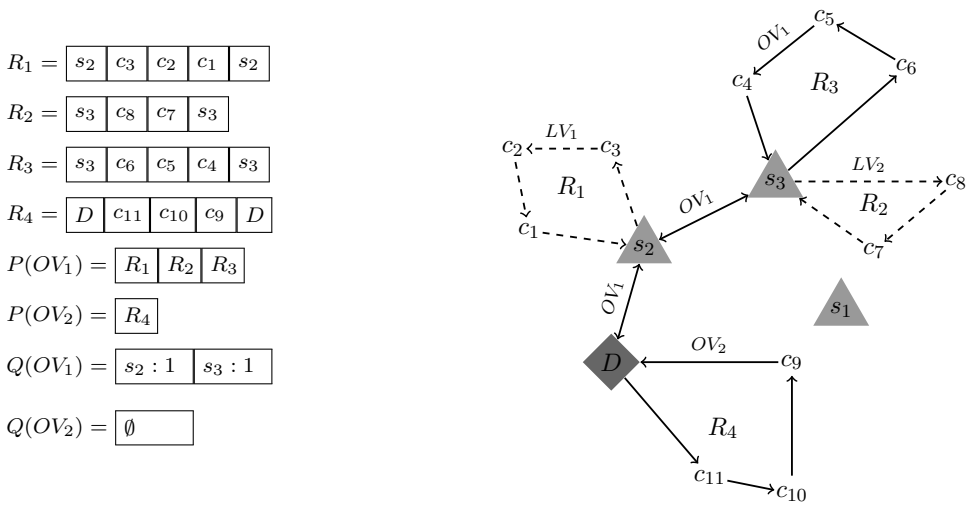

Figure 2: Example of a solution representation for the HMSME-VRP

4.3. Construction of initial solution

A two-phase construction heuristic has been developed to obtain a feasible initial solution for the HMSME-VRP. In the first phase, customer routes are formed and their origin is decided, that is, either a switch point or the central depot. In the second phase, the routes for the original vehicles are determined, that is, which switch points and in which order to visit and which customer route to distribute. Both phases utilize the nearest-neighbor heuristic.

First, a random customer is chosen, and as long as there are still unvisited customers and remaining capacity, the nearest customer to the last visited is added to the tour. When all customers are assigned to a route, each route is assigned to the nearest location $w \in W$, i.e. either to the central depot or to a switch point, which serves as each customer route's origin and end destination. The nearest location w is determined by the distances of the routes centroid and each candidate in the set W.

The number of customer routes assigned to each switch point serves as input to the second phase and is considered to be the demand for each switch point. The nearest neighbor determines the upper-level routes. If a customer route is assigned to the central depot, an original vehicle serves that customer route and only that. Otherwise, the original vehicles visit the nearest switch point until the capacity restriction is met. As the demand of a switch point may exceed the capacity of a visiting original vehicle, split deliveries are allowed, and each original vehicle is assigned one of the

customer routes originating from the last visited switch point of the upper-level route. Furthermore, if an original vehicle visits two switch points, the upper-level route is reversed and added to the end of the route as all swap bodies need to be returned to the central depot.

4.4. Intensification neighborhood structures

In this section, we describe the eight neighborhood structures used in the intensification phase to explore and improve the incumbent solution. All operators employ the best improvement strategy and only feasible moves are considered. The neighborhood structures are described in detail below.

Intra-route 2-opt (N_1). The classical 2-opt operator is applied to customer routes. It selects a customer route R_1 and two successive pairs of customers within that route (c_i, c_j) and (c_k, c_l). The pairs are split, then the first customers of the two pairs are connected, and the second customers are connected such that (c_i, c_k) and (c_j, c_l). Furthermore, the order of customers between the two selected pairs is reversed.

Inter-route Relocate (N_2). The operator selects two customers that do not belong to the same route of the customer (c_1, c_2) and (R_1, R_2). The first customer is relocated from the first route to the second and inserted into the position after the second customer, that is, c_1 is relocated from R_1 to R_2 and positioned after c_2. R_1 and R_2 can be assigned to the same switch point, but they do not have to.

Inter-route Exchange (N_3). Two customers (c_1, c_2) are selected from two different customer routes (R_1, R_2). The operator swaps the customer's route assignment. Similar to N_2, R_1 and R_2 can be assigned, but do not have to be of the same origin.

Intra-route Change Swap Location (N_4). This operator changes the origin of the customer routes. It selects an original vehicle OV_1, its upper-level route, one of the switch points visited S_1, and a location $w \in W$, which might be another switch point or the central depot. As the number of customer routes originating from a switch point within the upper-level route can range between one and three, several cases for this operator are possible:

- *One customer route R_1 originate from S_1.* R_1 originate from w instead of S_1 and the upper-level route, $Q(OV_1)$, is changed accordingly.
- *Two customer routes R_1, R_2 originate from S_1.* R_1 originate from w, while R_2 originates from S_1 or vice versa, or R_1 originates from w_1, while R_2 originates from w_2. The upper-level route, $Q(OV_1)$, is changed accordingly.

- *Three customer routes,* R_1, R_2, R_3 *originate from* $S1$. This case results in several scenarios: all customer routes origin is changed to the same w, that is, R_1, R_2, R_3 originates from w instead of S_1. Two customer routes can change their origin, while the third does not. For example, R_1 originates from w_1 and R_2 from w_2 instead of S_1, note that w_1 and w_2 might be the same location. Lastly, all three customer routes can change their origin; for example, R_1 originates from w_1, R_2 from w_2, and R_3 from w_3 instead of S_1

Inter-route L1 Relocate (N_5). The operator selects a customer route R_1, assigned to an original vehicle OV_1. Furthermore, it selects a $w \in W$ that is not assigned to OV_1 and another original vehicle OV_2. The operator changes the origin of the customer route to w, and its assignment to OV_2 and $Q(OV_1)$ and $Q(OV_2)$ is changed accordingly.

Inter-route L1 Exchange (N_6). This operator considers the upper-level routes of the original vehicles and the switch points they visit. It selects two customer routes (R_1, R_2) that are not assigned to the same original vehicle (OV_1, OV_2) and swap their assignment.

Intra-route L1 Change (N_7). The operator selects an original vehicle OV_1 and changes its upper-level route, that is, it changes the sequence of visited switch points by OV_1, that is, $Q(OV_1)$.

Inter-route L1 Insert (N_8). This operator selects a customer route R_1, originating from the central depot if there is one, a switch point S_1, and an original vehicle OV_1. It changes the origin of R_1 to S_1 and inserts it into the upper-level route of OV_1.

Figures 3-10 provide illustrative examples of the local search operators N_1 to N_8.

