Wageningen University - Department of Social Sciences

Operations Research and Logistics

MSc Thesis

Comparing Clustering Algorithms in a Time Window Constrained Vehicle Routing Problem

This thesis compares the influence of different clustering methods on the results of a clustering first routing second approach (CFRS) to a set of time window constrained vehicle routing problem instances (TWVRP). Finding a short path solution may reduce business expenses by reducing the number of vehicles required to serve all nodes and the total distance travelled by these vehicles.
The clustering methods K-means, DBSCAN, a sweep algorithm and MIP clustering are compared. After the clustering phase each cluster is routed with a MILP model. Incorporating the cannot-link constraint that prevents nodes from being clustered together into the different clustering algorithms allows for the consideration of time window constraints in the clustering phase. Cannot-link node pairs were identified before clustering as a preprocessing step.
The different approaches were tested on the R1, C1 and RC1 instances of the Solomon benchmark that where the shortest path is the best result. The shortest paths were found using MIP clustering followed by DBSCAN, the sweep algorithm and K-means. The approach using cannot-link constraints performs better on instances with a high time window density and restrictiveness.

Word count: 10.986
May 2023

Student	Jorn Kuijpers
Registration number	1036021
MSc program	Master Food Technology
Specialization	Sustainable Food Process Engineering
Commissioner	WUR
Supervisor(s)	Dr. Rene Haijema (WU)
Examiner/2nd supervisor	Dr. Dmytro Krushynskyi
Thesis code	ORL 88843

WAGENINGEN
UNIVERSITY & RESEARCH

Table of Contents

1 Introduction

The vehicle routing problem (VRP) was introduced in 1959 by Dantzig and Ramser as a proposition for finding the optimal routing of gasoline trucks (Dantzig & Ramser, 1959). Finding the optimal route for a fleet of vehicles is an NP-hard problem meaning that with an increase in problem size the number of possible solutions increase at a factorial rate and requires equally steep rising computational efforts (Toth & Vigo, 2002). By using VRP knowledge in the routing of a fleet of vehicles, the logistic costs of a company may be reduced but also the travelling and waiting time of drivers might be reduced, creating a more pleasant work environment. To reduce the computational power and duration of the calculation heuristics are used to generate an acceptable solution. The clustering first routing second approach (CFRS) has been a popular method since its introduction using the sweep algorithm (Gillett & Miller, 1974). With the growing importance of data science new clustering methods have emerged over the years and a variety has been used in CFRS approaches to the vehicle routing problem.

As different clustering algorithms cluster on different principles, the resulting clusters will vary and therefore the choice of clustering method may have an effect on the quality of the solution. (Han et al., 2012). (Cömert et al., 2017) compared the results of several types of modern clustering algorithms in a CFRS heuristic for solving the vehicle routing problem with hard time windows. The K-means, K-medoids and DBSCAN algorithms are used for clustering after which each cluster is solved as an individual travelling salesman problem (TSP). In a similar approach K-means, DBSCAN and a medoid heuristic were compared. However, after routing the results were optimized using the genetic algorithm (GA) (Gocken & Yaktubay, 2019). Other clustering methods used in literature include adaptations of the original sweep algorithm and mixed integer linear programming (MILP) models. The route for each vehicle is determined using exact methods such as MILP models but also heuristic approaches such as the Lin-Kernighan method (Shalaby et al., 2021; Zunic et al., 2020). In VRPTWs with different CFRS approaches in literature, the vehicle capacity is taken into consideration in the clustering phase but the time window constraints are solely considered in the routing phase.

Finding a feasible and acceptable solution to the vehicle routing problem on a large dataset with capacity and time window constraints is difficult due to the size of the problem. In literature the clustering first routing second approach has been widely used to solve vehicle routing problems. Comparisons of the performance of different clustering methods have been limited to centroid and density based algorithms such as K-means and DBSCAN. Other clustering methods such as the sweep algorithm and MILP models are not evaluated in the comparison. In the clustering phase of TWVRPs the time window constraints are not considered and time window complications in the routing phase require additional vehicles to route a cluster or transferring nodes to other clusters is needed. The research question of this thesis therefore is:

"What is the impact of different clustering methods, with consideration of time window constraints in the clustering phase, on the quality of solution in a clustering first routing second approach to the Vehicle Routing Problem with Time Windows?"

In this thesis the performance of different clustering methods will be compared for the CFRS approach to a TWVRP. The clustering methods that will be compared are K-means, DBSCAN, a sweep algorithm and MIP clustering. The clusters will be routed using a MILP model. The clustering

algorithms will be capacity constrained and a cannot-link constraint will be included that is inspired by the semi supervised COP K-means clustering algorithm (Wagstaff et al., 2001). The cannot-link constraints will be based on node pairs and in a preprocessing step, unfavorable node pairs will be identified and added to a cannot-link set. Node pairs in the cannot link set will be prevented from being clustered together. The different clustering methods and the influence of the cannot-link constraint will be tested on 29 datasets of the Solomon benchmark.

This thesis starts with a literature review of clustering methods in clustering first routing second approaches that are being used for solving different types of VRP. The different approaches are compared and the clustering, routing and potential optimization methods are classified. After that the methodology behind the research is explored in further detail. The method is presented as a mathematically formulated MILP model. The capacity and cannot-link constraints are elaborated and the different clustering methods are discussed. Finally the results are presented and discussed after which a conclusion is drawn.

2 Literature review

The vehicle routing problem was introduced as the truck dispatching problem, which had the goal of optimizing the routes of a fleet of gasoline trucks that supply gas stations from a terminal (Dantzig & Ramser, 1959). The general aim of a vehicle routing problem is to find the optimal routes for a fleet of vehicles that are subject to a set of constraints. The optimal routes may be dependent on the costs of the routes, the working hours for the drivers or the total emissions. Among the most common constraints are the capacities of the vehicles and the time windows of serving nodes (Konstantakopoulos et al., 2022). The vehicle routing problem is an NP-hard combinatorial optimization problem and the number of solutions increases at a factorial rate with the amount of nodes in the problem. As the number of solutions increases the required computational power increases and finding the global optimal solution will be increasingly time consuming (Toth & Vigo, 2002). To reduce the computational requirements and duration of finding a solution different methods have been developed over the years that attempt to find acceptable solutions.

2.1 VRP approaches

As the vehicle routing problem is a well-researched optimization problem and since its introduction in 1959 different strategies for finding exact and approximate solutions have been employed. A classification of the different approaches to solving VRPs is shown in Figure 2.1. These solution strategies can initially be divided into two categories: one-phase and two-phase methods. One-phase approaches are subdivided into exact methods and metaheuristics. Exact methods find the global optimum at the cost of long computational times. Metaheuristics is a general term for advanced search strategies that are applied in mathematical optimization. These strategies are not limited to VRPs and have applications for a range of optimization problems. The two-phase approaches are clustering first routing second (CFRS) and routing first clustering second (RFCS). These methods reduce the solution space in the first phase and find a solution in the second phase. The basic CFRS approach assigns all nodes to clusters after which a travelling salesman problem (TSP) is solved for each cluster. The RFCS approach generates a route that traverses all nodes, after which the route is divided into segments that are serviced by a vehicle.

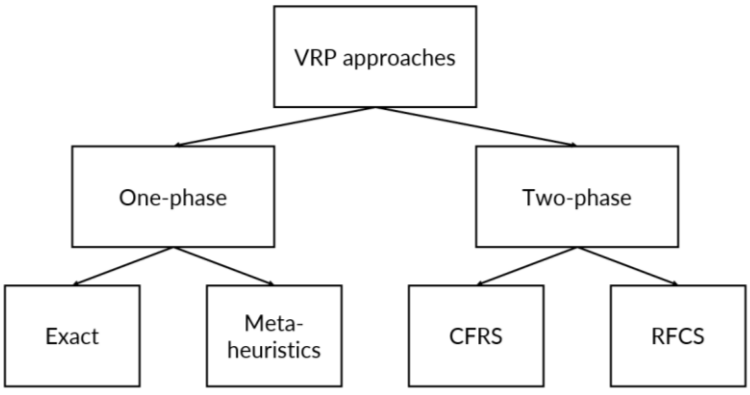

Figure 2.1 Classification of VRP approaches

The different strategies aim to reduce the duration of finding an acceptable solution in comparison to the baseline exhaustive search. Exhaustive search is a brute-force approach that involves evaluating each solution in the search space. Of the four strategies in the classification, exact methods are the only approach to consistently find the global optimal solution. The other methods, metaheuristics and two phase approaches, are not guaranteed to find the overall best solution. Of the exact methods some of the commonly employed approaches are column generation and Lagrange relaxation (Konstantakopoulos et al., 2022). These methods are common methods for large scale optimization problems and both approaches aim to reduce the computational capacity required for finding the global optimum relative to exhaustive search. However, finding the global optimum still requires an extensive exploration of the search space. As the size of the VRP increases linearly the solution space grows at a factorial rate. Despite the improvement in comparison to brute-force solving the VRP, exact methods still require steeply increasing amounts of time and processing power with increasing VRP sizes.

To improve the computational efficiency of solving VRPs, approximate methods have been developed. These approaches aim to reduce the computational complexity and solve VRPs in a shorter amount of time or with fewer computational resources. However, the reduced computational cost comes at the expense of a potentially lower quality of solution. These approaches are prone to getting stuck in local optima and explore less search space than the exact methods.

Despite this limitation, this trade-off between quality and computational efficiency is deemed acceptable in practical applications. The two-phase CFRS and RFCS approaches are techniques consisting of a set of rules for finding good solutions at reasonable computing cost while the one phase metaheuristics provide a guiding process for the search for a solution and are considered a more top-level strategy (Voß, 2008). In recent years, metaheuristics for VRPs have gained popularity as computing power has increased. Metaheuristics for vehicle routing problems are among the state-of-the-art approaches. Metaheuristics are advanced guiding methods for finding solutions to optimization problems. These metaheuristics are not problem specific but can be employed for diverse optimization problems as they are based on an optimization concept that can be applied in different circumstances. The one-phase approach of metaheuristics explore the search space in an iterative manner in which the direction of exploration is based on altering the variables intelligently (Osman & Kelly, 1997). An example of this intelligent alteration of the variables is the genetic algorithm (GA) where new solutions are generated by combining the variables of different solutions in the population.

The two-phase approaches are among the first solution approaches to be used for VRPs. In contrast to metaheuristics, these are not applicable to multiple fields and are specifically designed for finding approximate solutions for VRPs (Voß, 2008). The two-phase approaches reduce the problem space by dividing the problem into two phases, an assignment phase and a routing phase. The CFRS and RFCS approaches differ in the order of the clustering and routing phases. Specifically, CFRS first clusters the nodes before routing the clusters while RFCS first constructs a complete route that passes through all nodes and then divides the route into segments that are assigned to individual vehicles, based on the constraints of the problem.

The CFRS approach was initially proposed to solve a capacitated vehicle routing problem (CVRP) using a sweep algorithm in its clustering phase. The first iteration sweep algorithm traverses the

nodes in a clockwise direction and assigns each node it encounters to vehicle one until its capacity is reached, then it assigns the next nodes to the next available vehicle. This process of vehicle assignment reduces the full truck dispatching problem into multiple TSPs that can be solved individually. Subsequently, each TSP is\\was routed individually to find the optimal route for each vehicle using the Lin and Kernighan heuristic method (Gillett & Miller, 1974; Lin & Kernighan, 1971).

As a result of the complete independence between the clustering and routing phases, two phase approaches carry the risk of attaining suboptimal, locally optimized solutions instead of globally optimal ones (Huang & Hu, 2012). In practice, it may not be realistic to obtain the global optimal solution for VRPs due to the computational time required. As a result, locally optimized solutions obtained through two-phase approaches may still be deemed acceptable. However, the quality of the overall solution is heavily influenced by the clustering phase. Since the routing phase is dependent on the clusters, the choice of clustering algorithm is crucial when designing an effective CFRS approach.

2.2 Clustering methods

The goal of clustering algorithms is to group objects together so that they are similar to other objects in the same cluster and dissimilar to objects in other clusters. However, as different clustering algorithms cluster on different principles, the resulting clusters may vary and therefore the choice of clustering method will have an effect on the cluster formation (Han et al., 2012). A range of clustering algorithms has been used to solve VRPs using the CFRS heuristic. These clustering methods include a variety of sweeping algorithms, distance based clustering such as K-means and density based clustering such as DBSCAN. Given the impact of the clustering method on the final results of a CFRS approach, it is crucial to carefully consider the clustering method used. Therefore, it is essential to identify which clustering methods are effective for VRPs and to carefully evaluate their impact on the results.

The CFRS approaches in literature can be classified by a set of characteristics. Primary characteristics are VRP type, clustering algorithm and routing method. Secondary characteristics are dataset size, cluster optimization method and a potential optimization method. Table 2.1 shows the classification of CFRS approaches in literature.

Table 2.1 Classification of CFRS approaches in literature.