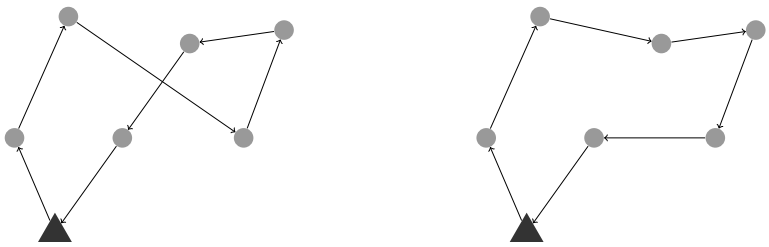

Figure 3: Intra-route 2-opt (N_1)

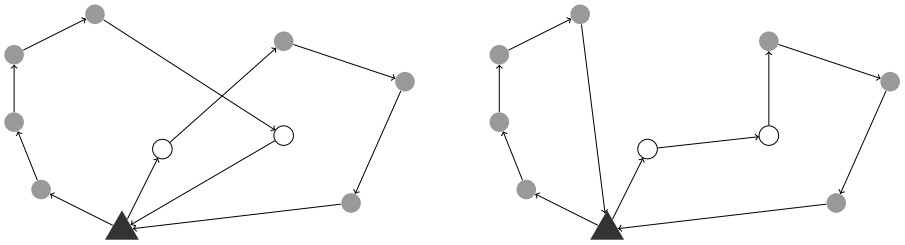

Figure 4: Inter-route Relocate (N_2)

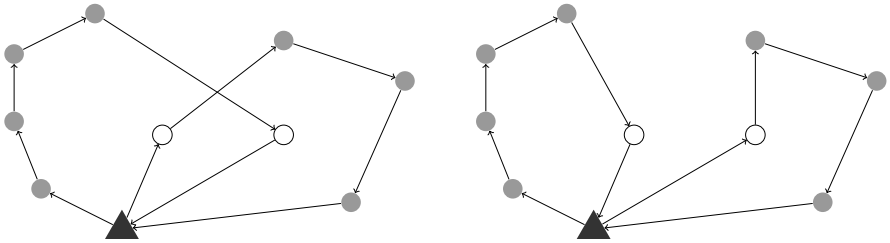

Figure 5: Inter-route Exchange (N_3)

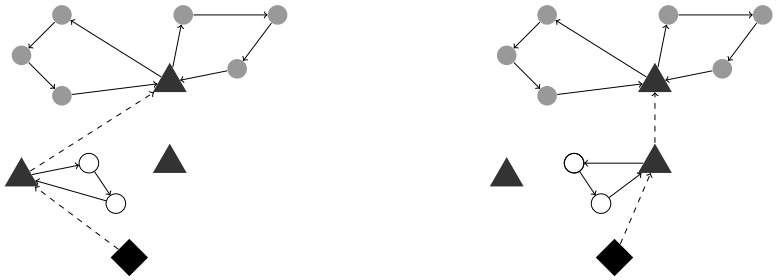

Figure 6: Intra-route Change Swap Location (N_4)

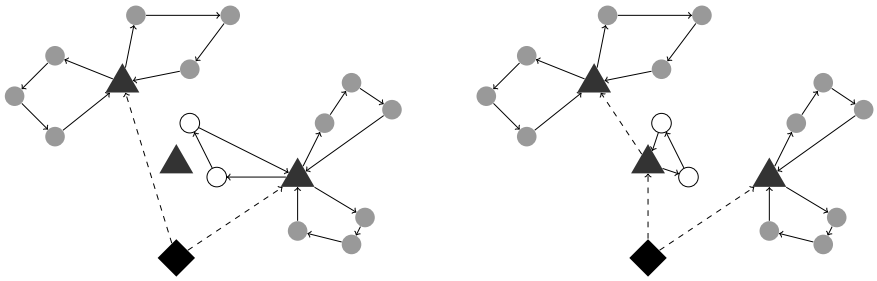

Figure 7: Inter-route L1 Relocate (N_5)

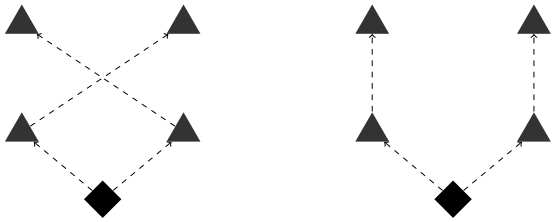

Figure 8: Inter-route L1 Exchange (N_6)

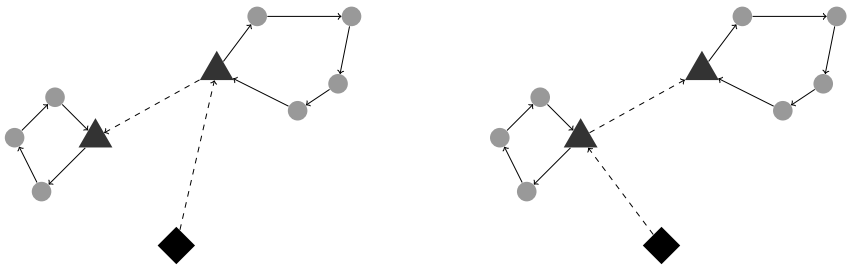

Figure 9: Intra-route L1 Change (N_7)

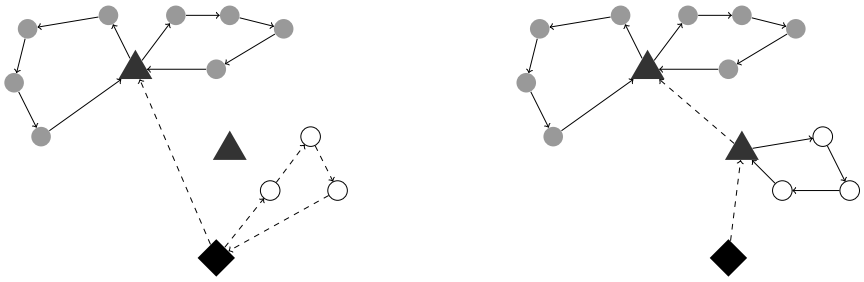

Figure 10: Inter-route L1 Insert (N_8)

Although several of the neighborhood structures resemble previously reported structures in the VRP literature, major modifications had to be made to N_4, N_5, N_6, and N_8 in order to adapt them to the HMSME-VRP. Whereas the remaining neighborhoods needed less extensive modifications.

4.5. Diversification neighborhood structures

The diversification phase consists of the shaking procedure. In each shaking, a number of random jumps are implemented based on randomly selected neighborhood structures as a way to escape the local optimum. The shaking procedure takes the incumbent solution X and the number of jumps to perform k as input. It then returns a new solution X' in which k random jumps have been performed randomly from the set of neighborhood structures. The shaking operators used in this paper are described below, while the pseudo-code for the shaking procedure is presented in Algorithm 1.

Inter-route Exchange Shake (S_1). Selects two customers at random who do not belong to the same customer route, $R_1 \neq R_2$, and swaps their route assignment. Similar to N_3 but where the change is made at random.

Inter-route Merge Shake (S_2). This operator selects two customer routes at random, R_1 and R_2, and connects the head of R_1 with the tail of R_2 and vice versa, where the head designates the first half of a route and the tail the second half.

4.6. Complete solution procedures

The components of a GVNS can be used differently to form various schemes. As mentioned above, different strategies can be used in the intensification phase of the VND. However, the bVND method is often used in the literature. It receives an incumbent solution, X, and a set of local search operators, which are applied in sequential order. Suppose that a local search operator finds an improved solution. The search then restarts from the first

Algorithm 1 Shaking procedure

```
procedure SHAKE(X, k)
    i ← 0
    while i < k do
        operator = random(S_1, S_2)
        X' ← operator(X)
        X ← X'
        i ← i + 1
    end while
end procedure
```

operator until all operators have been utilized, and no improvements have been found. The algorithmic procedure of bVND is provided in Algorithm 2.

Algorithm 2 Basic VND

```
procedure BVND(X, l_max)
    l ← 1
    while l ≤ l_max do
        X' ← N_i(X)
        if f(X') < f(X) then
            X ← X'
            l ← 1
        else
            l ← l + 1
        end if
    end while
    return X
end procedure
```

In addition to the bVND, this paper uses an adaptive version of the bVND (Karakostas et al., 2020; Karakostas and Sifaleras, 2022). That is, the order in which the operators of the intensification phase are explored is changed according to their respective success in the previous VND iteration. In each VND iteration, the success of each operator is recorded and then used in a re-ordering mechanism to re-order the operators. The re-ordering mechanism is presented in Algorithm 3, while the pseudo-code for the adaptive bVND is presented in Algorithm 4.

Algorithm 3 Re-ordering

```
procedure OPERATORS_ORDER(Opr_Order, Opr_Rec)
    if an improvement is found then
        Opr_Order ← Descending_Order(Opr_Order, Opr_rec)
    end if
    return Opr_Order
end procedure
```

Algorithm 4 Adaptive Basic VND

```
procedure Adaptive_bVND(X, l_max, Opr_Order)
    for l ← 1, l_max do
        Opr_Rec(l) ← 0
    end for
    l ← 1
    while l ≤ l_max do
        X' ← Opr_Order_l(X)
        if f(X') < f(X) then
            X ← X'
            Opr_Rec(l) ← Opr_Rec(l) + 1
            l ← 1
        else
            l ← l + 1
        end if
    end while
    return X, Opr_Rec
end procedure
```

In addition to the incumbent solution and the set of local search operators, the adaptive bVND is supplied with the order of the operators. Furthermore, it returns Opr_Rec in addition to the solution X, which is a one-dimensional array in which each local search operator is assigned a position. The array is used to store the number of times each local search operator has contributed to an improved solution and then used in the reordering procedure, Algorithm 3, to sort the operators in descending order. As the VNS continues to perform more and more iterations, the adaptive mechanism will emphasize more execution time on the most contributing operators.

Algorithm 5 describes the complete schema of the Adaptive GVNS. In the GVNS, the initial order of the local search operators corresponds to their respective numbers described above. These were determined during the pre-evaluation phase based on their cardinality, when applicable, performance, and contributions to improving the solution. It is worth mentioning that auxiliary procedures have also been developed and used to guarantee the correct objective value and the feasibility of the solution.

Algorithm 5 Adaptive General VNS

```
procedure GVNS_adaptive(X, k_max, cpu_time, l_max, Opr_Order)
    for l ← 1, l_max do
        Opr_Rec(l) ← 0
    end for
    while time < cpu_time do
        for k ← 1, k_max do
            X' ← Shake(X, k)
            X'' ← Adaptive_bVND(X', l_max, Opr_Order)
            Operators_reorder(Opr_Order, Opr_rec)
            if f(X'') < f(X) then
                X ← X''
                k ← 1
            end if
        end for
    end while
    return X
end procedure
```

The inputs for $GVNS_{adaptive}$ are an initial feasible solution, X, the shaking intensity, k_{max}, that is, the maximum number of successive applications of shaking operators to an incumbent solution, the maximum execution time corresponds to *cpu_time* and lastly the cardinality number of local search operators and their initial order, l_{max} and Opr_Order, respectively. In each iteration, the incumbent solution X is diversified using the shaking procedure, where the magnitude of the diversification of the new solution X' is determined by the current value k. Subsequently, the new solution X' is applied to $Adaptive\_bVND$, which returns an improved solution X'' as well as the record of the performance of local search operators Opr_rec. The re-ordering mechanism then updates the order of the local search operators before the new solution X'' is compared to the current best solution X. If a better solution is obtained, the new solution is stored as the current best, and the value k is reset to one. This is then repeated until the maximum execution time is reached. The conventional $GVNS$ algorithm follows the same procedure as $GVNS_{adaptive}$ with the same input parameters except for the vector Opr_rec, and instead of applying $Adaptive\_bVND$ the bVND schema is utilized. The pseudo-code for the conventional $GVNS$ algorithm is presented in Algorithm 6.

Algorithm 6 General VNS

```
procedure GVNS(X, k_max, cpu_time, l_max)
    while time < cpu_time do
        for k ← 1, k_max do
            X' ← Shake(X, k)
            X'' ← bVND(X', l_max)
            if f(X'') < f(X) then
                X ← X''
                k ← 1
            end if
        end for
    end while
    return X
end procedure
```

5. Computational results

This section presents and discusses the results of our computational experiments. The proposed solution algorithms presented in Section 4.6 and the complete mathematical model presented in Tadaros et al. (2023) were coded in Python 3.8. To solve the mathematical model the Gurobi Optimization solver v9.5.0 was used and all experiments were conducted using a 2.3 GHz Quad-Core Intel Core i7 processor with 32 GB of RAM on a macOS.

First, the generated instances in this paper are presented in Section 5.1. Section 5.2 the procedure of tuning the parameters for the presented algorithms is described and discussed. In Section 5.3, the performance of the model and the proposed algorithms are evaluated and analyzed as well as the contribution of the local search operators. Lastly, in Section 5.4 statistical tests are presented for comparing the solutions of the proposed algorithms.

5.1. Problem instances

As this is a new problem, the only existing benchmark instances are the 12 small instances presented in Tadaros et al. (2023). Therefore, in this paper, 48 new benchmark instances of realistic size have been generated. The generated instances are divided into three subsets based on how the customers have been located: semi-clustered, random, and clustered. As previously mentioned, the HMSME-VRP originates from the policies of a Nordic distribution company. The procedure to generate the semi-clustered instances is developed by trying to mimic the behavior and the distribution of customers in a real-life setting from which the problem originates. We refer to this subset as semi-clustered instances since customers have been sited based on clustering and randomly by a uniform distribution resulting in densely populated areas and some randomization and sparsely located

customers. The second subset includes only randomly placed customers, while the customers of the third subset have been placed solely by clustering. Each problem instance has a name of the form X-nY-sZ where X denotes which subset the instance belongs to semi-clustered (SC), random (R), or clustered (C). Y denotes the total number of nodes, and Z is the number of switch points. A more detailed description of the instance generation is provided in the online supplementary material, and the complete set of generated instances is available at `https://github.com/ahotad/HMSME-VRP-Instances.git`.

5.2. Parameter settings

In any VNS schema, the shaking intensity parameter, k_{max}, is critical to the efficiency of the solution method. If the value is set too small, there is a risk of getting stuck in local optimum. On the other hand, if set to large, the procedure is similar to a random walk. For each developed GVNS method, there might be different most suitable values, and in each case that value is empirically determined through computational experiments. However, it should be emphasized that different values of k_{max} do not necessarily lead to considerable variations in the solutions obtained, but it is nevertheless crucial to examine the most suitable value for each problem and algorithm. Therefore, we estimate this parameter for each algorithm by examining three different k_{max} values using the semi-clustered instance set. The values examined are $k_{max} = 10$, 15, and 20, respectively. In Table 3, the average solution of ten runs for each problem instance, algorithm, and the corresponding k_{max} value is reported, as well as the maximum CPU time in seconds for each run.