Clustering	Specific Clustering	Article	VRP Type	Dataset Size	Routing Algorithm	Optimization
K-means	K-means	(Cömert et al., 2017)	TWVRP	78	MILP	None
	K-means	(Cömert et al., 2018)	CVRP	78	Branch-and-bound	None
	K-means	(Le et al., 2022)	TWVRP	39	MILP	None
	Fuzzy C-means	(Shalaby et al., 2021)	CVRP	32-16000	Lin-Kernighan	None
	K-means	(Gocken & Yaktubay, 2019)	TWVRP	100	Random-based algorithm	**Route:** Genetic Algorithm
Other centroid based algorithms	PAM	(Cömert et al., 2017)	TWVRP	78	MILP	None
	K-medoids	(Cömert et al., 2018)	CVRP	78	Branch-and-bound	
	Centroid-based heuristic	(Gocken & Yaktubay, 2019)	TWVRP	100	Random-based algorithm	**Route:** Genetic Algorithm
	Centroid-based heuristic	(Shin & Han, 2011)	CVRP	31-79	Lin-Kernighan	None
Sweep algorithm	Sweep algorithm	(Gillett & Miller, 1974)	CVRP	29	Lin-Kernighan	None
	Time Window Length	(Armbrust et al., 2022)	TWVRP	250-1000	MILP	None
	Distance based Sweep Nearest	(Peya et al., 2019)	CVRP	32-80	GA, ACO & PSO	None
	Time Window based algorithm	(Zunic et al., 2020)	MDVRP	237	MILP	None
Density based algorithms	Capacitated DBSCAN	(Cömert et al., 2017)	TWVRP	78	MILP	None
	Capacitated DBSCAN	(Gocken & Yaktubay, 2019)	TWVRP	100	Random-based algorithm	**Route:** Genetic Algorithm
	Capacitated SNN	(Gocken & Yaktubay, 2019)	TWVRP	100	Random-based algorithm	**Route:** Genetic Algorithm

Integer programming	Integer programming	(Garside & Laili, 2019)	MTPVRP	29	Tabu search	None
	Integer programming	(Fisher & Jaikumar, 1981)	CVRP	50-100	constraint relaxation	None
	Integer programming	(Boonsam et al., 2011)	TWVRP	377	Lin-Kernighan	None

Although the CFRS approach is considered to be a more conventional method than some metaheuristics, the approach has been revisited over the years considering a variety of clustering algorithms and VRP types. The most common VRP types for which the CFRS approach is used are the CVRP and TWVRP. It is not unexpected that these VRP types are highly represented as in larger literature reviews the share of CVRP and TWVRP exceed 40%. Research utilizing the CFRS approach has been conducted as early as 1974 and as recent as 2022.

The sweep algorithm, introduced in 1974, is considered to be the pioneering version of the CFRS approach (Gillett & Miller, 1974). The sweep algorithm has seen many iterations over the years and new applications are still being developed. The algorithm employs a node-ordering method based on specific characteristics such as distance to the depot or time window length, and the first node in the order is assigned to cluster one. The algorithm then adds the closest node to the first node, and continues adding nodes to the cluster as long as the capacity constraint is not exceeded. Once the capacity limit is reached, the process of filling a cluster is restarted (Armbrust et al., 2022; Peya et al., 2019; Zunic et al., 2020). An advantage of this hierarchical approach is that it guarantees consistent clustering results, a well-designed algorithm may consistently find acceptable solutions in a short amount of time. The main limitation is that the hierarchical nature limits the exploration of the search space as the sweep algorithm produces identical results on the same dataset across multiple iterations.

The most prevalent clustering methods in literature are the centroid based algorithms. The general methodology of centroid-based algorithms such as K-means and K-medoids involves assigning data points to a cluster based on their distance from that clusters' centroid. Of the centroid based algorithms, K-means is the most common. Other employed centroid based algorithms are PAM and K-medoids. Generic K-means is an unsupervised learning algorithm that clusters data into the cluster with the closest centroid. The variable K is a predetermined parameter that specifies the number of clusters to be created. K centroids are randomly chosen and after the assignment of datapoints the centroids are recalculated as the mean of all the datapoints in the cluster. In VRP application capacitated K-means, a semi-supervised version of the generic K-means algorithm, is often implemented. Capacitated K-means does not only assign data to the cluster with the closest center, each data point or node has a demand value and the sum of these demands in a cluster may not exceed a vehicle capacity. Adding this capacity constraint to a cluster ensures that the summed demands of the nodes in a cluster do not exceed the capacity of the vehicle that is to serve the cluster (Cömert et al., 2017, 2018; Gocken & Yaktubay, 2019; Le et al., 2022). In comparisons of the performance of clustering algorithms in CFRS approaches for the VRPTW, capacitated K-means was found to be the best performing clustering method (Cömert et al., 2017; Gocken & Yaktubay, 2019).

Density based clustering algorithms are unsupervised learning methods that group data points into clusters based on their proximity to high density data regions. The most prevalent density based algorithm in literature is density-based spatial clustering of applications with noise (DBSCAN). DBSCAN is an unsupervised clustering algorithm that assigns data points to clusters based on their pairwise distances. This algorithm is initialized by assigning core points that are identified based on the values of two input parameters, MinPts and Eps. A datapoint is deemed a core point if it has at least MinPts neighbors within a distance of Eps. Once the core points are identified, all points within the Eps-neighborhood of a core point are added to the corresponding cluster. The algorithm then iteratively adds more points to the same cluster as long as they are within the Eps distance of the previously added datapoints. One major advantage of DBSCAN is that it does not require a prior specification of the number of clusters, but instead derives the cluster structure from the input data and the parameters. However, this implicates that the choice of MinPts and Eps heavily influences the clustering results. Datapoints that do not belong to any cluster and are located beyond the Eps-distance of any core point are considered noise and are not assigned to any cluster. When applying DBSCAN to a VRP, setting the Eps value to be at least the distance between the most isolated node and its closest neighbor is recommended to prevent nodes from being classified as noise. In literature the DBSCAN algorithm is capacity constrained similar to K-means and the sweep algorithm. This reclassifies the capacitated DBSCAN from an unsupervised learning algorithm to a semi-supervised learning algorithm (Cömert et al., 2017; Gocken & Yaktubay, 2019).

Clustering through mixed integer linear programming (MILP) was introduced for CFRS approaches to the VRP in 1981 by clustering the nodes using a Generalized Assignment Problem (GAP) and subsequently solving a TSP for each cluster (Fisher & Jaikumar, 1981). At the time it outperformed all existing heuristics. Clustering with a MILP model involves optimizing an objective function subject to a set of constraints. This provides tailored clustering applications and straightforward incorporation of additional constraints. However, compared to other clustering algorithms such as K-means and DBSCAN, MILP clustering application can be computationally intensive and constructing a MILP clustering model may require domain-specific expertise. An objective function used for MILP clustering for CFRS approaches is minimizing the distance between nodes within clusters (Boonsam et al., 2011). The MILP clustering approach is used for more complex VRP types such as the Periodic Multi Trip VRP (MPVRP) as it allows for easier incorporation of additional constraints (Garside & Laili, 2019).

When designing a CFRS approach, research that compares the performance of clustering algorithms could be valuable. (Cömert et al., 2017) evaluated the performance of centroid based and density based clustering algorithms in a CFRS approach to solving a TWVRP instance. In a CFRS approach that was afterwards optimized using the genetic algorithm (GA) two centroid based and two density based clustering algorithms were compared (Gocken & Yaktubay, 2019). In these comparisons the centroid based clustering algorithms performed better than the density based algorithms. A comparison of different sweep algorithms was conducted in a separate study. The nodes were clustered based on different characteristics in the algorithms (Armbrust et al., 2022). The comparison of centroid- and density based algorithms to other clustering methods such as the sweep algorithm or MILP clustering is limited.

2.3 TSP

In the second phase of CFRS approaches, the routes within the clusters are generated. As the clustering methods generally impose capacity constraints such that the total demands of the nodes in a cluster do not exceed the vehicle capacity, clusters can be serviced by a single vehicle. A TSP, which involves finding the shortest route that visits each node exactly once, can be solved for each cluster. In the CFRS approaches in literature, routing within the clusters was done using a variety of methods. Among the most common methods are the Lin & Kernighan heuristic algorithm and MILP models. The Lin & Kernighan heuristic is a method to finding a high quality route in a short amount of time. Unlike exact methods such as MILP routing or the Branch-and-Bound algorithm that explore the entire search space, the Lin & Kernighan heuristic is an approximate method and does not guarantee finding the optimal route.

The challenge in clustering for different types of VRP is that as the constraints in the problem become more advanced, the routing within the clusters becomes more complicated. If the constraints cannot be fully incorporated in the clustering phase, infeasible node combinations may still occur in clusters which prevents solving a TSP for that cluster. A routing algorithm that takes into account time windows and route feasibility is required when routing a cluster for a TWVRP, an example can be found in (Gocken & Yaktubay, 2019). A random based routing algorithm is adapted to relocate nodes to different clusters when they cause infeasibility in a route. The general approach in literature consisted of solving a TSP for each cluster. More vehicles may be introduced if a single vehicle is not feasible which expands the TSP to a small-scale VRP (Le et al., 2022). In a contrasting approach, clusters were routed in two phases in a multi-depot VRP (MDVRP), first assigning clusters to vehicles after which the vehicles were assigned to a depot (Zunic et al., 2020). After solving a VRP different metaheuristics can be used to improve on the existing solution, in the research that was considered one study used a GA for route improvement, which attempts to find improvements on current solutions by repeatedly modifying them (Gocken & Yaktubay, 2019).

3 Methodology

This thesis addresses the TWVRP by applying a CFRS approach to solve 29 instances of the problem. The R1, C1 and RC1 datasets from the Solomon benchmark are used as test cases. These datasets are first clustered using four different clustering algorithms after which the clusters are routed with a MILP model. The total distance travelled by all vehicles is used as a performance metric for comparing the performance of the different clustering algorithms. A capacity and cannot-link constraint were implemented into the clustering algorithms. The algorithms are K-means, DBSCAN, a sweep algorithm and MIP Clustering. This section begins with describing the TWVRP in a MILP formulation. This is followed by an elaboration on the cannot-link constraint that has been implemented in the clustering methods. Then the individual clustering algorithms are discussed and the pseudocode of the algorithms is presented. Lastly, the routing within the clusters using a MILP model is discussed.

3.1 Mathematical model

In a TWVRP nodes must be served in their specified time window by a vehicle that does not have its capacity exceeded. The TWVRP is presented as a mathematical model to illustrate the objective and constraints for the CFRS approach. The objective function and set of constraints for this TWVRP can be described in a general mathematical model adapted from (Cordeau et al., 2002). The model is formulated as a MILP model and is presented below in objective function 3.1 and constraints 3.2 through 3.23. The objective function aims to minimize the total distance traveled by all vehicles, while the constraints ensure that each customer is served exactly once within its time window and the vehicle capacity is not exceeded. This mathematical model contains elements that are relevant in both the clustering phase and the routing phase. Constraints 3.9 through 3.11 formulate the capacity and cannot-link constraints that are implemented in the clustering algorithms. Constraints 3.12 through 3.16 formulate the time window constraints that are essential in the routing phase. The notations used in the MILP formulation are presented below.

Notations:

$\mathcal{V} = set\ of\ all\ nodes\ including\ the\ depot\ (0, 1, \dots, n)$

$\mathcal{N} = set\ of\ all\ nodes\ excluding\ the\ depot\ (\ 1, 2, \dots, n)$

$\mathcal{K} = set\ of\ all\ vehicles\ (1, 2, \dots, q)$

$i = node\ in\ set\ \mathcal{N}\ /\ node\ or\ depot\ in\ set\ \mathcal{V}$

$k = vehicle\ in\ set\ \mathcal{K}$

$n = number\ of\ nodes\ in\ the\ dataset$

$q = number\ of\ vehicles\ in\ set\ \mathcal{K}$

$t_{ij} = Eucliadean\ distance\ between\ node\ i\ and\ node\ j$

$x_{ijk} = binary\ variable, 1\ if\ arc\ (i, j)\ is\ travelled\ by\ vehicle\ k, 0\ if\ not$

$x_{ik} = binary\ variable, 1\ if\ node\ i\ is\ serviced\ by\ vehicle\ k, 0\ if\ not$

$w_{ij} = binary\ variable, 1\ if\ nodes\ i\ and\ j\ are\ served\ by\ the\ same\ vehicle, 0\ if\ not$

$y_{ij} = binary\ variable, 1\ if\ there\ is\ a\ cannot\ link\ constraint\ between\ node\ i\ and\ node\ j, 0\ if\ not$

$d_i = demand\ of\ node\ i$

$s_i = service\ time\ of\ node\ i$

$a_i = start\ of\ time\ window\ node\ i$

$b_i = end\ of\ time\ window\ node\ i$

$m_{ik} = arrival\ at\ node\ i\ when\ serviced\ by\ vehicle\ k$

$l_{ik} = time\ of\ leaving\ node\ i\ when\ serviced\ by\ vehicle\ k$

$C = capacity\ of\ each\ vehicle\ k$

$M = big\ M$

$u_i = MTZ\ index\ for\ nodes\ i\ in\ set$

$$minimize \sum_{i=0}^{\mathcal{V}} \sum_{j\neq i}^{\mathcal{V}\backslash\{i\}} \sum_{k=1}^{\mathcal{K}} x_{ijk} t_{ij} \quad (3.1)$$

s.t.