Table 3: k_{max} evaluation of $GVNS$ and $GVNS_{Adaptive}$ on the semi-clustered instance set

		$GVNS_{Adaptive}$			$GVNS$		
Instance	Cpu-time	$k_{max} = 10$	$k_{max} = 15$	$k_{max} = 20$	$k_{max} = 10$	$k_{max} = 15$	$k_{max} = 20$
SC-n34-s2	60	9526.8	9395.9	9524.6	9524.1	9564.1	9558.1
SC-n40-s2	60	6658.3	6524.7	6638.5	6660	6646.5	6729.2
SC-n40-s3	60	7549.9	7500.5	7531	7544.5	7579.5	7758.5
SC-n46-s5	60	5551.2	5530	5667.8	5535.8	5558.4	5567.8
SC-n66-s7	60	13422.2	13736.1	14000	13914.5	13840.5	13837.8
SC-n71-s6	60	12861.4	12743.2	12934.5	12875.2	12981.2	12546.9
SC-n77-s6	60	13268.3	13403.2	13658.6	13561.1	13647	13555.1
SC-n93-s7	60	17104.9	17437.1	17247.9	17344.4	17250.9	17535.5
SC-n153-s15	180	15122	15043.8	14986.5	15252.3	15033.1	15043
SC-n194-s16	180	17250.6	17382.4	17434.6	17281.4	17365.2	17683.3
SC-n197-s12	180	16308	16608.7	16608.7	16595.4	16309.9	16107.1
SC-n199-s18	180	18661.8	18448.4	18439.7	18538	18559.8	18458.3
SC-n239-s27	360	16657.9	16574.6	16246.6	16421.4	16613.4	16314.1
SC-n240-s19	360	16870.9	16589.3	16588.1	16961.6	16676.9	17300.6
SC-n274-s15	360	19891.6	19955.1	20113.9	20066.6	19868.6	19984.8
SC-n309-s21	360	18078.5	18320.7	18366	18322.3	18337.8	18250.2
Average		**14049.02**	14074.61	14124.19	14149.91	**14114.55**	14139.39

As can be seen in Table 3, the shaking intensity corresponding to the value k_{max} equal to 10 performs slightly better than the other two for the $GVNS_{adaptive}$ algorithm. This can presumably be explained by the fact that less intense shaking allows for more VND iterations, and hence more time in the intensification phase. Although at the expense of a lower degree of exploration of the solution space than the other two options with higher k_{max} values. For the conventional $GVNS$ algorithm, the k_{max} value equal to 15 yielded the best results, which is a more balanced procedure between the exploitation and exploration of the solution space compared to the adaptive version and a k_{max} value of 10.

5.3. Computational results and analysis

To evaluate the performance and efficiency of the two algorithms, tests were conducted on the 12 instances reported in Tadaros et al. (2023). Additionally, we generated 6 more instances by the same procedure as in Tadaros et al. (2023), extending the set to include instances with 30 nodes and 5 switch points. The results obtained by Gurobi correspond to a maximum time limit of one hour for instances up to 23 nodes, while for instances including 27 nodes and more, the time limit was set to two hours. For the two GVNS algorithms, the CPU time was set to 60 seconds and the parameters were set to the best reported in Table 3, i.e. a k_{max} value of 10 and 15 for the $GVNS_{adaptive}$ and the conventional $GVNS$, respectively. The results obtained by Gurobi and the two algorithms are reported in Table 4. Average results of the GVNS algorithms correspond to the average solution obtained by ten runs, while the best corresponds to the best solution found of those ten runs. Instances which gurobi could solve to optimality are marked with an $*$.

Table 4: Comparison of achieved results by $GVNS$, $GVNS_{Adaptive}$ and Gurobi

	Gurobi		Adaptive		Conventional	
Instance	**LB**	**UB**	**Best**	**Average**	**Best**	**Average**
U-n13-s3∗	59	59	**59**	59	**59**	59
L-n13-s3∗	59	59	**59**	59	**59**	59
H-n13-s3∗	73	**73**	77	77.1	77	77
U-n16-s3∗	80	80	**80**	80	**80**	80
L-n16-s3	81	95	**95**	95	**95**	96
H-n16-s3∗	84	84	**84**	91.6	92	92.9
U-n20-s4	94	105	**105**	105	**105**	105.4
L-n20-s4	92	107	**105**	109.1	**105**	111.4
H-n20-s4	98	117	**116**	120.4	**116**	124.8
U-n23-s4	112	129	**125**	125	**125**	126.2
L-n23-s4	115	144	**142**	147.9	**143**	147.3
H-n23-s4	103	143	**142**	148.5	**142**	145.4
U-n27-s5	122	138	**138**	138.8	**138**	139.2
L-n27-s5	114	148	**146**	149.8	**144**	152.8
H-n27-s5	112	155	**154**	159.4	**151**	158.8
U-n30-s5	-	-	154	154	154	154.4
L-n30-s5	-	-	165	169.9	165	170.8
H-n30-s5	-	-	171	182.6	174	187.5

The results show that both VNS algorithms are efficient and produce competitive results compared to the Gurobi solver. For the 15 instances in which Gurobi could provide bounds within the time limit, the conventional version found a better result in 7 instances and matched the upper bound in 6 instances, while the adaptive version was able to find a better solution in 7 instances and the same as the upper bound in 7 instances.

5.3.1. Analysis of local search operators

Table 5 presents the contribution of each local search operator for the runs on the instances reported in Table 4. The rows of average improvement show the average magnitude of improvement for each operator in those cases where the operator is able to improve the incumbent solution. On the contrary, rank represents the frequency with which a specific operator has found a better solution than the incumbent. A rank of 1 indicates that the operator in question has the highest frequency of finding improved solutions.

Table 5: Contributions of local search operators

		N1	**N2**	**N3**	**N4**	**N5**	**N6**	**N7**	**N8**
Average improvement	**Adaptive**	2.09	3.51	17.32	4.29	6.19	3.41	2.96	3.47
	Conventional	2.54	5.51	14.46	4.06	4.95	2.70	2.77	2.59
Rank	**Adaptive**	2	4	1	3	5	7	6	8
	Conventional	1	3	2	4	5	6	7	8

As seen in Table 5, the average improvement per operator is generally

higher for the adaptive versions except for operators N_1 and N_2. Their rank in their respective version might partially explain the exception for the N_1 and N_2 as both operators have a lower rank in the conventional version and, thereby, a higher frequency of finding improved solutions. Additionally, it is noteworthy that operator N_3 stands out with the highest average improvement, significantly contributing to the overall by the inter-route operators.

It is also worth mentioning that there are dependencies among the operators, for example, N_2 and N_3, where specific changes in one operator can result in fewer alternatives in the other, and vice versa. The use of operators varies depending on the instance and its characteristics. For example, operator N_2 is seldom used in unitary instances. Another example is the N_8 operator, which can only be utilized in cases where a customer route is assigned to the central depot. However, the operator is vital in searching for the optimal solution as without it, the algorithms could get trapped in local optimum if that case occurred. This is further demonstrated by the fact that although it is the operator with the highest rank in both versions of the algorithms, its average improvement is higher than several other operators.

In order to further evaluate the contributions and the necessity of the local search operators, three different versions of each algorithm were tested. The first version, referred to as v1, includes all 8 local search operators, v2 the 4 operators with the lowest rank, and v3 the 2 lowest rank operators. The average objective value for each instance and version is reported in Table 6.