$$\sum_{j\neq i}^{\mathcal{N}\backslash\{i\}} \sum_{k=1}^{\mathcal{K}} x_{ijk} = 1 \qquad \forall\, i \in \mathcal{N} \qquad (3.2)$$

$$\sum_{j\neq i}^{\mathcal{N}\backslash\{i\}} \sum_{k=1}^{\mathcal{K}} x_{jik} = 1 \qquad \forall\, i \in \mathcal{N} \qquad (3.3)$$

$$\sum_{j\neq i}^{\mathcal{V}\backslash\{i\}} x_{ijk} - \sum_{j\neq i}^{\mathcal{V}\backslash\{i\}} x_{jik} = 0 \qquad \forall\, i\in\mathcal{V}\ \forall\, k\in\mathcal{K} \qquad (3.4)$$

$$\sum_{i=1}^{\mathcal{N}} x_{0jk} = 1 \qquad \forall\, k \in \mathcal{K} \qquad (3.5)$$

$$\sum_{i=1}^{\mathcal{N}} x_{j0k} = 1 \qquad \forall\, k \in \mathcal{K} \qquad (3.6)$$

$$u_i - u_j + n(x_{ijk}) \leq n - 1 \qquad \forall\, i \in \mathcal{V}\ \forall\, j \in \mathcal{V}\backslash\{i\}\ \forall\, k \in \mathcal{K} \qquad (3.7)$$

$$u_0 = 0 \qquad (3.8)$$

$$\sum_{i=1}^{\mathcal{N}} d_i x_{ik} \leq C \qquad \forall\, k \in \mathcal{K} \qquad (3.9)$$

$$x_{ik} + x_{jk} - 1 \leq w_{ij} \qquad \forall\, i \in \mathcal{N}\ \forall\, j \in \mathcal{N}\backslash\{i\}\ \forall\, k \in \mathcal{K} \qquad (3.10)$$

$$w_{ij} + y_{ij} \leq 1 \qquad \forall\ i \in \mathcal{N}\ \forall\, j \in \mathcal{N}\backslash\{i\} \qquad (3.11)$$

$$l_{ik} + t_{ij} - M(1 - x_{ijk}) \leq m_{jk} \qquad \forall\, i \in \mathcal{V}\ \forall\, j \in \mathcal{V}\backslash\{i\}\ \forall\, k \in \mathcal{K} \qquad (3.12)$$

$$m_{ik} + s_i \leq l_{ik} \qquad \forall\, i \in \mathcal{N}\ \forall\, k \in \mathcal{K} \qquad (3.13)$$

$$a_i + s_i \leq l_{ik} \qquad \forall\, i \in \mathcal{N}\ \forall\, k \in \mathcal{K} \qquad (3.14)$$

$$m_{ik} \leq b_i \qquad \forall\, i \in \mathcal{V}\ \forall\, k \in \mathcal{K} \qquad (3.15)$$

$$l_{0k} = 0 \qquad \forall\, k \in \mathcal{K} \qquad (3.16)$$

$$x_{ijk} \in \{0,1\} \qquad \forall\, i \in \mathcal{V}\ \forall\, j \in \mathcal{V}\backslash\{i\}\ \forall\, k \in \mathcal{K} \qquad (3.17)$$

$$x_{ik} \in \{0,1\} \qquad \forall\, i \in \mathcal{N}\ \ \forall\, k \in \mathcal{K} \qquad (3.18)$$

$$y_{ij} \in \{0,1\} \qquad \forall\, i \in \mathcal{N}\ \forall\, j \in \mathcal{N}\backslash\{i\} \qquad (3.19)$$

$$w_{ij} \in \{0,1\} \qquad \forall\, i \in \mathcal{N}\ \forall\, j \in \mathcal{N}\backslash\{i\} \qquad (3.20)$$

$$m_{ik} \geq 0 \qquad \forall\, i \in \mathcal{V}\ \forall\, k \in \mathcal{K} \qquad (3.21)$$

$$l_{ik} \geq 0 \qquad \forall\, i \in \mathcal{V}\ \forall\, k \in \mathcal{K} \qquad (3.22)$$

$$u_i \geq 0 \qquad \forall\, i \in \mathcal{V} \qquad (3.23)$$

Formula 3.1 is the objective function of the model which aims to minimize the sum over all arcs between nodes i and j that are visited in the solution, multiplied by the cost of the arc. The cost of the arc is equal to the Euclidean distance between nodes i and j. This is the objective function for all Solomon benchmark datasets and relatively simple objective function as only the total distance is minimized and the setup cost for vehicles for example is not considered.

Constraints 3.2 and 3.3 set that each node is arrived at and left exactly once. Constraint 3.4 is a flow constraint that forces that each node is arrived at and left from by the same vehicle. Constraints 3.5 and 3.6 ensure that the depot is arrived at and left exactly once by each vehicle.

Constraints 3.7 and 3.8 are Miler-Tucker-Zemlin (MTZ) subtour elimination constraints. 3.7 forces the MTZ index variable u to be larger for node j than that of node i if arc (i,j) is travelled by a vehicle. Constraint 3.8 sets the MTZ of the depot to 0, making it the starting point of each tour.

Constraint 3.9 is a capacity constraint per vehicle, limiting the sum of demands of the nodes that are serviced by a vehicle to the capacity of the vehicle.

Constraint 3.10 and 3.11 form the cannot-link constraint. The binary variable w_{ij} is an indicator of whether node i and j are serviced by the same vehicle. In constraint 3.16 the cannot-link constraint is enforced as the sum of w_{ij} and y_{ij} is limited to 1. Therefore, each node pair can either have a cannot-link constraint or can be serviced by the same vehicle. Node pairs determined to be a cannot-link pair in preprocessing cannot be serviced by the same vehicle.

Constraints 3.12 through 3.16 are time window constraints. Constraint 3.12 is a big M formulation that forces the arrival time at node j to be larger than the time the vehicle leaves node i plus the time it takes to travel from node i to j. The big M relaxation ensures that this constraint is exclusively enforced for arcs that are travelled by a vehicle.

Constraint 3.13 sets the time a vehicle leaves node i to be larger than the time the vehicle arrived at node i plus the service time of the node. Constraint 3.14 forces the time a vehicle leaves node i to be larger than the earliest arrival time of the node plus the service time of the node. The service time of the nodes in the Solomon benchmark is consistent for each dataset.

Constraint 3.15 forces the arrival time at node i to be earlier than the maximum arrival time for the node. Constraint 3.16 ensures that each vehicle leaves the depot at the start of the total amount of time.

Constraints 3.17 through 3.23 are non-zero and binary constraints for the variables. Constraint 3.17 sets the variable whether arc (i, j) is travelled by vehicle k to binary. Constraints 3.18 and sets whether nodes i and j are serviced by vehicle k to binary. Constraint 3.19 sets the indicator of whether nodes i and j are in the cannot link set to binary. Constraint 3.20 sets the indicator of whether nodes i and j are in the same cluster to binary. Constraint 3.21 and 3.22 force the arrival at and leaving from time for each node to be larger than or equal to 0. Constraint 3.23 ensures each MTZ index is larger than or equal to 0.

3.2 Cannot-link clustering

The cannot-link constraint was introduced as a component of COP K-means, a semi-supervised clustering variant of the regular unsupervised K-means algorithm. In COP K-means the algorithm is equipped with a cannot-link set and a must-link set, where datapoint pairs are subjected to either a cannot-link or a must-link constraint. The cannot-link constraint is introduced to ensure that certain pairs of datapoints are not clustered together, while the must-link constraint ensures that certain pairs are clustered together (Wagstaff et al., 2001). After initializing K centroids at random, the datapoints are clustered in random order while the constraints are adhered to. Datapoints containing conflicting must-link and cannot-link constraints are excluded from the clustering process. A variation of the COP K-means has been proposed for DBSCAN, COP DBSCAN, where datapoints may have a must-link or cannot-link constraint as in the COP K-means algorithm (Ruiz et al., 2010).

For this thesis the concept of the cannot-link constraint has been implemented into four different clustering algorithms. The K-means, DBSCAN, a sweep algorithm and MILP clustering algorithms are equipped with a capacity constraint and the cannot-link constraint. These proposed clustering algorithms allow for the consideration of time window constraints in the clustering phase through the addition of the cannot-link constraint. Whether a pair of nodes should be considered a cannot-link pair is decided in a preprocessing step that compares all node pair combinations. The decision to assign cannot-link to a node pair is based on the fraction of the shortest time window that the trip between the nodes is feasible. This preprocessing can be set to a more conservative fraction, where only impossible node pairs are excluded and a more progressive approach where more node pairs are added to the cannot-link set.

Deciding which pairs of nodes should be included in the cannot-link set is the essence of including time windows in the clustering phase of CFRS. In the COP K-means method a must-link and cannot-link set change the K-means algorithm from unsupervised to semi-supervised learning. The cannot-link set is implemented for VRP clustering applications as it allows for a preprocessing step where unsuitable node pairs are identified. This preprocessing step, however, needs to be defined carefully. The cannot-link constraint as formulated in section 3.1 in formulae 3.10 and 3.11 is a hard constraint and therefore makes certain node combination in a cluster impossible. The risk of adding a hard cannot-link constraint to the clustering method is that if the cannot-link constraints are not decided on carefully the optimal solution may be eliminated from the search space.

The exclusion of incompatible node pairs from clusters does not prevent larger combination errors from occurring. Suppose the combinations of nodes A and B, B and C and A and C do not cause infeasible clusters based on paired distances and time windows. However, the combination of the three nodes in a single route might cause a violation of time window constraints. This thesis is limited to node pairs, preprocessing more advanced node combinations is not considered.

The set of equations presented in formula 3.24 through 3.27 represent the preprocessing steps performed for each pair of nodes. The set of formulae is based on the absolute time window length of the two nodes and the Euclidean distance between them. Below the notations, some of which are repeated from the mathematical model, are presented.

Notations:

$\mathcal{N} = set\ of\ all\ nodes\ excluding\ the\ depot\ (1, 2, \dots, n)$

$i = node\ in\ set\ \mathcal{N}$

$n = number\ of\ nodes\ in\ the\ dataset$

$t_{ij} = Euclidean\ distance\ between\ node\ i\ and\ node\ j$

$a_i = start\ of\ the\ time\ window\ of\ node\ i$

$b_i = end\ of\ the\ time\ window\ of\ node\ i$

$s_i = service\ time\ of\ node\ i$

$tw_i = time\ window\ length\ of\ node\ i$

$tw_{ij} = shortest\ time\ window\ of\ nodes\ i\ and\ j$

$z_{ij} = absolute\ time\ of\ feasibility\ of\ arc\ (i, j)$

$p_{ij} = feasibility\ fraction\ of\ arc\ (i, j)$

$$tw_i = b_i - a_i \qquad \forall\, i \in \mathcal{N} \qquad (3.24)$$

$$tw_{ij} = \min\{tw_i, tw_j\} \qquad \forall\, i \in \mathcal{N}\ \forall\, j \in \mathcal{N}\backslash\{i\} \qquad (3.25)$$

$$z_{ij} = \min\{b_j - a_i - s_i - t_{ij},\ tw_{ij}\} \qquad \forall\, i \in \mathcal{N}\ \forall\, j \in \mathcal{N}\backslash\{i\} \qquad (3.26)$$

$$p_{ij} = \max\left\{0, \frac{z_{ij}}{tw}\right\} + \max\left\{0, \frac{z_{ji}}{TW}\right\} \qquad \forall\, i \in \mathcal{N}\ \forall\, j \in \mathcal{N}\backslash\{i\} \qquad (3.27)$$

Formula 3.24 determines the absolute time window length of node i and node j. The minimum time window length is determined with formula 3.25.

Formula 3.26 determines the absolute time length of the arcs (i, j) and (j, i) that is feasible. This indicates how much time is available for each direction of travel and is indicated with z_{ij} and z_{ji}.

In formula 3.27 the absolute viable route time is converted to a fraction relative to the shortest time window. This is done for both directions of the arc and its value is set to zero if it is negative,

as that means that the arc is impossible. The sum of both factors is p_{ij}, which is the fraction of the arcs (i,j) and (j,i) that is feasible relative to the shortest time window of nodes i and j.

This fraction is used to decide whether node pair (i,j) should be included in the cannot-link set. If p_{ij} is smaller or equal to a certain threshold, node pair (i,j) is added to the cannot-link set which will prevent the nodes from being clustered together. If p_{ij} is larger than the threshold the node pair is not considered as a cannot-link pair and nodes i and j can be clustered together. For conservative clustering the threshold is set to 0.001, where exclusively impossible node combinations are added to the cannot-link set. As the fractions of the shortest time window are summed, the highest theoretical value is 2. This would mean that both directions (i,j) and (j,i) are possible for at least time length tw_{ij}. A threshold of 0.2 implies that at least one direction must be possible for at least 20% of the shortest time window length tw_{ij}. The cannot-link set that is generated is used for clustering in the four different clustering algorithms.

3.3 K-means clustering

The proposed capacitated K-means algorithms with capacity and cannot-link constraints (cap-cl-K-means) is an expansion of the K-means algorithm. The capacity and cannot-link constraints transform the algorithm into a semi-supervised clustering algorithm. The pseudocode of the cap-cl-K-means is shown in Figure 3.1. The algorithm starts by choosing K random datapoints as the starting centroids. Iteratively nodes that are not clustered are added to the cluster with the closest centroid and the centroids are recalculated.

Algorithm 1 K-means

Let N be the set of all nodes and K be the number of desired clusters.
Select K random points as initial centroids from N and store them in C.
Let P be the initially empty set of clustered nodes.
while $P \neq N$ **do**
  Select random node n from $N \setminus P$.
  Calculate distances between n and each centroid $c \in C$, add to set D.
  Sort the pairs $(c, d(c,n))$ by ascending distance, add to set S.
  for each pair $(c,d) \in S$ **do**
    if capacity and cannot - link constraints are not violated for cluster k with centroid c **then**
      Assign node n to the cluster k with centroid c.
      $P = P \cup \{n\}$.
      Update centroid c_k as mean of nodes in the updated cluster k.
      break.

Figure 3.1 pseudocode for the cap-cl-K-means algorithm (Gocken & Yaktubay, 2019; Wagstaff et al., 2001)

Choosing the number of clusters for K-means is known to be a difficult decision. For the regular unsupervised algorithm the elbow method is the most common, which varies K to be a range of cluster numbers. Afterwards K is plotted against the within cluster sum of squares and the value of K where the graph visibly changes from high slope to low slope, the elbow, is chosen as K. For a VRP the within cluster sum of squares is not a solid indicator for K as limitations on capacity or time windows are not considered. In this thesis two values for K were considered. First, as the optimal solution for the datasets is known, K was set equal to the number of vehicles that

generated the optimal solution. Secondly, as the limiting factor for the R1, C1 and RC1 datasets is known to be time, a time estimation was made for each dataset as if it was a CVRP. This resulted in an estimation of the number of vehicles required to serve all nodes if time windows were not a limiting factor. On this base estimation additional vehicles were added based on the total amount of time, time window restrictiveness and time window density. This resulted in a higher number of clusters for all datasets in comparison to the optimal number of vehicles. The exact calculations can be inspected in section A.1 in the appendix. The results for K-means with an optimal number of clusters is indicated with K-means-opt and the results for the calculated number of vehicles is indicated with K-means-calc in this thesis.