Table 6: Results obtained by different local search operator combinations

	Adaptive			**Conventional**		
Instance	**v1**	**v2**	**v3**	**v1**	**v2**	**v3**
U-n13-s3	59	59.8	63.8	59	59.8	65
L-n13-s3	59	61.4	65	59	64.6	66.2
H-n13-s3	77.1	77	83.5	77	77	85.5
U-n16-s3	80	80	90	80	80	93.3
L-n16-s3	95	97.8	107.4	96	97.9	109.3
H-n16-s3	91.6	95.8	107.9	92.9	99.8	113.9
U-n20-s4	105	105.4	118.3	105.4	106.2	116.4
L-n20-s4	109.1	112.1	127.6	111.4	111	122.1
H-n20-s4	120.4	123.1	130.4	124.8	122.8	133.5
U-n23-s4	125	126.4	143.3	126.2	126.4	135.5
L-n23-s4	147.9	149	161.2	147.3	149.3	159.1
H-n23-s4	148.5	150.8	158.5	145.4	152.5	160
U-n27-s5	138.8	139.6	153.2	139.2	139	152.4
L-n27-s5	149.8	151.5	166.4	152.8	152.2	169.2
H-n27-s5	159.4	161.3	177.2	158.8	162.5	172.4
U-n30-s5	154	157.1	171.5	154.4	158.1	164
L-n30-s5	169.9	170	189.9	170.8	176.3	179.9
H-n30-s5	182.6	183.1	206.2	187.5	186.3	200.6

Focusing on the unitary instances, there are minor differences between the versions in the smaller instances. However, the difference between the ob-

tained objective values increases with the size of the instances. On the other hand, for the instances with low and high demand distributions, such a difference is present even in the instances of smaller size. Furthermore, a Friedman test was performed based on the different versions. The p-value for both the adaptive and conventional versions is considerably lower than 0.01, which confirms a statistically significant difference between the results obtained by the different versions and the superiority of v1, including all 8 local search operators.

5.3.2. Computational results of generated instances

For the computational analysis of the two algorithms based on the newly generated instances, we used the same computing environment and settings, but with extended CPU times. The extended CPU times correspond to 180 seconds for the small instances, 600 seconds for medium instances, and 1200 seconds for large instances. In Table 7, the best and average results are reported for all semi-clustered instances. Note that, as previously mentioned, ten runs have been executed for each such instance and algorithm, where the average result is the average of those ten runs, while the best solution corresponds to the best solution obtained by those ten runs.

Table 7: Average and best solutions for the $GVNS_{Adaptive}$ and $GVNS$ for the semi-clustered instances

		$GVNS_{Adaptive}$			$GVNS$		
Instance	cpu-time	Avg	Best	Gap	Avg	Best	Gap
SC-n34-s2	180	**9300**	9194	1.14	9377.8	9194	1.96
SC-n40-s2	180	**6423.2**	**6399**	0.38	6646	6490	2.35
SC-n40-s3	180	**7429.2**	**7273**	2.1	7497.2	7323	2.32
SC-n46-s5	180	**5426.9**	**5274**	2.82	5442.4	5348	1.73
SC-n66-s7	180	**13269**	**12688**	4.38	13587.8	12946	4.72
SC-n71-s6	180	**12301.9**	**11677**	5.08	12533.7	12196	2.69
SC-n77-s6	180	**13059.4**	12503	4.26	13128.3	**12293**	6.36
SC-n93-s7	180	**16561.8**	**15886**	4.08	17204.4	16423	4.54
SC-n153-s15	600	14772.2	**14012**	5.15	**14752.5**	14204	3.72
SC-n194-s16	600	**17257.9**	**16206**	6.1	17362	16275	6.26
SC-n197-s12	600	**15900.5**	**14902**	6.28	16211.2	15612	3.7
SC-n199-s18	600	**18137.6**	**17232**	4.99	18647.4	17809	4.5
SC-n239-s27	1200	16288.6	**15135**	7.08	**16204.2**	15440	4.72
SC-n240-s19	1200	**16776**	16129	3.86	16783.8	**15944**	5
SC-n274-s15	1200	**19674.7**	**19028**	3.29	20279.9	19415	4.26
SC-n309-s21	1200	**18167.7**	**17069**	6.05	18267	17218	5.74

The results in Table 7 show that the adaptive version produces better results than the classic version and outperforms it in 14 of the 16 semi-clustered instances for the average and best results. This could potentially be explained by the fact that the adaptive version, as more VND runs are performed, focuses more on the most promising neighborhoods, while in the conventional version the search is conducted in the same manner regardless of the neighborhood's performance. The results obtained by the randomized

and clustered instances both support the indication that the $GVNS_{adaptive}$ is a more suitable schema for the HMSME-VRP. Table 8 presents the results of the randomly generated instances, while the results of the clustered instances are reported in Table 9. Furthermore, the figures 11 and 12 depict the number of wins between the two algorithms for each subset of instances for average and best solutions, respectively.

Table 8: Average and best solutions for the $GVNS_{Adaptive}$ and classic $GVNS$ for the random instances

		$GVNS_{Adaptive}$			$GVNS$		
Instance	cpu-time	Avg	Best	Gap	Avg	Best	Gap
R-n38-s6	180	**6688.7**	**6333**	5.32	6722.6	6593	1.93
R-n49-s8	180	**7883**	**7628**	3.23	8079.9	7791	3.58
R-n62-s8	180	**8943.9**	**8492**	5.05	9217.3	8693	5.69
R-n71-s3	180	**11991.5**	**11392**	5	12480.1	11715	6.13
R-n84-s9	180	**12587.7**	**11429**	9.21	12803.9	12094	5.54
R-n86-s6	180	**13720.3**	**12784**	6.82	13758.4	13279	3.48
R-n96-s7	180	**13866.3**	**13220**	4.66	14118.2	13690	3.03
R-n104-s5	180	**15255.3**	**14201**	6.91	15953.3	14761	7.47
R-n169-s14	600	17203.4	**16168**	6.02	**17187.6**	16685	2.92
R-n173-s15	600	**20647.5**	19482	5.64	20972.3	**19321**	7.87
R-n183-s14	600	25725.3	24920	3.13	**25237.9**	**23201**	8.07
R-n190-s13	600	**21943.4**	**20416**	6.96	22124.3	20639	6.71
R-n261-s15	1200	**29283.8**	**27390**	6.47	29453.2	28014	4.88
R-n265-s23	1200	**31485.4**	**29833**	5.25	31898.9	30927	3.05
R-n266-s27	1200	**34434.3**	**33097**	3.88	34960.5	33703	3.6
R-n298-s29	1200	**30905.2**	29842	3.44	30956.9	**29705**	4.04

Table 9: Average and best solutions for the $GVNS_{Adaptive}$ and classic $GVNS$ for the clustered instances

		$GVNS_{Adaptive}$			$GVNS$		
Instance	cpu-time	Average	Best	Gap	Average	Best	Gap
C-n44-s10	180	4630.2	4499	2.83	**4588.6**	**4478**	2.41
C-n46-s7	180	**6262.1**	6123	2.22	6351.5	**6100**	3.96
C-n50-s2	180	**6865.3**	**6649**	3.15	6993.8	6807	2.67
C-n59-s4	180	6711.8	**6420**	4.35	**6692.9**	6473	3.29
C-n61-s7	180	**8354.2**	**8087**	3.2	8595.1	8150	5.18
C-n69-s2	180	**7752**	**7516**	3.04	7960	7672	3.62
C-n75-s2	180	**8301.5**	**7920**	4.6	8434.4	8111	3.83
C-n99-s10	180	**13178**	**12387**	6	13527.9	12984	4.02
C-n119-s16	600	**10713.6**	**9880**	7.78	11150.4	10712	3.93
C-n199-s19	600	14566.5	**13639**	6.37	**14441.2**	13778	4.59
C-n203-s15.	600	**18255.7**	17504	4.12	18380.6	**17415**	5.25
C-n206-s10	600	18896.6	**17890**	5.33	**18769.5**	18065	3.75
C-n269-s22	1200	23009.5	**21501**	6.56	**22885.1**	21928	4.18
C-n285-s29.	1200	26874.7	25876	3.71	**26812.4**	**25646**	4.35
C-n294-s18	1200	**25665.2**	24942	2.82	26461.5	**24857**	6.06
C-n307-s28	1200	**21315.2**	20299	4.77	21337.5	**19899**	6.74

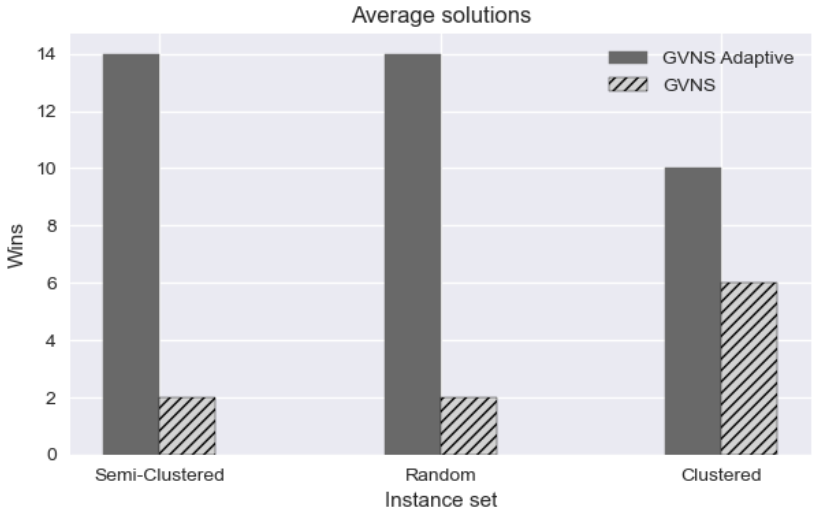

Figure 11: GVNS$_{adaptive}$ vs. classic GVNS based on average solutions

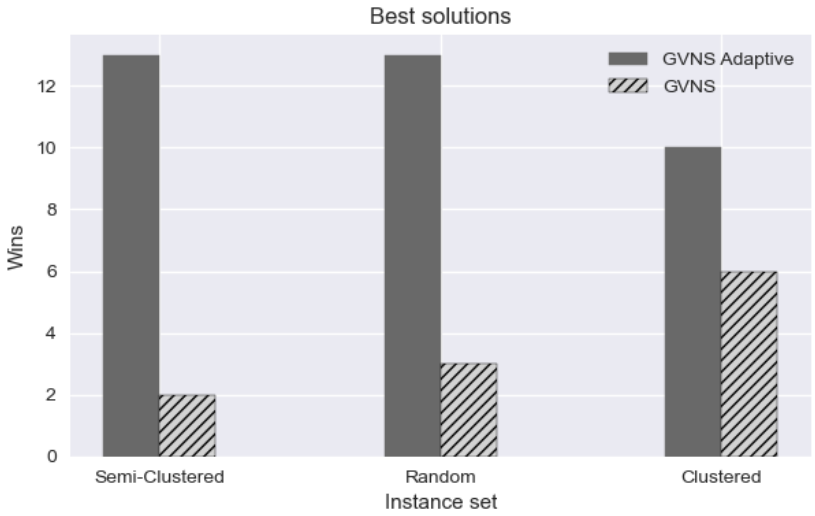

Figure 12: GVNS$_{adaptive}$ vs. classic GVNS based on best solutions

In the same way as for the semi-clustered instances, $GVNS_{adaptive}$ outperforms the conventional $GVNS$ in both average and best solutions for the subset of random and clustered instances. Regarding random instances, the

adaptive version obtains a better average solution in 14 of 16 instances and a lower best solution on 13 occasions compared to the conventional version. However, these results are less significant for clustered instances, where the adaptive version finds a better solution in 10 of the 16 instances with respect to average and best solutions.

5.4. Statistical tests

As shown in Table 4, both VNS algorithms are competitive compared to Gurobi, providing good quality solutions in a short time frame for small instances. However, as Gurobi could not find a feasible solution on larger instances within a reasonable time, in order to compare the two GVNS versions, a Wilcoxon Signed-Rank Test was performed on average and best solutions for the three instance sets to determine if there is statistical significance between the two algorithms with regard to performance. The test is the non-parametric version of the paired T-test and is used to compare the means of the two samples. In this particular instance, the null hypothesis would be that the mean of the samples, i.e., for each algorithm and subset of the instances, is the same, whereas the alternative hypothesis is that they are not. The results of the Wilcoxon test are reported in Table 10.

Table 10: Wilcoxon test results

		P-value
Semi-clustered instances	**Average solutions**	0.00763
	Best solutions	0.04089
Random instances	**Average solutions**	0.00336
	Best solutions	0.00919
Clustered instances	**Average solutions**	0.05768
	Best solutions	0.14386

Multiple conclusions can be drawn based on the results of the Wilcoxon test. First, there is statistical significance between the two algorithms with regard to their performance for the semi-clustered and randomly generated instances. The p-value of the average solutions of the semi-clustered instances average and the best solutions of the random instances are less than 1%, which means that even with a confidence level of 1%, we could reject the null hypothesis. However, the best solutions for the semi-clustered instances would require a significance level of 5% to reject the null hypothesis. Although there is no statistically significant difference between the two algorithms for clustered instances, the p-value is just above 5%. Furthermore, when comparing the p-values of the average and the best solutions for all subsets of the instances, the p-values of the average solutions are smaller than those of the best solution in all cases, which means that there is a

larger variation in the solutions obtained by the classic $GVNS$ than in the $GVNS_{adaptive}$.

6. Conclusions and future research

Being one of the most studied combinatorial optimization problems, there is a wide range of extensions and subclasses of the VRP. This paper studies the HMSME-VRP, a newly introduced version of the VRP that originates from a real-world application, and the policies of a Nordic distribution company. The VRP is an $\mathcal{NP}$-hard problem, and, as a version of that problem, the HMSME-VRP adds further complexity. Previous studies have shown that only small-sized instances can be solved using a commercial solver in a reasonable time. In order to solve realistically sized instances, in this paper, two different GVNS-based heuristics with multiple neighborhood structures, both in the diversification and intensification phase, have been developed and presented. In addition to a conventional GVNS, an adaptive GVNS has been proposed, which utilizes an intelligent re-ordering mechanism of the local search operators in the intensification phase based on their success in the previous iteration. Both algorithms provide good-quality solutions and outperform Gurobi.

Furthermore, 48 new benchmark instances have been generated and divided into three subsets based on their properties. The subsets correspond to randomly placed customers, clustered customers, and semi-clustered customers; of which the latter better mimics the distribution of switch points and customers in the real-life situation that initiated this study. The computational experimentation based on these benchmark instances shows that the adaptive version outperforms the conventional version of the GVNS schema in most of the cases with respect to both average and best solutions. Moreover, this superiority is statistically confirmed by the Wilcoxon test.