The cap-cl-K-means algorithm contains two sections with a high degree of randomness. First, the centroids are chosen to be K random nodes from the dataset. This starts each iteration of the algorithm differently. Secondly, the nodes are clustered in a random order. An advantage of this high degree of randomness is that with more iterations a larger segment of the solution space may be explored. A potential negative effect may be that the results at a lower number of iterations might be worse than they would have been with a more organized approach. Due to this high degree of randomness the cap-cl-K-means algorithm was applied on each dataset multiple times after each result was routed and the solution quality was compared. As the computational requirements of routing depend heavily on the time window restrictiveness and density, the number of iterations decreases as the density and restrictiveness decrease. The number of loops for each dataset can be viewed in **T**able A.9

3.4 DBSCAN clustering

Density based clustering methods like DBSCAN, cluster based on the density of datapoints within a certain range. DBSCAN classifies datapoints as core points, border points or noise, based on the number of other datapoints that lie within a distance of Eps from it. If at least MinPts points are within distance Eps, the datapoint is considered a core point, if no other datapoints are within distance Eps a datapoint is considered noise. For this thesis the DBSCAN was outfitted with a capacity constraint and cannot-link constraint (cap-cl-DBSCAN). The proposed algorithm is based on the semi-supervised COP DBSCAN and capacitated DBSCAN algorithms. The pseudocode for the cap-cl-DBSCAN algorithm is presented in Figure 3.2.

Algorithm 2 DBSCAN

Let N be the set of all nodes, Eps be the radius threshold, and $MinPts$ be the minimum number of points required to form a dense region.
Determine the core nodes, denoted by set $C \subseteq N$, such that each node in C has at least $MinPts$ neighbors within distance Eps.
Let P be the set of clustered nodes, initially empty.
while $C \cap P \neq C$ **do**
    Select a random core node n from the set $C \setminus P$.
    Create a new cluster k and assign n to cluster k.
    $P = P \cup \{n\}$.
    Let N_n be the set of nodes within distance Eps from n.
    for each node $m \in N_n$ **do**
        if $m \in C$ and $m \notin P$ **then**
            Add m to cluster k.
            $P = P \cup \{m\}$.
            Let N_m be the set of nodes within distance Eps from m.
            $N_n = N_n \cup N_m$.
    while capacity or cannot-link constraint is violated for cluster k **do**
        if cannot-link constraint is violated **then**
            $V \leftarrow$ set of nodes in cluster k that violate cannot-link constraint.
            Select node $v \in V$ that is furthest from n.
        else
            Select node v in cluster k that is furthest from n.
        Remove node v from cluster k.
        $P = P \setminus \{v\}$.

Figure 3.2 pseudocode for the cap-cl-DBSCAN algorithm (Gocken & Yaktubay, 2019; Ruiz et al., 2010)

The Eps and MinPts settings for the dataset types are shown in Table A.4 in the appendix. As the DBSCAN chooses the first center node randomly, the clustering algorithm was applied multiple times per dataset, depending on the computational requirements for the routing. Out of the iterations the best solution was chosen. The number of loops that were used for each dataset can be found in Table A.9

3.5 Sweep algorithm

The sweep algorithm for a CVRP was expanded with the cannot-link constraint as described in section 3.3 (Peya et al., 2019). The sweep algorithm fills vehicles until the capacity of the vehicle is exceeded, after which the next vehicle is filled. The proposed sweep algorithm starts filling the first vehicle with the node furthest from the depot. Subsequently the nodes closest to this starting node are added to the vehicle if they do not violate the cannot link constraint. When the addition of the upcoming node exceeds the capacity, the process is started again for the next vehicle. The pseudocode of the sweep algorithm with capacity and cannot-link constraints (cap-cl-sweep) is presented in Figure 3.3.

Algorithm 3 Sweep

Let P be the set of clustered nodes.
Start first cluster.
while P does not contain all nodes from set N **do**
  Select node n that is the furthest from the depot from set $N \setminus P$.
  $P \leftarrow P \cup \{n\}$.
  while capacity constraint is not violated **do**
    Select node m from set $N \setminus P$ that is closest to n.
    if cannot-link constraint is not violated **then**
      Add node m to the cluster.
      $P \leftarrow P \cup \{m\}$.
  Start next cluster.

Figure 3.3 pseudocode for the cap-cl-sweep algorithm (Peya et al., 2019; Wagstaff et al., 2001)

3.6 MIP clustering

As the result quality of solutions to the Solomon benchmark is determined by the total distance travelled, the MIP clustering method minimizes the internal distance of clusters. The formulation is based on a MILP model that minimizes the internal distance of clusters which was used for clustering biological data (Nascimento et al., 2010). The MILP formulation is shown in formulae 3.28 through 3.38. The objective function (3.28) and cannot-link constraints (3.31 and 3.32) were non-linear in the implementation of the clustering algorithm and solved using the MIQP quadratic programming solver in GurobiPy. The non-linear model is presented in Section A.2 of the appendix. The run-time of the MIP model was limited to 15 minutes for R1 datasets and 5 minutes for C1 and RC1 datasets. The notations used for the MILP formulation of the clustering model are identical to those of the mathematical formulation presented in Section 3.1, and are shown below.

Notations:

$\mathcal{N} = set\ of\ all\ nodes\ excluding\ the\ depot\ (1, 2, \dots, n)$

$\mathcal{K} = set\ of\ all\ vehicles\ (1, 2, \dots, q)$

$i = node\ in\ set\ \mathcal{N}$

$k = vehicle\ in\ set\ \mathcal{K}$

$n = number\ of\ nodes\ in\ the\ dataset$

$q = number\ of\ vehicles\ in\ set\ \mathcal{K}$

$t_{ij} = eucliadean\ distance\ between\ node\ i\ and\ node\ j$

$x_{ijk} = binary\ variable, 1\ if\ arc\ (i, j)\ is\ travelled\ by\ vehicle\ k, 0\ if\ not$

$x_{ik} = binary\ variable, 1\ if\ node\ i\ is\ serviced\ by\ vehicle\ k, 0\ if\ not$

$w_{ij} = binary\ variable, 1\ if\ nodes\ i\ and\ j\ are\ served\ by\ the\ same\ vehicle, 0\ if\ not$

$y_{ij} = binary\ variable, 1\ if\ there\ is\ a\ cannot\ link\ constraint\ between\ node\ i\ and\ node\ j, 0\ if\ not$

$d_i = demand\ of\ node\ i$

$C = capacity\ of\ each\ vehicle\ k$

$$minimize \sum_{i=1}^{\mathcal{N}} \sum_{j \neq i}^{\mathcal{N} \setminus \{i\}} t_{ij} w_{ij} \quad (3.28)$$

s.t.

$$\sum_{k=1}^{\mathcal{K}} x_{ik} = 1 \qquad \forall\, i \in \mathcal{N} \qquad (3.29)$$

$$\sum_{i=1}^{\mathcal{N}} x_{ik} \geq 1 \qquad \forall\, k \in \mathcal{K} \qquad (3.30)$$

$$x_{ik} + x_{jk} - 1 \leq w_{ij} \qquad \forall\, i \in \mathcal{N}\ \forall\, j \in \mathcal{N} \setminus \{i\}\ \forall\, k \in \mathcal{K} \qquad (3.31)$$

$$w_{ij} + y_{ij} \leq 1 \qquad \forall\, i \in \mathcal{N}\ \forall\, j \in \mathcal{N} \setminus \{i\} \qquad (3.32)$$

$$\sum_{i=1}^{\mathcal{N}} x_{ik}\, d_i \leq C \qquad \forall\, k \in \mathcal{K} \qquad (3.33)$$

$$x_{ik} + x_{jk} - 1 \leq w_{ij} \qquad \forall\, i \in \mathcal{N}\ \forall\, j \in \mathcal{N} \setminus \{i\}\ \forall\, k \in \mathcal{K} \qquad (3.34)$$

$$w_{ij} + y_{ij} \leq 1 \qquad \forall\, i \in \mathcal{N}\ \forall\, j \in \mathcal{N} \setminus \{i\} \qquad (3.35)$$

$$x_{ik} \in \{0,1\} \qquad \forall\, i \in \mathcal{N}\ \forall\, k \in \mathcal{K} \qquad (3.36)$$

$$y_{ij} \in \{0,1\} \qquad \forall\, i \in \mathcal{N}\ \forall j \in \mathcal{N} \setminus \{i\} \qquad (3.37)$$

$$w_{ij} \in \{0,1\} \qquad \forall\, i \in \mathcal{N}\ \forall j \in \mathcal{N} \setminus \{i\} \qquad (3.38)$$

The objective function 3.30 minimizes the internal distance within clusters. Instead of only the arcs that are travelled, the sum of all arcs in the cluster is minimized.

Constraints 3.31 and 3.32 are assignment constraints. Constraint 3.31 limits the assignment of each node to one cluster. Constraint 3.32 ensures that each cluster contains at least one node.

Constraint 3.33 is the capacity constraint that limits the sum of demands of all nodes in a cluster to the capacity of a vehicle. Constraints 3.35 and 3.36 form the cannot-link constraint that prevents pairs of nodes that are deemed unfavorable during preprocessing from being clustered together.

Constraints 3.36, 3.37 and 3.38 are binary constraints that set the variables x_{ik}, y_{ij} and w_{ij} respectively to being binary.

Like the cap-cl-K-means algorithm from Section 3.3, the MIP model takes a number of clusters as an argument. To be able to make a fair comparison the MIP model was used for clustering twice for each dataset, once using the known optimal number of vehicles and once using the calculated number of vehicles as elaborated in appendix section A.1. The results of the MIP model with an optimal number of clusters is indicated with MIP-opt and the results for the calculated number of vehicles is indicated with MIP-calc in this thesis.

3.7 Routing

As the time windows have been considered during clustering and at least infeasible pairs have been prevented from being clustered together, depending on the strictness of the preprocessing, clusters can be routed using an exact MILP approach. A derivation of the mathematical model as shown in functions 3.1 through 3.23 was reformulated into code and solved in Python using the GurobiPy package. The capacity and cannot-link constraints were removed as these have been implemented in the clustering methods.

While infeasible node pairs were excluded from clusters, other node combinations may still cause infeasible clusters. Therefore, the routing was done with an increasing number of vehicles. The clusters were attempted to be routed with one vehicle and if that was infeasible, two vehicles and eventually three vehicles. This approach was chosen to reduce computational times and test the clustering quality of the different methods. In the case of clusters that were not routed within 20 minutes, the MIPGap of Gurobi was increased, generating a route with a lower confidence of being optimal.

3.8 Solomon dataset

The datasets used in this thesis are from the Solomon benchmark (Solomon, 1987). The Solomon dataset is a benchmark dataset from 1987 that has since been used to quantify the result quality for TWVRP approaches. The benchmark dataset consists of 56 problem instances divided over six classes: random 1 (R1), random 2 (R2), clustered 1 (C1), clustered 2 (C2), random-clustered 1 (RC1) and random-clustered 2 (RC2). Each instance contains a central depot and 100 nodes that are divided over the grid randomly, clustered or as a mix of both.

The R1, C1 and RC1 classes have a homogeneous fleet with a relatively low capacity. The R2, C2 and RC2 classes have a homogeneous fleet with a relatively high capacity, this allows for many nodes to be routed by a single vehicle in comparison to the R1, C1 and RC1 classes. As the routing method in this thesis is an exact MILP model, the R2, C2 and RC2 datasets were not considered to limit the computational requirements.

The R1, C1 and RC1 classes contain 12, 9 and 8 datasets respectively. In these datasets the nodes remain at their positions while the time windows are altered. The R1, C1 and RC1 datasets are plotted in Figure A.1, Figure A.2 and Figure A.3 respectively in the appendix. The time windows of the datasets vary in two dimensions, the time window density and the time window restrictiveness. Either the time window density is altered between instances or the time window restrictiveness is altered at 100% frequency.

The time window density of the instances are divided over 100, 75, 50 and 25%, restricting a percentage of the nodes to time windows with identical restrictiveness. The time window restrictiveness is altered at 100% time window density. The average time window length can span from 4% to 51% of the total amount of available time. In some instances the time window length is constant while in others the shortest time window may be as short as 8% of the total available time while the longest may be up to 83% of the total available time, this widest spread is found in instance R111 (Gendreau & Tarantilis, 2010; Solomon, 1987, 2005). The specific characteristics of the Solomon VRP instances can be viewed in Table A.1 in the appendix.

4 Results and discussions

The method of clustering may influence the quality of result of a CFRS approach to a VRP. Four different clustering methods have been evaluated on the TWVRP Solomon benchmark datasets. The R1, C1 and RC1 dataset classes each consisting of 12, 9 and 8 individual datasets respectively. The clustering methods that were used are K-means, DBSCAN, a sweep algorithm and MIP clustering. Each of these clustering algorithms has been equipped with a capacity constraint and a cannot-link constraint. For the comparison of clustering methods datasets were clustered with the capacity set to 200 as specified in the benchmark, the cannot-link constraint was set to 0.001, a conservative setting that solely excludes impossible node pairs from being clustered together.

For the evaluation of the influence of the implementation of the cannot-link constraint in the clustering methods, 8 individual datasets have been clustered and routed with different strictness levels during preprocessing. The datasets were chosen on their time window restrictiveness, density and the similarity between the dataset classes.

4.1 Cluster method performance

The overall performance of the different clustering methods on the benchmark datasets with a cannot-link strictness of 0.001 can be seen in Table 4.1. The rows contain the average percentage above optimality for the different clustering methods in the columns. The first row contains the average percentage aggregated over all datasets, the second row contains the average percentage for the R1 class datasets, the third row for the C1 class datasets and the fourth row for the RC1 class datasets. The individual results for all datasets can be seen in Table A.6 in the appendix.

The best overall performing clustering method is the MIP clustering method supplied with the optimal number of vehicles (MIP-opt). The second-best performer is DBSCAN followed by MIP with the calculated number of vehicles (MIP-calc), the sweep algorithm and the worst overall performance is by the K-means algorithms (K-means-opt and K-means-calc).

The variation in results of the different algorithms can be seen in Table 4.2 which displays the best performance, worst performance and overall standard deviation of the clustering methods. The highest difference in performance is found for the MIP-calc method, the lowest difference in performance is found for the sweep algorithm which also has the lowest standard deviation. The highest standard deviation is found for the DBSCAN algorithm.

The performance within different dataset classes differs from the total average. The MIP-opt clustering method performs best in both R1 and RC1 class datasets. It is only outperformed in C1 by DBSCAN but still found optimality in 7 out of 9 datasets. The MIP-calc performed slightly worse than MIP-opt in R1 and RC1. This is not unexpected as the estimated number of vehicles was more conservative resulting in a higher number of clusters and a higher number of vehicles. This is demonstrated severely in the C1 datasets where the difference in percentage above optimality between the two methods is 28.7%.

The DBSCAN algorithm performs extremely well in the C1 datasets, finding optimality 7 out of 9 datasets and finding a solution within 0.2% of optimality for the other two datasets. The DBSCAN algorithm performs third best in RC1, after MIP-opt and MIP-calc but ahead of the sweep and K-means algorithms. For R1 the only algorithm performing worse than DBSCAN is the K-means-calc

algorithm. The clear result quality degradation from C1 to RC1 to R1 is likely due to the lower presence of areas with a high node density on which the algorithm clusters.

The sweep algorithm performs the most consistently out of the clustering algorithms with a respective 25.1, 25.1 and 24.1 average percentage above optimality for R1, C1 and RC1 and the lowest difference in performance and standard deviation as shown in Table 4.2. This high consistency can be explained by the hierarchical approach of the sweeping algorithm.

The K-means algorithm has the overall worst performance with an average of 30.0 and 31.1 percent above optimality K-means-opt and K-means-calc respectively. Only for the R1 dataset class both K-means settings outperform DBSCAN and K-means-calc outperforms the sweep algorithm. While the K-means algorithm was the best performing algorithm in the reference literature, the performance of the K-means algorithm in this research is likely due to the high degree of randomness in the algorithm. The centroids are chosen randomly and clusters are filled randomly. The starting centroids could heavily influence the final clusters and a more constructive choice of the starting centroids may improve the consistency and quality of the results. In the current looping system, all iterations of clusters are routed and the best is chosen. Another improvement may be clustering many more times and choosing the best clustering result before routing.

Table 4.1 The solution percentage above optimal for the different clustering methods and dataset types with a CL strictness of 0.001

	KMEANS-OPT	KMEANS-CALC	MIP-OPT	MIP-CALC	DBSCAN	SWEEP
AVERAGE	30.0	31.1	13.9	23.6	17.3	24.9
R1	26.8	28.1	21.4	22.7	27.5	25.3
C1	37.8	41.8	1.4	30.1	0.0	25.1
RC1	25.9	23.7	16.6	17.5	21.4	24.1

Table 4.2 The spread and standard deviation of the clustering methods and dataset types with CL strictness of 0.001

	KMEANS-OPT	KMEANS-CALC	MIP-OPT	MIP-CALC	DBSCAN	SWEEP
MAX	56.0	57.2	33.6	51.7	40.4	36.1
MIN	16.8	14.8	0.0	0.8	0.0	10.6
STDEV	10.4	9.6	9.9	11.7	13.6	7.6

Table 4.3 shows which percentage of clusters for each clustering method could not be routed with a single vehicle. 29.9% up to 80.1% of the clusters could be routed with a single vehicle for MIP-opt, K-means-opt, DBSCAN and sweep algorithms for the R1 and RC1 datasets. Only for the C1 dataset all class all clusters could be routed with a single vehicle for MIP-opt, MIP-calc and DBSCAN.

Two methods of infeasibility may cause the prevention of routing a cluster with a single vehicle, "combination infeasibility" and "filling infeasibility". "Combination infeasibility" of a pair of nodes is prevented by the cannot-link constraint, if the combination of node i and node j in a cluster guarantee infeasibility, the cannot-link constraint withs strictness 0.001 prevents them from being clustered together. If node k is added to this equation, the paired combinations of node i, j and k may all be possible but the full combination of those three nodes in a single cluster may prevent routing with a single vehicle. This could occur with three but also four, five or more nodes that generate an impossible combination. This means that with a single vehicle one of the nodes cannot be reached before its latest possible arrival time.

The other method that might prevent a cluster from being routed by a single vehicle is the "filling infeasibility". The capacity of the vehicles is set to 200 as specified by the benchmark, however as the average demands of the nodes is 14.4, 17.9 and 17.1 for R1, C1 and RC1 respectively, the capacity is reached after 13.9, 11.2 and 11.7 nodes. For the R1 dataset where the percentage of clusters that could not be served by one vehicle is highest, 14 nodes served by one vehicle would mean a total service time of 140 on a total available time of 230. This means that 61% of a vehicles' time is spent as service time without any driving times or waiting times due to time window constraints. Even without time window constraints this could cause the vehicle to reach the depot after the full time has elapsed. This effect is suspected to be highest in the sweep algorithm, which fills clusters until the capacity has been reached which may therefore cause the high need for extra vehicles.

For the estimated number of clusters, which was calculated conservatively and was equal to or higher than the optimal number of vehicles for each dataset, the number of clusters that cannot be routed with one vehicle is drastically reduced. This reduction may be a result of the on average lower number of nodes in each cluster. This would reduce the chances of both the "combination infeasibility" and the "filling infeasibility" from occurring.

Table 4.3 The percentage of clusters that could not be routed with a single vehicle for the different clustering methods and dataset types with a CL strictness of 0.001.

	KMEANS-OPT	KMEANS-CALC	MIP-OPT	MIP-CALC	DBSCAN	SWEEP
AVERAGE	46.1	11.0	29.9	5.1	39.3	65.1
R1	49.6	11.6	44.6	7.6	58.3	80.1
C1	34.4	5.6	0.0	0.0	0.0	35.7
RC1	53.8	16.2	41.7	6.9	55.0	75.7

As the TWVRP is an NP-hard problem, finding the optimal solution may take extremely long, therefore heuristic solutions approaching optimality but within reasonable time can be considered functional as well. Figure 4.1 duration of each clustering method in seconds on a logarithmic scale as a function of the time window length. Figure 4.2 shows the duration in seconds on a logarithmic scale as a function of the time window density.

The K-means, DBSCAN and sweep algorithm calculation duration increase steeply with an increase in the average time window length and a decrease in time window density. This increase in

calculation time can be explained by the MILP routing method. As the time window length increases or the density decreases, the number of possible routes increases as well, which increases computational requirements. The MIP clustering methods show a more consistent duration as the clustering will take as long as the time limit allows. The MIP-opt shows an increase in duration at a higher time window length while the MIP-calc remains more constant. Table 4.3 shows that the number of clusters that need to be routed with multiple vehicles is reduced for MIP-calc, which reduces the computation times. This pattern is seen for the K-means-opt and K-means-calc methods as well.

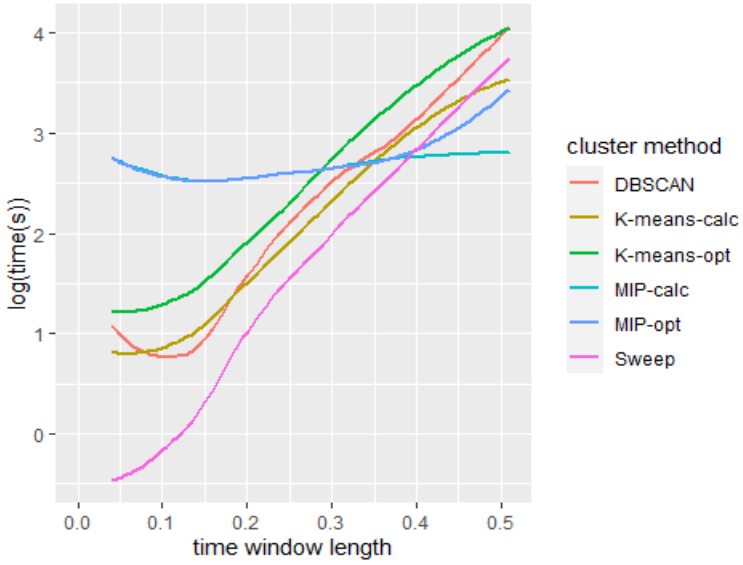

Figure 4.1 The duration of each clustering method solving the VRP with different time window lengths.

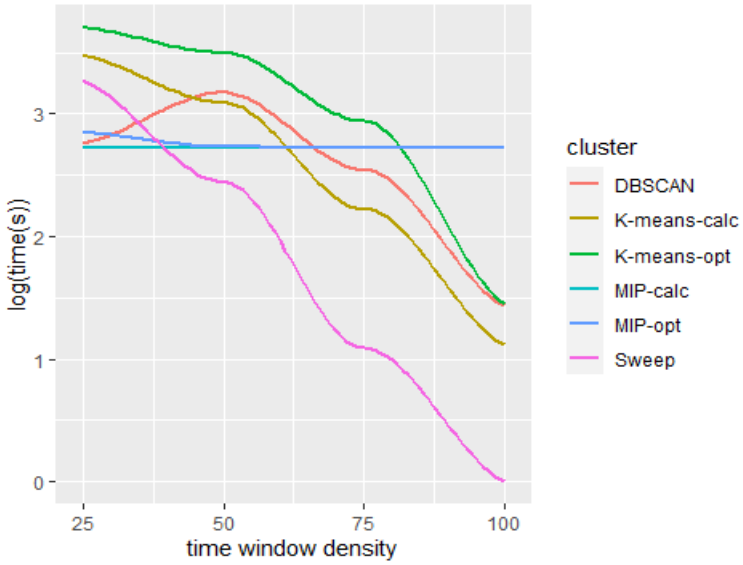

Figure 4.2 The duration of each clustering method solving the VRP with different time window densities.

4.2 Impact of cannot-link constraint

The cannot-link constraint which was proposed as a component of COP K-means semi supervised clustering has been included in the four different clustering methods used. The restrictiveness of the cannot-link constraint is dependent on the preprocessing step from chapter 3.3.

To quantify the influence of the cannot-link constraint on the quality of solution the strictness of the preprocessing has been modified to four distinct levels for 8 different datasets. These levels of restrictiveness are: no cannot-link constraint, 0.001, 0,2 and 0.5. The restrictiveness of 0.001 is extremely conservative, meaning that only impossible combinations are excluded from clusters. As the time window factor is added for both directions of the arc, a restrictiveness of 0.2 means that at least 20% of one of the arc directions should be feasible. The restrictiveness of 0.5 means that at least 50% of one of the directions should be feasible. The datasets that have been used all have a time window density of 100%. The average time window length was circa 4%, 13% or 20% of the total available time. The time windows with an average length of 20% varied in length within the dataset, those of 4% and 13% are constant. The full results are shown in Table A.8 in the appendix.

Figures 4.3, 4.4 and 4.5 show the influence of the cannot-link constraint on the result aggregated over the clustering methods for dataset classes R1, C1 and RC1 respectively. The range of the result quality, especially for the C type datasets, is high due to the aggregation over the different clustering methods as the overall best and worst performance of clustering algorithms was in the C1 dataset class.

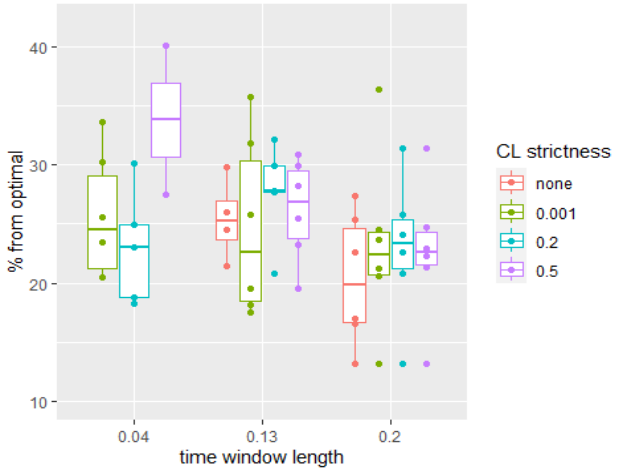

Figure 4.3 Percentage from optimality for different time window lengths at different CL restrictiveness levels for the R class datasets

Figure 4.3 shows the influence of preprocessing strictness and time window length on the performance of the CFRS approach. For the highly restrictive time windows no results were found for clustering without the cannot-link constraint as clusters that could not be routed with 3 vehicles were deemed infeasible. This indicates that the considering time window constraints in the clustering phase improves the quality of solution for the R class dataset with highly restrictive time windows. With a decrease in time window restrictiveness, the impact of the cannot-link constraint changes. For the dataset with an average time window length of 13% of the total available time, the exclusion of infeasible node pairs from clusters (CL 0.001) improves on the

approach without cannot-link constraint. A more progressive preprocessing step generates longer routes than the base-line clustering without cannot-link constraint. For the least restrictive time windows, the clustering with cannot link constraint performed worse than clustering without cannot-link constraint for all strictness settings.

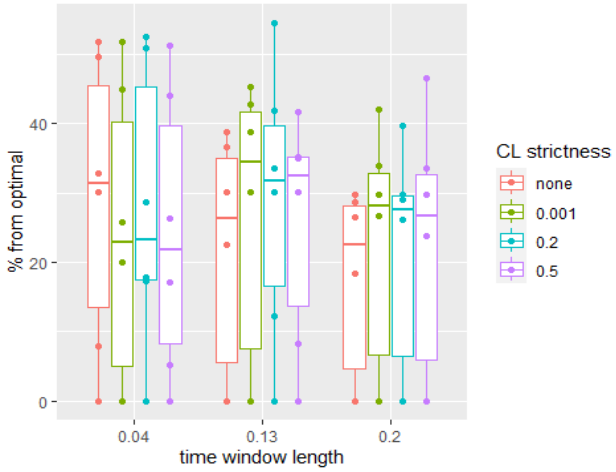

Figure 4.4 Percentage from optimality for different time window lengths at different CL restrictiveness levels for the C class datasets

Figure 4.4 shows the results of the CFRS approach with different preprocessing strictness settings (CL strictness). For the dataset with the most restrictive time windows the inclusion of the cannot-link constraint improves the average result of the CFRS approach in comparison to no cannot-link constraint. For both the medium and least restrictive datasets, the inclusion of the cannot link constraint decreased the performance of the clustering methods as it increased the total distance travelled by vehicles in the solutions.