From a practical point of view, since the problem studied originates from a Nordic distribution company, the proposed algorithms constitute practical tools that can be useful for such companies. The HMSME-VRP enables companies to cover large geographical areas without significant investments in infrastructure. However, as a commercial solver cannot be used to solve the model, the proposed algorithm efficiently solves large-size instances within minutes and can thus be used in the process of designing distribution routes.

Several interesting future research directions are possible. Regarding the problem statement, additional constraints may be introduced to better model real-life aspects of the transportation system. Such extensions could, for example, include time windows, service times, or maximum route

time. Given the problem's complexity and the failure of exact solvers to solve large-scale instances, it would be interesting to verify the performance of the proposed GVNS algorithms by comparing them either to other metaheuristics, for instance, ant-colony optimization or to decomposition approaches, such as column generation schemes.

References

Absi, N., Cattaruzza, D., Feillet, D., and Housseman, S. (2017). A relax-and-repair heuristic for the swap-body vehicle routing problem. *Annals of Operations Research*, 253(2):957–978.

Accorsi, L. and Vigo, D. (2020). A hybrid metaheuristic for single truck and trailer routing problems. *Transportation Science*, 54(5):1351–1371.

Anderluh, A., Hemmelmayr, V. C., and Nolz, P. C. (2017). Synchronizing vans and cargo bikes in a city distribution network. *Central European Journal of Operations Research*, 25:345–376.

Anderluh, A., Nolz, P. C., Hemmelmayr, V. C., and Crainic, T. G. (2021). Multi-objective optimization of a two-echelon vehicle routing problem with vehicle synchronization and 'grey zone'customers arising in urban logistics. *European Journal of Operational Research*, 289(3):940–958.

Brimberg, J., Salhi, S., Todosijević, R., and Urošević, D. (2023). Variable neighborhood search: The power of change and simplicity. *Computers & Operations Research*, 155:106221.

Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51.

Cuda, R., Guastaroba, G., and Speranza, M. G. (2015). A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199.

Drexl, M. (2011). Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. *Revista de Métodos Cuantitativos para la Economía y la Empresa*, 12:5–38.

Duarte, A., Sánchez-Oro, J., Mladenović, N., and Todosijević, R. (2018). Variable neighborhood descent. In Martí, R., Pardalos, P., and Resende, M., editors, *Handbook of Heuristics*, pages 341–367. Springer, Cham.

European Commission (2023). *EU transport in figures: statistical pocketbook 2023*. Publications Office.

Guo, J., Long, J., Xu, X., Yu, M., and Yuan, K. (2022). The vehicle routing problem of intercity ride-sharing between two cities. *Transportation Research Part B: Methodological*, 158:113–139.

Hansen, P., Mladenović, N., Brimberg, J., and Pérez, J. A. M. (2019). Variable neighborhood search. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of Metaheuristics*, pages 57–97. Springer International Publishing, Cham.

Hansen, P., Mladenović, N., Todosijević, R., and Hanafi, S. (2017). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, 5(3):423–454.

Huber, S. and Geiger, M. J. (2014). Swap body vehicle routing problem: A heuristic solution approach. In González-Ramírez, R. G., Schulte, F., Voß, S., and Ceroni Díaz, J. A., editors, *Computational Logistics*, pages 16–30, Cham. Springer International Publishing.

Kalatzantonakis, P., Sifaleras, A., and Samaras, N. (2020). Cooperative versus non-cooperative parallel variable neighborhood search strategies: a case study on the capacitated vehicle routing problem. *Journal of Global Optimization*, 78(2):327–348.

Kalatzantonakis, P., Sifaleras, A., and Samaras, N. (2023). A reinforcement learning-variable neighborhood search method for the capacitated vehicle routing problem. *Expert Systems with Applications*, 213:118812.

Karakostas, P. and Sifaleras, A. (2022). A double-adaptive general variable neighborhood search algorithm for the solution of the traveling salesman problem. *Applied Soft Computing*, 121:108746.

Karakostas, P., Sifaleras, A., and Georgiadis, M. C. (2020). Adaptive variable neighborhood search solution methods for the fleet size and mix pollution location-inventory-routing problem. *Expert Systems with Applications*, 153:113444.

Leach, D. Z. and Savage, C. J. (2012). Impact assessment: High capacity vehicles.

Liimatainen, H., Pöllänen, M., and Nykänen, L. (2020). Impacts of increasing maximum truck weight–case finland. *European Transport Research Review*, 12:1–12.

Lin, S.-W., Vincent, F. Y., and Lu, C.-C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, 38(12):15244–15252.

Liu, R. and Jiang, S. (2022). A variable neighborhood search algorithm with constraint relaxation for the two-echelon vehicle routing problem with simultaneous delivery and pickup demands. *Soft Computing*, 26(17):8879–8896.

Macrina, G., Pugliese, L. D. P., Guerriero, F., and Laporte, G. (2020). Crowd-shipping with time windows and transshipment nodes. *Computers & Operations Research*, 113:104806.

Miranda-Bront, J., Curcio, B., Méndez-Díaz, I., Montero, A., Pousa, F., and Zabala, P. (2017). A cluster-first route-second approach for the swap body vehicle routing problem. *Annals of Operations Research*, 253(2):935–956.

Mirmohammadsadeghi, S. and Ahmed, S. (2015). Memetic heuristic approach for solving truck and trailer routing problems with stochastic demands and time windows. *Networks and Spatial Economics*, 15(4):1093–1115.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Pei, J., Mladenović, N., Urošević, D., Brimberg, J., and Liu, X. (2020). Solving the traveling repairman problem with profits: A novel variable neighborhood search approach. *Information Sciences*, 507:108–123.

Perboli, G., Tadei, R., and Vigo, D. (2011). The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45(3):364–380.

Sluijk, N., Florio, A. M., Kinable, J., Dellaert, N., and Van Woensel, T. (2022). Two-echelon vehicle routing problems: A literature review. *European Journal of Operational Research*.

Smiti, N., Dhiaf, M. M., Jarboui, B., and Hanafi, S. (2020). Skewed general variable neighborhood search for the cumulative capacitated vehi-

cle routing problem. *International Transactions in Operational Research*, 27(1):651–664.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

Song, L., Gu, H., and Huang, H. (2017). A lower bound for the adaptive two-echelon capacitated vehicle routing problem. *Journal of Combinatorial Optimization*, 33:1145–1167.

Tadaros, M., Migdalas, A., and Samuelsson, B. (2023). A note on the hierarchical multi-switch multi-echelon vehicle routing problem. *Optimization Letters*, 17:1469–1486.

Todosijević, R., Hanafi, S., Urošević, D., Jarboui, B., and Gendron, B. (2017). A general variable neighborhood search for the swap-body vehicle routing problem. *Computers & Operations Research*, 78:468–479.

Toffolo, T., Christiaens, J., Van Malderen, S., Wauters, T., and Vanden Berghe, G. (2018). Stochastic local search with learning automaton for the swap-body vehicle routing problem. *Computers & Operations Research*, 89:68–81.

Toth, P. and Vigo, D. (2002). *The vehicle routing problem*. SIAM.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2010). GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. *Engineering Applications of Artificial Intelligence*, 23(5):780–794.