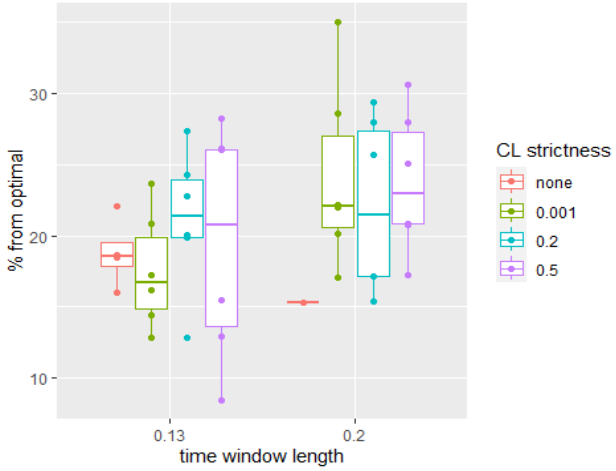

Figure 4.5 Percentage from optimality for different time window lengths at different CL restrictiveness levels for the RC class datasets

For the RC class there was no highly restrictive time window dataset with time windows of 4% of the total available time. Similar to the performance in the R class datasets, the clustering algorithms showed the best performance with a highly conservative cannot-link constraint for

the dataset with a medium restrictiveness. More progressive preprocessing reduced the average performance of the clustering algorithms. For the RC dataset with the least restrictive time windows only one solution was found without the cannot-link constraint, which is in contrast with the comparable datasets of the R and C class. This might be caused by specific characteristics of the dataset such as incompatible node pairs close to each other. The single feasible solution is better than those found with the cannot-link constraint included in the clustering phase, which is consistent with the other low-restrictive time window results.

The impact of the cannot-link constraint is highly problem specific and dependent on the time window restrictiveness and dispersion of the nodes on the grid. The influence of the implementation of the cannot link constraint is governed by the preprocessing step that decides on the cannot-link node pairs. In TWVRP instances with high time window restrictiveness and random dispersion of nodes the cannot-link constraint that excludes node pairs from being clustered together may be an improvement to a CFRS approach.

5 Conclusion, limitations and future research

Even though the vehicle routing problem has seen many approaches over the years and the current state-of-the-art approaches involve highly developed metaheuristics, CFRS is still an important segment of VRP methodology. The choice of clustering method has a major influence in the final quality of the result and a comparison of different methods may help others make this choice. A high-quality solution may reduce costs, greenhouse gas emissions or improve the working environment for drivers. For this research four different clustering methods have been evaluated based on benchmark VRPTW datasets. Overall, the MIP clustering was the most effective for the R1 and RC1 dataset classes of the Solomon benchmark, followed by DBSCAN which performed best on the C1 class datasets. The sweep algorithm had a lower quality of results than MIP clustering and DBSCAN. The least effective clustering method was K-means.

Based on the reference literature K-means was the most promising clustering method, however the high degree of randomness used in this adaptation may be the origin of the low result quality. An approach with a more constructive choice of starting centroids may improve on the results generated by the algorithm. The DBSCAN algorithm performed exceptionally well in the C1 category of datasets. As these datasets consist of high-density node groups, the consistency is not surprising. The DBSCAN algorithm was less effective in the R1 and RC1 type datasets, these are less densely scattered compared to C1. The DBSCAN algorithm was tested with one MinPts and Eps setting, exploring a wider range of settings might improve on the results for R1 and RC1. The sweep algorithm performed consistently having the lowest range of result quality and standard deviation. However, as the clusters were consistently filled up to the capacity without consideration if the number of nodes was feasible within the total amount of available time, few clusters could be routed with a single vehicle. Redefining the capacity constraint may improve the algorithm. The MIP clustering, based on within cluster distances, was the best performing clustering method. The contrast in results between an estimated number of clusters and the known optimal number of clusters was large. The determination of the number of clusters to make is a key decision that will dominate the quality of results for this clustering method.

The implementation of the cannot-link constraint in the clustering phase of CFRS for TWVRP is not decisively an improvement or diminishment to the method. Outside of high density, high restrictiveness time windows, the cannot-link constraint based on infeasible node pairs does not seem to perform better, or even slightly worse than these clustering methods without the cannot-link constraint. In the specific datasets with high density, high restrictiveness time windows, the cannot-link constraint has proven to be able to generate acceptable solutions where the clustering algorithms without cannot-link constraint could not.

In further CFRS research the implementation of the cannot-link constraint may be considered for high time window density and restrictiveness datasets. The preprocessing for the cannot-link decision must be done carefully. For general or non-categorized dataset applications the MIP clustering that minimizes the within cluster distance is the safest approach. Datasets with regional high-density areas may benefit from clustering with the DBSCAN algorithm. Implementations of the current sweep and K-means algorithm would require extensive improvements on the methods.

References

Armbrust, P., Maier, K., & Truden, C. (2022). Sweep Algorithms for the Vehicle Routing Problem with Time Windows. *Optimization and Learning 5th International Conference*, 135–144. https://doi.org/10.1007/978-3-031-22039-5

Berger, J., & Barkaoui, M. (2004). A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers and Operations Research*, *31*(12), 2037–2053. https://doi.org/10.1016/S0305-0548(03)00163-1

Boonsam, P., Suthikarnnarunai, N., & Chitphaiboon, W. (2011). Assignment Problem And Vehicle Routing Problem For An Improvement Of Cash Distribution. *Proceedings of the World Congress on Engineering and Computer Science*, *II*, 1–5.

Cömert, S. E., Yazgan, H. R., Kır, S., & Yener, F. (2018). A cluster first-route second approach for a capacitated vehicle routing problem: A case study. *International Journal of Procurement Management*, *11*(4), 399–419. https://doi.org/10.1504/IJPM.2018.092766

Cömert, S. E., Yazgan, H. R., Sertvuran, I., & Şengül, H. (2017). A new approach for solution of vehicle routing problem with hard time window: an application in a supermarket chain. *Sadhana - Academy Proceedings in Engineering Sciences*, *42*(12), 2067–2080. https://doi.org/10.1007/s12046-017-0754-1

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., & Soumis, F. (2002). 7. VRP with Time Windows. In P. Toth & D. Vigo (Eds.), *The Vehicle Routing Problem* (pp. 157–193). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718515.ch7

Cordeau, J.-F., Laporte, G., & Mercier, A. (2001). A unified tabu search algorithm for vehicle routing problems with soft time windows. *Journal of the Operational Research Society*, *52*, 928–936.

Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, *6*(1), 80–91. https://www.jstor.org/stable/2627477

Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, *11*(2), 109–124. https://doi.org/10.1002/NET.3230110205

Gambardella, L., Taillard, É. D., & Agazzi, G. (1999). MACS-VRPTW : a Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In *New Ideas in Optimization* (pp. 63–76). McGraw-Hill.

Garside, A. K., & Laili, N. R. (2019). A Cluster-First Route-Second Heuristic Approach to Solve The Multi-Trip Periodic Vehicle Routing Problem. *Industrial Engineering Journal: Journal of Science and Applied Industrial Engineering*, *20*(2), 172–181.

Gendreau, M., & Tarantilis, C. D. (2010). Solving large-scale vehicle routing problems with time windows: The state-of-the-art. *Centre Interuniversitaire de Recherche Sur Les Reseaux d'entreprise, La Logistique et Le Transport*, *4*, 1–45. https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2010-04.pdf

Gillett, B. E., & Miller, L. R. (1974). A Heuristic Algorithm for the Vehicle-Dispatch Problem. *Operations Research*, *22*(2), 340–349.

Gocken, T., & Yaktubay, M. (2019). Comparison of different clustering algorithms via genetic algorithm for VRPTW. *International Journal of Simulation Modelling*, *18*(4), 574–585. https://doi.org/10.2507/IJSIMM18(4)485

Han, J., Kamber, M., & Pei, J. (2012). Cluster Analysis: Basic Concepts and Methods. In *Data mining: Data mining concepts and techniques* (3rd ed., pp. 333–496). Elsevier. https://doi.org/10.1109/ICMIRA.2013.45

Homberger, J. (2000). *Verteilt-parallele Metaheuristiken zur Tourenplanung: Lösungsverfahren für das Standardproblem mit Zeitfensterrestriktionen*. Deutscher Universitätsverlag.

Homberger, J., & Gehring, H. (1999). Two Evolutionary Metaheuristics For The Vehicle Routing Problem With Time Windows. *INFOR*, *37*(3), 297–318. https://doi.org/10.1080/03155986.1999.11732386

Huang, M., & Hu, X. (2012). Large Scale Vehicle Routing Problem : an Overview. *International Journal of Innovative Computing, Information and Control*, *8*(8), 5809–5819.

Konstantakopoulos, G. D., Gayialis, S. P., & Kechagias, E. P. (2022). Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification. *Operational Research*, *22*(3), 2033–2062. https://doi.org/10.1007/s12351-020-00600-7

Le, T. D. C., Nguyen, D. D., Oláh, J., & Pakurár, M. (2022). Clustering algorithm for a vehicle routing problem with time windows. *Transport*, *37*(1), 17–27. https://doi.org/10.3846/transport.2022.16850

Li, H., & Lim, A. (2003). Local search with annealing-like restarts to solve the VRPTW. *European Journal of Operational Research*, *150*(1), 115–127. https://doi.org/10.1016/S0377-2217(02)00486-1

Lin, S., & Kernighan, B. W. (1971). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, *21*(2), 498–516.

Mester, D., Bräysy, O., & Dullaert, W. (2007). A multi-parametric evolution strategies algorithm for vehicle routing problems. *Expert Systems with Applications*, *32*(2), 508–517. https://doi.org/10.1016/j.eswa.2005.12.014

Nascimento, M. C. V., Toledo, F. M. B., & de Carvalho, A. C. P. L. F. (2010). Investigation of a new GRASP-based clustering algorithm applied to biological data. *Computers and Operations Research*, *37*(8), 1381–1388. https://doi.org/10.1016/j.cor.2009.02.014

Osman, I. H., & Kelly, J. P. (1997). Meta-Heuristics Theory and Applications. *Journal of the Operational Research Society*, *48*(6), 657–657. https://doi.org/10.1057/palgrave.jors.2600781

Peya, Z. J., Akhand, M. A. H., Sultana, T., & Hafizur Rahman, M. M. (2019). Distance based Sweep Nearest algorithm to solve Capacitated Vehicle Routing Problem. *International Journal of Advanced Computer Science and Applications*, *10*(10), 259–264. https://doi.org/10.14569/ijacsa.2019.0101036

Rochat, Y., & Taillard, É. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, *1*(1), 147–167. https://doi.org/10.1007/BF02430370

Rousseau, L. M., Gendreau, M., & Pesant, G. (2002). Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics*, *8*(1), 43–58. https://doi.org/10.1023/A:1013661617536

Ruiz, C., Spiliopoulou, M., & Menasalvas, E. (2010). Density-based semi-supervised clustering. *Data Mining and Knowledge Discovery*, *21*(3), 345–370. https://doi.org/10.1007/s10618-009-0157-y

Shalaby, M., Mohammed, A., & Kassem, S. (2021). Supervised fuzzy C-means techniques to solve the capacitated vehicle routing problem. *International Arab Journal of Information Technology*, *18*(Special issue 3), 452–463. https://doi.org/10.34028/iajit/18/3A/9

Shaw, P. (1997). *A new local search algorithm providing high quality solutions to vehicle routing problems*.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science*, *1520*, 417–431. https://doi.org/10.1007/3-540-49481-2_30

Shin, K., & Han, S.-Y. (2011). A Centroid-Based Heuristic Algorithm for the Capacitated Vehicle Routing Problem. *Computing and Informatics*, *30*(4), 721–732. https://www.cai.sk/ojs/index.php/cai/article/download/192/162/694

Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, *35*(2), 254–265. https://pubsonline.informs.org/doi/abs/10.1287/opre.35.2.254

Solomon, M. M. (2005). *VRPTW Benchmark Problems*. http://web.cba.neu.edu/~msolomon/problems.htm

Taillard, É. D., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J. Y. (1997). A hyrid tabu search for the vehicle routing problem with soft time windows. *Transportation Science*, 507-. https://doi.org/10.1287/ trsc.31.2.170

Toth, P., & Vigo, D. (2002). An Overview of Vehicle Routing Problems. In P. Toth & D. Vigo (Eds.), *The Vehicle Routing Problem* (1st ed., pp. 1–26). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718515.CH1

Voß, S. (2008). Metaheuristics. In C. Floudas & P. Pardalos (Eds.), *Encyclopedia of Optimization* (pp. 2061–2075). Springer, Boston, MA. https://doi.org/10.1007/978-0-387-74759-0_367

Wagstaff, K., Cardie, C., Rogers, S., & Schrödl, S. (2001). Constrained K-means Clustering with Background Knowledge. *International Conference on Machine Learning ICML*, 577–584. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.4624&rep=rep1&type=pdf

Zunic, E., Donko, D., Supic, H., & Delalic, S. (2020). Cluster-based approach for successful solving real-world vehicle routing problems. *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems, FedCSIS 2020*, *21*, 619–626. https://doi.org/10.15439/2020F184

A Appendix

Table A.1 Quantification of the Solomon VRP instances

SOLOMON	CAPACITY	TW %	MIN TW LENGTH	MAX TW LENGTH	AVG TW LENGHT	TMAX	MINTW /TMAX	MAXTW /TMAX	AVG(TW) /TMAX
R101	200	100	10	10	10	230	0.04	0.04	0.04
R102	200	75	10	10	10	230	0.04	0.04	0.04
R103	200	50	10	10	10	230	0.04	0.04	0.04
R104	200	25	10	10	10	230	0.04	0.04	0.04
R105	200	100	30	30	30	230	0.13	0.13	0.13
R106	200	75	30	30	30	230	0.13	0.13	0.13
R107	200	50	30	30	30	230	0.13	0.13	0.13
R108	200	25	30	30	30	230	0.13	0.13	0.13
R109	200	100	37	83	58.9	230	0.16	0.36	0.26
R110	200	100	23	177	86.5	230	0.10	0.77	0.38
R111	200	100	19	191	93.1	230	0.08	0.83	0.40
R112	200	100	73	166	117.6	230	0.32	0.72	0.51
C101	200	100	37	89	60.8	1236	0.03	0.07	0.05
C102	200	75	43	81	61.3	1236	0.03	0.07	0.05
C103	200	50	43	81	59.9	1236	0.03	0.07	0.05
C104	200	25	43	79	60.6	1236	0.03	0.06	0.05
C105	200	100	75	177	121.6	1236	0.06	0.14	0.10
C106	200	100	29	387	156.15	1236	0.02	0.31	0.13
C107	200	100	180	180	180	1236	0.15	0.15	0.15
C108	200	100	149	353	243.3	1236	0.12	0.29	0.20
C109	200	100	360	360	360	1236	0.29	0.29	0.29
RC101	200	100	30	30	30	240	0.13	0.13	0.13
RC102	200	75	30	30	30	240	0.13	0.13	0.13
RC103	200	50	30	30	30	240	0.13	0.13	0.13
RC104	200	25	30	30	30	240	0.13	0.13	0.13
RC105	200	100	10	120	54.3	240	0.04	0.50	0.23
RC106	200	100	60	60	60	240	0.25	0.25	0.25
RC107	200	100	41	155	88.2	240	0.17	0.65	0.37
RC108	200	100	27	180	112.3	240	0.11	0.75	0.47

(Solomon, 1987, 2005)

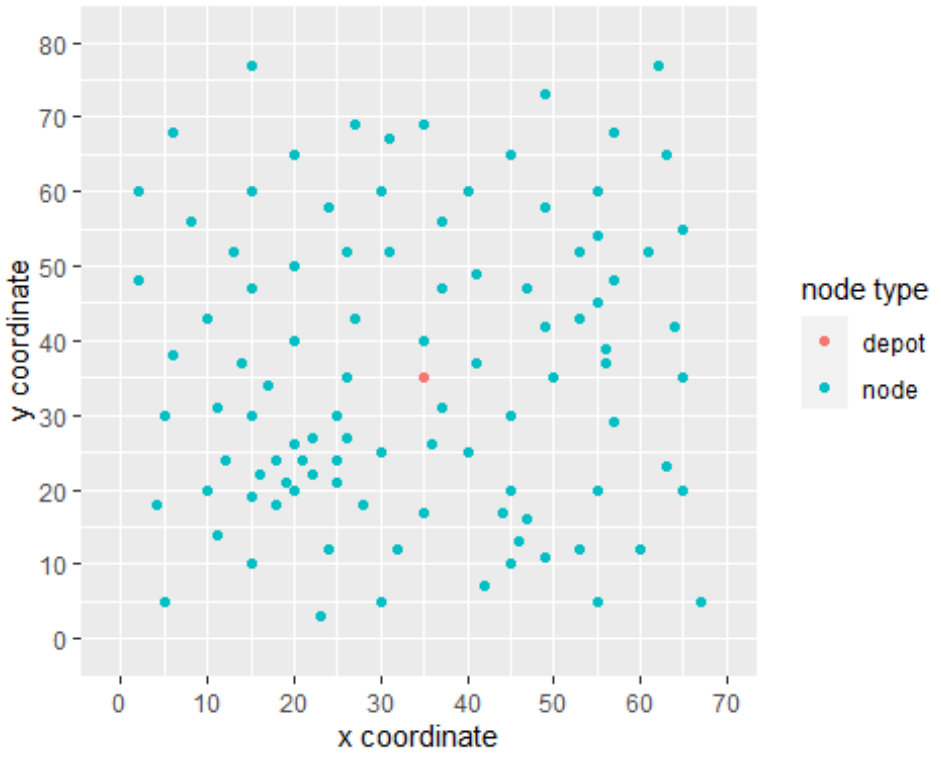

Figure A.1 The depot and nodes of the R1 type datasets of the Solomon Benchmark

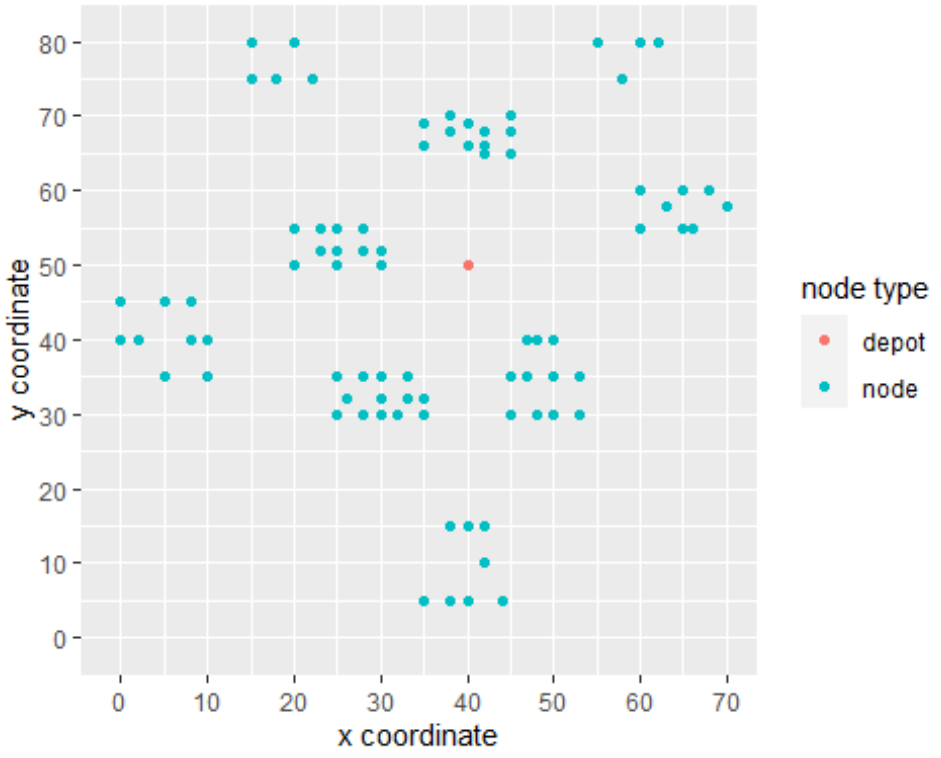

Figure A.2 The depot and nodes of the C1 type datasets of the Solomon Benchmark

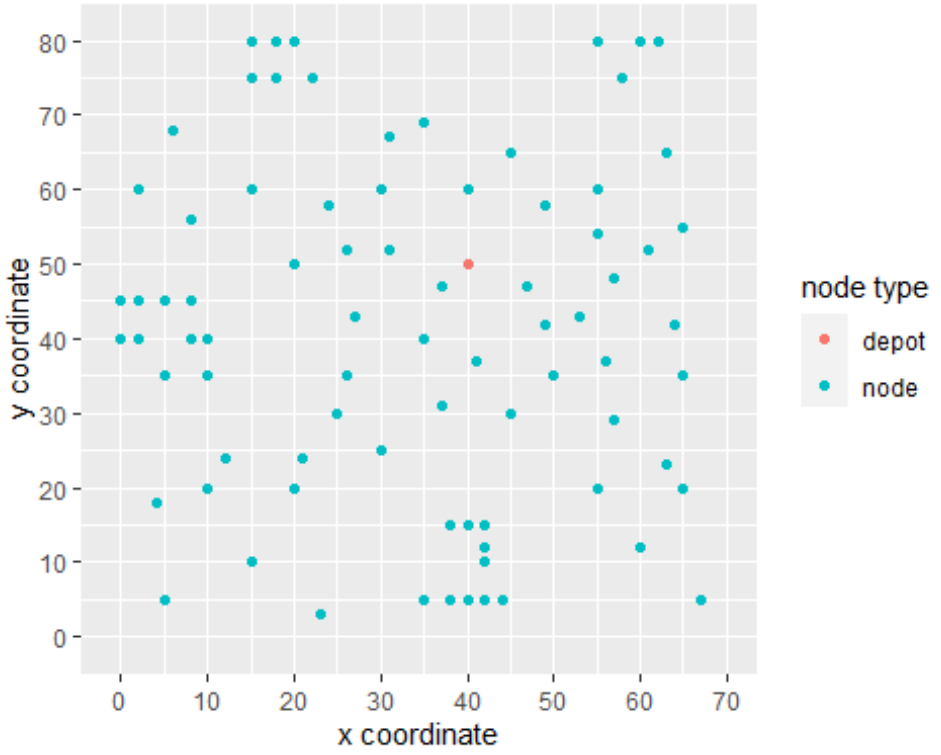

Figure A.3 The depot and nodes of the RC1 type datasets of the Solomon Benchmark

A.1 Calculation of the number of vehicles

Time is known to be the limited resource in the R1, C1 and RC1 datasets. Determining how many nodes could be serviced with the available total time taking into account the service time per node and average driving time. A factor can be added based on the time window density and time window tightness.

$$V_b = \frac{n}{t_{total}/(t_{service}+t_{drive})} \quad \text{(A.1)}$$

$$F = TW_{density}{}^2 * \left(1 - TW_{average}\right)^2 * \frac{t_{total}}{10} * n \quad \text{(A.2)}$$

$$V_{data} = \frac{V_b * t_{total}+F}{t_{total}} \quad \text{(A.3)}$$

Table A.2 Vb determination per dataset type

DATASETS	T_{TOTAL}	T_{DRIVE}	$T_{SERVICE}$	N	V_B
R1	230	20	10	100	13.04
C1	1236	20	90	100	8.89
RC1	240	20	10	100	12.5

Table A.3 Vehicle number per dataset

DATASET	V_B	$TW_{DENSITY}$	$TW_{AVERAGE}$	F	V_{DATA}	$V_{OPTIMAL}$
R101	13	1	0.04	2119.68	22	19
R102	13	0.75	0.04	1192.32	18	17

R103	13	0.50	0.04	529.92	15	13
R104	13	0.25	0.04	132.48	14	9
R105	13	1	0.13	1740.87	21	14
R106	13	0.75	0.13	979.239	17	12
R107	13	0.50	0.13	435.218	15	10
R108	13	0.25	0.13	108.804	13	9
R109	13	1	0.26	1259.48	18	11
R110	13	1	0.38	884.12	17	10
R111	13	1	0.40	828	17	10
R112	13	1	0.51	552.23	15	9
C101	9	1	0.05	11154.9	18	10
C102	9	0.75	0.05	6274.63	14	10
C103	9	0.50	0.05	2788.73	11	10
C104	9	0.25	0.05	697.181	10	10
C105	9	1	0.10	10011.6	17	10
C106	9	1	0.13	9355.28	17	10
C107	9	1	0.15	8930.1	16	10
C108	9	1	0.20	7910.4	15	10
C109	9	1	0.29	6230.68	14	10
RC101	12	1	0.13	1816.56	20	14
RC102	12	0.75	0.13	1021.82	16	12
RC103	12	0.50	0.13	454.14	14	11
RC104	12	0.25	0.13	113.535	12	10
RC105	12	1	0.23	1422.96	18	13
RC106	12	1	0.25	1350	18	11
RC107	12	1	0.37	952.56	16	11
RC108	12	1	0.47	674.16	15	10

Table A.4 Eps and MinPts settings for the DBSCAN algorithm

DATASET	MINPTS	EPS
R1	3	15
C1	3	10
RC1	3	15

Table A.5 Optimal distances, vehicles and references for the Solomon benchmark

DATASET	OPTIMAL	NV OPTIMAL	REFERENCE
R101	1650.80	19	(Homberger, 2000)
R102	1486.12	17	(Rochat & Taillard, 1995)
R103	1292.68	13	(Li & Lim, 2003)
R104	1007.31	9	(Mester et al., 2007)
R105	1377.11	14	(Rochat & Taillard, 1995)
R106	1252.03	12	(Mester et al., 2007)
R107	1104.66	10	(Shaw, 1997)
R108	960.88	9	(Berger & Barkaoui, 2004)
R109	1194.73	11	(Homberger & Gehring, 1999)
R110	1118.84	10	(Mester et al., 2007)

R111	1096.72	10	(Rousseau et al., 2002)
R112	982.14	9	(Gambardella et al., 1999)
C101	828.94	10	(Rochat & Taillard, 1995)
C102	828.94	10	(Rochat & Taillard, 1995)
C103	828.06	10	(Rochat & Taillard, 1995)
C104	828.78	10	(Rochat & Taillard, 1995)
C105	828.94	10	(Rochat & Taillard, 1995)
C106	828.94	10	(Rochat & Taillard, 1995)
C107	828.94	10	(Rochat & Taillard, 1995)
C108	828.94	10	(Rochat & Taillard, 1995)
C109	828.94	10	(Rochat & Taillard, 1995)
RC101	1696.95	14	(Taillard et al., 1997)
RC102	1554.75	12	(Taillard et al., 1997)
RC103	1261.67	11	(Shaw, 1998)
RC104	1135.48	10	(Cordeau et al., 2001)
RC105	1629.44	13	(Berger & Barkaoui, 2004)
RC106	1424.73	11	(Berger & Barkaoui, 2004)
RC107	1230.48	11	(Shaw, 1997)
RC108	1139.82	10	(Taillard et al., 1997)