Appendix A. Instance Generation

For this paper, 48 instances were generated following the procedure described in Algorithm 7. These are divided into three different subsets based on how customers have been located, that is, instances of randomly located customers (R), of clustered customers (C), and semi-clustered instances (SC).

Algorithm 7 Instance Generation

Require: lC,hC: lowest and highest number of clustered customers,
lR,hR: lowest and highest number of randomly located customers,
lS,hS: lowest and highest number of switch points,
lD,hD: lowest and highest demand

Ensure: *cust* : an array of customer locations,
customers_demand: an array of customer demand,
switches: an array of switch locations,
dep: an array of the depot location,
Cap: the vehicle capacity

procedure INSTANCE GENERATION(lC,hC, lR,hR, lS,hS, lD,hD)
    $num\_clust \leftarrow \text{random}(lC, hC)$
    $num\_ran \leftarrow \text{random}(lR, hR)$
    $tot\_cust \leftarrow num\_clust + num\_ran$
    $num\_switch \leftarrow \text{random}(lS, hS)$
    $switches \leftarrow \text{random}(200, 800, \text{size} = (num\_switch, 2))$
    $corners \leftarrow \text{array}([[0, 0], [1000, 0], [0, 1000], [1000, 1000]])$
    $randCust \leftarrow \text{random}(0, 1000, \text{size} = (num\_ran, 2))$
    $dep \leftarrow \text{random}(0, 1000, \text{size} = (1, 2))$
    $demand \leftarrow \text{random}(lD, hD, \text{size} = (tot\_cust, 1))$
    if $num\_clust > 0$ **then**
        $cust \leftarrow CLUSTERING(num\_clust, switches)$
    else
        $cust \leftarrow \text{array}([])$
    end if
    $cust \leftarrow \text{concatenate}((cust, rand\_C))$
    $customers\_demand \leftarrow \text{concatenate}((cust, demand), \text{axis} = 1)$
    $n \leftarrow \text{int}(tot\_cust/12)$
    $p \leftarrow \text{int}(tot\_cust/8)$
    $R_{min} \leftarrow tot\_cust/\text{random}(n, p)$
    $Cap \leftarrow \lceil \text{R}_{\text{min}} * \text{total\_demand}/\text{tot\_cust} \rceil$
    return *cust*, *customers_demand*, *switches*, *dep*, Cap
end procedure

The parameters used in Algorithm 7 are listed in Table A.11. For each subset, eight small, four medium, and four large instances were generated. The parameters for the demand generation are listed in Table A.12. An instance from each subset is depicted in Figures A.13–A.15 for illustration

purposes.

Table A.11: Parameters for instance generation

Instance Type	Size	Parameters					
		lC	hC	lR	hR	lS	hS
SC	Small	20	70	5	30	2	10
	Medium	85	160	15	40	10	20
	Large	170	260	30	40	15	30
R	Small	0	0	25	100	2	10
	Medium	0	0	100	200	10	20
	Large	0	0	200	300	15	30
C	Small	25	100	0	0	2	10
	Medium	100	200	0	0	10	20
	Large	200	300	0	0	15	30

Table A.12: Parameters for demand generation

Variance	Values	lD	hD
Low	Small	5	10
	Large	50	100
High	Small	1	10
	Large	1	100

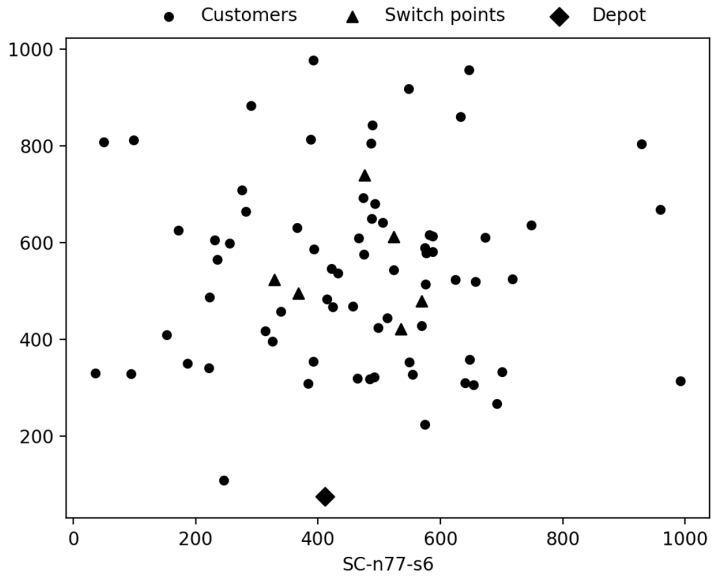

Figure A.13: SC-n77-s6

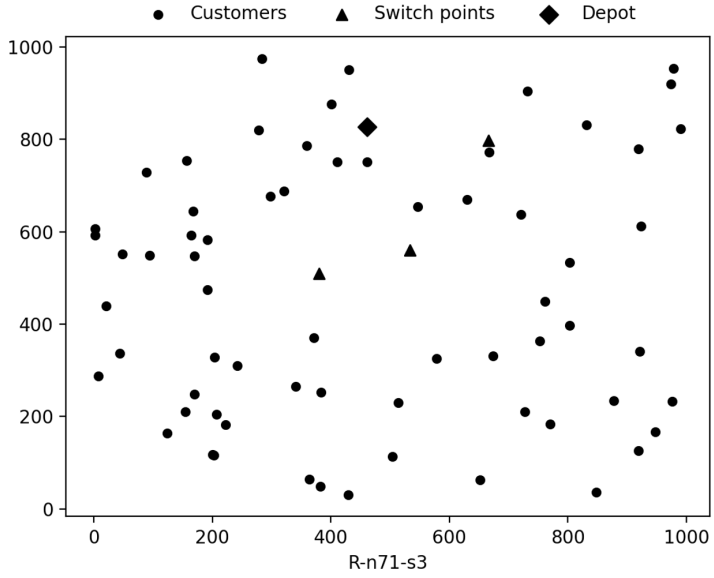

Figure A.14: R–71-s3

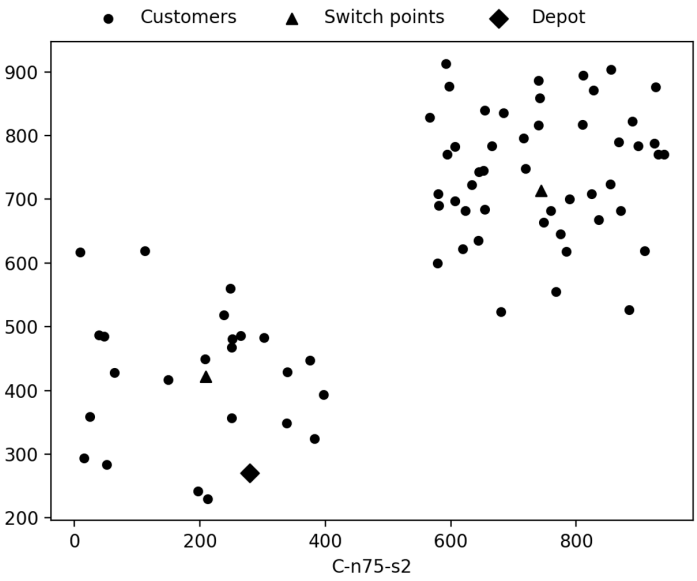

Figure A.15: C-n75-s2