Table A.6 Results of the clustering algorithms with CL strictness 0.001

DATASET	KMEANS-NV-OPT	KMEANS-NV-CALC	MILP-NV-OPT	MILP-NV-CALC	DBSCAN	SWEEP	OPTIMAL
R101	2037.23	1989.34	2205.36	2073.15	1989.51	2149.14	1650.80
R102	1735.10	1790.64	1688.04	1733.21	1959.79	2012.43	1486.12
R103	1563.47	1492.64	1450.72	1430.47	1679.72	1610.86	1292.68
R104	1353.45	1341.99	1186.72*	1198.06	1295.43*	1317.02*	1007.31
R105	1645.96	1815.85	1732.02	1869.03	1617.70	1626.82	1377.11
R106	1514.35	1600.61	1461.00	1691.12	1447.57	1488.86	1252.03
R107	1359.30	1419.51	1399.54	1287.42	1386.12	1392.20*	1104.66
R108	1406.37	1249.63	1196.73	1169.04	1349.12*	1291.48*	960.88
R109	1440.51	1629.27	1476.73	1488.05	1448.52	1352.51	1194.73
R110	1396.76	1438.15	1340.24	1356.80	1544.88	1295.07	1118.84
R111	1403.92	1446.45	1259.16	1370.08	1320.67	1347.99*	1096.72
R112	1408.11*	1296.58	1256.13*	1192.39	1378.96*	1314.91*	982.14
C101	1043.22	1201.72	828.94	1257.41	828.94	994.24	828.94
C102	1137.65	1183.59	828.94	1044.88	828.94	929.94	828.94
C103	1169.16	1160.99	828.94	869.26	828.94	959.73	828.06
C104	1077.00	1303.24	835.79	835.79	828.94	1128.23*	828.78
C105	1074.62	1099.93	828.94	1178.54	828.94	1047.10	828.94
C106	1293.27	1188.04	925.51	1254.44	828.94	1052.76	828.94
C107	1204.67	1182.49	828.94	1150.45	828.94	1078.23	828.94
C108	1177.39	1109.70	828.94	1049.40	828.94	1075.24	828.94
C109	1102.08	1147.27	828.94	1067.92	828.94	1066.97	828.94
RC101	2099.21	1970.61	1914.82	2049.91	1941.29	1988.89	1696.95
RC102	1825.78	1785.59	1729.35	1773.33	1794.60	1720.05	1554.75
RC103	1773.35	1673.81	1517.27	1447.59	1625.39	1677.63*	1261.67

RC104	1597.98	1390.31	1307.91	1309.41	1502.63	1502.15	1135.48
RC105	1958.11	1990.63	1987.28	1907.71	2095.97	2200.59	1629.44
RC106	1694.87	1809.73	1680.37	1692.11	1592.95	1680.56*	1424.73
RC107	1509.04	1536.00	1387.93	1511.00	1466.77	1493.87*	1230.48
RC108	1404.69	1477.65	1375.86*	1330.51	1376.48	1427.42*	1139.82

Routes noted with an asterisk * contain 1 or more clusters that were not routed within 20 minutes and therefore the MIPGap constraint, or the certainty of optimality was reduced.

Table A.7 number of vehicles required to route above optimal.

DATASET	KMEANS-NV-OPT	KMEANS-NV-CALC	MILP-NV-OPT	MILP-NV-CALC	DBSCAN	SWEEP	NV OPTIMAL
R101	6	5	9	8	7	10	19
R102	4	4	3	4	9	7	17
R103	3	4	3	3	9	7	13
R104	6	6	5	5	6	8	9
R105	7	12	9	9	5	4	14
R106	5	8	4	10	4	6	12
R107	7	8	7	6	6	6	10
R108	6	5	4	4	6	6	9
R109	5	8	6	7	6	5	11
R110	5	8	5	7	9	6	10
R111	6	8	3	7	5	6	10
R112	6	7	4	6	7	6	9
C101	4	8	0	8	0	4	10
C102	3	5	0	4	0	2	10
C103	1	1	0	1	0	0	10
C104	0	3	0	0	0	0	10
C105	4	7	0	7	0	5	10
C106	6	7	0	7	0	5	10
C107	7	7	0	6	0	6	10
C108	4	5	0	5	0	6	10
C109	2	5	0	4	0	3	10
RC101	10	9	7	8	7	8	14
RC102	6	7	5	7	6	5	12
RC103	6	6	4	4	6	6	11
RC104	7	4	3	3	7	6	10
RC105	7	10	7	7	9	10	13
RC106	6	10	6	7	5	5	11
RC107	4	6	3	5	5	4	11
RC108	4	6	4	5	5	5	10

Table A.8 Results of clustering methods with different CL constraint restrictiveness levels

DATASET	KMEANS-NV-OPT	KMEANS-NV-CALC	MILP-NV-OPT	MILP-NV-CALC	DBSCAN	SWEEP	OPTIMAL
R101_NO	Inf	Inf	Inf	Inf	Inf	Inf	1650.80
R101_0.01	2037.23	1989.34	2205.36	2073.15	1989.51	2149.14	1650.80

R101_0.2	Inf	1951.40	2148.31	2031.14	1960.42	2062.84	1650.80
R101_0.5	Inf	Inf	Inf	Inf	2313.03	2104.12	1650.80
C101_NO	1240.43	1100.44	828.94	1258.52	895.08	1078.23	828.94
C101_0.01	1043.22	1201.72	828.94	1257.41	828.94	994.24	828.94
C101_0.2	1066.45	1263.25	972.53	1250.09	828.94	976.22	828.94
C101_0.5	872.04	1194.06	1047.07	1253.54	828.94	971.48	828.94
R105_NO	1671.90	1786.99	1735.07	1714.17	Inf	Inf	1377.11
R105_0.01	1645.96	1815.85	1732.02	1869.03	1617.70	1626.82	1377.11
R105_0.2	1663.20	1820.35	1757.97	1789.16	1759.95	Inf	1377.11
R105_0.5	1697.03	1802.02	1727.10	1765.22	1646.22	1789.36	1377.11
C107_NO	1015.12	1132.44	828.94	1150.45	828.94	1078.23	828.94
C107_0.01	1204.67	1182.49	828.94	1150.45	828.94	1078.23	828.94
C107_0.2	1280.53	1106.22	1175.26	828.94	929.94	1078.23	828.94
C107_0.5	1120.69	1118.65	1173.64	828.94	897.13	1078.23	828.94
RC101_NO	2013.14	2071.15	1967.95	2009.49	Inf	Inf	1696.95
RC101_0.01	2099.21	1970.61	1914.82	2049.91	1941.29	1988.89	1696.95
RC101_0.2	2036.78	2083.91	1914.82	2034.21	2161.17	2109.56	1696.95
RC101_0.5	2176.02	2138.58	1916.30	2139.76	1959.83	1839.81	1696.95
R109_NO	1398.20	1521.88	1497.60	1464.48	1392.60	1352.51	1194.73
R109_0.01	1440.51	1629.27	1476.73	1488.05	1448.52	1352.51	1194.73
R109_0.2	1442.70	1570.09	1502.06	1482.33	1464.74	1352.51	1194.73
R109_0.5	1448.89	1569.21	1469.07	1461.50	1490.64	1352.51	1194.73
C108_NO	981.65	1066.91	828.94	1048.88	828.94	1075.24	828.94
C108_0.01	1177.39	1109.70	828.94	1049.40	828.94	1075.24	828.94
C108_0.2	1157.20	1044.95	828.94	1070.10	828.94	1075.24	828.94
C108_0.5	1215.28	1025.51	828.94	1107.56	828.94	1075.24	828.94
RC105_NO	1878.35	Inf	Inf	Inf	Inf	Inf	1629.44
RC105_0.01	1958.11	1990.63	1987.28	1907.71	2095.97	2200.59	1629.44
RC105_0.2	1909.25	2084.93	1879.71	1908.64	2048.43	2107.90	1629.44
RC105_0.5	1968.25	2037.43	1969.13	1910.07	2128.34	2085.71	1629.44

Table A.9 calculation lengths and number of loops per clustering method

DATASET	LOOPS KMEANS OPT	LOOPS KMEANS CALC	LOOPS DBSCAN	KMEANS-NV-OPT	KMEANS-NV-CALC	MILP-NV-OPT	MILP-NV-CALC	DBSCAN	SWEEP
R101	50	50	50	19.19	17.32	910.49	913.91	18.2	0.38
R102	50	50	50	280.67	255.79	909.09	910.3	1062.08	10.92
R103	30	20	30	1437.02	888.61	909.27	909.8	6181.75	110.24
R104	2	10	2	5063.85	2235.56	2147.07	909.78	8118.63	5012.51
R105	50	50	50	63.22	24	908.65	911	152.12	1.74
R106	50	50	30	2703.94	472.13	908.78	909.94	1846.41	34.57
R107	5	20	1	4450.38	1843.46	961.34	910.6	2044.11	2376.55

R108	2	10	2	1406.37	6108.03	1196.73	908.54	1349.11	1291.48
R109	5	10	2	239.98	49.24	909.02	911.14	180.88	14.03
R110	5	10	2	5381.15	1017.84	971.31	912.28	1830.38	1059.19
R111	2	10	2	1140.31	889.56	920.17	911.73	2050	1353.58
R112	2	10	2	11008.83	2817.67	3945.75	909.78	15278.1	8939.68
C101	50	10	50	15.34	2.18	305.62	310.5	8.45	0.38
C102	50	10	50	1298.36	20.23	305.81	307.79	30.15	1.17
C103	30	10	10	3586.16	1231.98	306.31	306.37	358.67	69.63
C104	10	10	2	4542.48	2427.51	305.96	306.5	1.46	725.68
C105	30	20	10	15.09	5.64	306.31	308.89	2.36	0.52
C106	30	20	10	25.69	7.09	307.04	310.29	2.62	1.37
C107	30	20	10	18.7	5.99	305.68	310.22	2.4	0.58
C108	30	20	10	46.92	18.91	305.7	309.52	7.11	2.57
C109	30	20	10	242.57	133.54	306.44	308.86	32.51	13.78
RC101	30	50	10	35.06	32.44	309.06	311.95	22.42	4.06
RC102	30	50	10	596.24	323.36	310.24	312.23	254.22	50.76
RC103	20	20	5	4375.7	1186.87	314.11	311.19	1121.78	333.74
RC104	10	10	3	20648.46	2402.54	315.78	308.45	6973.8	2552.95
RC105	20	20	5	387.71	180.34	312.57	313.48	317.69	112.12
RC106	20	20	5	172.91	96.07	315.34	309.79	545.1	244.82
RC107	20	20	5	2787.78	930.98	310.54	310.13	775.93	313.01
RC108	10	20	3	8435.16	3585.67	739.8	310.75	2676.44	903.55

Table A.10 the length of the cannot-link sets with different thresholds for the explored datasets.

SOLOMON	AVG(TW)	CL0.001	CL0.2	CL0.5
R101	0.04	1861	1961	2116
C101	0.05	628	722	890
R105	0.13	950	1215	1636
C107	0.15	0	33	238
RC101	0.13	1545	1795	2238
R109	0.26	181	398	901
C108	0.20	0	2	8
RC105	0.23	898	1041	1292

A.2 Non-linear MIQP clustering model

Notations:

$\mathcal{N} = set\ of\ all\ nodes\ excluding\ the\ depot\ (1, 2, \dots, n)$

$\mathcal{K} = set\ of\ all\ vehicles\ (1, 2, \dots, q)$

$i = node\ in\ set\ \mathcal{N}$

$k = vehicle\ in\ set\ \mathcal{K}$

$n = number\ of\ nodes\ in\ the\ dataset$

$q = number\ of\ vehicles\ in\ set\ \mathcal{K}$

$t_{ij} = eucliadean\ distance\ between\ node\ i\ and\ node\ j$

$x_{ijk} = binary\ variable, 1\ if\ arc\ (i,j)\ is\ travelled\ by\ vehicle\ k, 0\ if\ not$

$x_{ik} = binary\ variable, 1\ if\ node\ i\ is\ serviced\ by\ vehicle\ k, 0\ if\ not$

$y_{ij} = binary\ variable, 1\ if\ there\ is\ a\ cannot\ link\ constraint\ between\ node\ i\ and\ node\ j, 0\ if\ not$

$d_i = demand\ of\ node\ i$

$C = capacity\ of\ each\ vehicle\ k$

$$minimize \sum_{i=1}^{\mathcal{N}} \sum_{j \neq i}^{\mathcal{N}\backslash\{i\}} t_{ij} \sum_{k=1}^{\mathcal{K}} x_{ik} x_{jk} \quad (A.1)$$

s.t.

$$\sum_{k=1}^{\mathcal{K}} x_{ik} = 1 \qquad \forall\ i\ \in \mathcal{N} \qquad (A.2)$$

$$\sum_{i=1}^{\mathcal{N}} x_{ik} \geq 1 \qquad \forall\ k\ \in \mathcal{K} \qquad (A.3)$$

$$\sum_{i=1}^{\mathcal{N}} x_{ik}\, d_i \leq C \qquad \forall\ k\ \in \mathcal{K} \qquad (A.4)$$

$$x_{ik} x_{jk} y_{ij} = 0 \qquad \forall\ i \in \mathcal{N}\ \forall j \in \mathcal{N}\backslash\{i\}\ \forall k \in \mathcal{K} \qquad (A.5)$$

$$x_{ik} \in \{0,1\} \qquad \forall\ i \in \mathcal{N}\ \forall\ k \in \mathcal{K} \qquad (A.6)$$

$$y_{ij} \in \{0,1\} \qquad \forall\ i \in \mathcal{N}\ \forall j \in \mathcal{N}\backslash\{i\} \qquad (A.7)$$

A.3 Licenses

Behrouz-Babaki/COP-Kmeans: MIT License

Gurobi: Academic License

A.4 Specifications hardware

Processor	Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz
Installed RAM	16.0 GB (15.9 GB usable)
System type	64-bit operating system, x64-based processor
Edition	Windows 10 Education

Version	22H2
OS build	19045.2728
Gurobi	Gurobi 10.0.1
Python	Python 3.10 (64 bit)
PyCharm	Pycharm Community edition 2022.2.2