DEPARTMENT OF ENGINEERING MANAGEMENT

A two-level Variable Neighbourhood Search for the Euclidean Clustered Vehicle Routing Problem

Christof Defryn & Kenneth Sörensen

UNIVERSITY OF ANTWERP
Faculty of Applied Economics

City Campus
Prinsstraat 13, B.226
B-2000 Antwerp
Tel. +32 (0)3 265 40 32
Fax +32 (0)3 265 47 99
www.uantwerpen.be

FACULTY OF APPLIED ECONOMICS

DEPARTMENT OF ENGINEERING MANAGEMENT

A two-level Variable Neighbourhood Search for the Euclidean Clustered Vehicle Routing Problem

Christof Defryn & Kenneth Sörensen

RESEARCH PAPER 2015-002
JANUARY 2015

University of Antwerp, City Campus, Prinsstraat 13, B-2000 Antwerp, Belgium
Research Administration – room B.226
phone: (32) 3 265 40 32
fax: (32) 3 265 47 99
e-mail: joeri.nys@uantwerpen.be

D/2015/1169/002

A two-level Variable Neighbourhood Search for the Euclidean Clustered Vehicle Routing Problem

Christof Defryn [*†] Kenneth Sörensen [†]

January 2015

Abstract

In this paper, a metaheuristic approach is presented to solve the Clustered Vehicle Routing Problem (CluVRP). The CluVRP, in which customers are grouped into predefined clusters, can be seen as a generalisation of the classical Capacitated Vehicle Routing Problem (CVRP). When serving all these customers with a given fleet of vehicles it should be ensured that clients belonging to the same cluster are served by one vehicle, sequentially in the same path (CluVRP with hard cluster constraints). In a second phase, these constraints will be relaxed as we will define the CluVRP with soft cluster constraints. The proposed metaheuristic approach tries to find the optimal solution for both problems by combining two variable neighbourhood search algorithms, exploring the distribution area at two different levels. The algorithm is tested on different benchmark instances from the literature with up to 484 nodes, obtaining high quality solutions.

Keywords: Clustered Vehicle Routing Problem, Variable Neighbourhood Search, metaheuristics

[*]corresponding author. Email: christof.defryn@uantwerpen.be

[†]University of Antwerp, Departement of Engineering Management, ANT/OR - Operations Research Group

1 Introduction and Literature review

Introduced by Dantzig and Ramser (1959), the Vehicle Routing Problem (VRP) is one of the well known and widely studied problems in the operations research community. Many variants of the standard VRP-formulation have been proposed and solved during the last decades. In this paper, we will focus on the *Clustered Vehicle Routing Problem (CluVRP)*, which can be seen as a traditional capacitated vehicle routing problem (CVRP) in which the customers (nodes) are partitioned into predefined clusters. In the *strict* sense — with *hard clustering constraints* — all customers belonging to the same cluster should be visited by the same vehicle, *linked sequentially in the same path.*

The idea of client clustering is introduced by Chisman (1975) when defining the *clustered travelling salesman problem.* Here, an optimal Hamiltonian path is to be found in which all clients are visited exactly once. These clients, however, are assigned to a set of predefined clusters and an extra constraint is imposed, saying that all clients belonging to the same cluster are to be served sequentially in the same path. The main contributions regarding the clustered travelling salesman problem consist of a Tabu Search heuristic (Laporte et al., 1997) and Genetic algorithms (Ding et al., 2007; Potvin and Guertin, 1996). Besides the number of vehicle routing applications, Laporte et al. (2002) identified a whole range of alternative fields of application for the clustered travelling salesman problem, such as manufacturing (machine scheduling, plate cutting, optimisation of resource usage in a production process), IT (disk fragmentation, optimisation of computer program structure) and microscopy (cytology).

The *Clustered Vehicle Routing Problem (CluVRP)* is introduced by Sevaux et al. (2008) in order to model the parcel delivery activities of courier companies. A multi-objective approach for the vehicle routing problem for courier companies is presented by Janssens et al. (2015). A common practice in the supply chains of these companies is the sorting of parcels into bins, usually containing the orders that need to be delivered within a specific distribution *zone.* The routing problem is therefore solved initially at the level of the zones (clusters). Afterwards, inside each zone the clients are visited in milk runs. The idea of client clustering can also be used in problems where it is desirable that certain clients are served by the same vehicle. This can be the pooling of clients demanding a similar service, requesting a specific repairman skill, or when the customer-supplier relationship is perceived important by one of the parties.

The existing literature on the CluVRP is rather scarce. In Pop et al. (2012), an *exact method* for solving the CluVRP is formulated as an extension of the Generalized Vehicle Routing Problem (GVRP). The GVRP is closely related to the CluVRP, as both problems deal with the idea of clustered customers. However, in the GVRP, it is sufficient to serve only one customer in

every cluster (Ghiani and Improta, 2000). A new compact and effective integer programming formulation for the CluVRP is proposed by Battarra et al. (2014). This exact approach is used as a benchmark for the results discussed in this paper.

When it comes to finding a good *heuristic solution*, Barthélemy et al. (2010) introduced the idea of transforming the CluVRP into the CVRP. This is achieved by constructing an altered distance matrix, where a large distance M is added to all inter cluster edges. In this way the vehicle has a higher probability to serve first all the customers of a single cluster before leaving the cluster. In Barthélemy et al. (2010) this approach is combined with a Simulated Annealing metaheuristic. Battarra et al. (2013) combine an Iterated Local Search with a Memetic Algorithm framework.

To the best of our knowledge, all current heuristic contributions make use of the conversion towards the more classical CVRP to solve the CluVRP. In this paper, an alternative metaheuristic approach is proposed that is able to exploit the clustered structure of the problem in order to reduce the problem complexity. We show that the combination of a first variable neighbourhood search (VNS) at the global — cluster — level and a second VNS at individual client level, might lead to high quality solutions in limited calculation time.

The paper is structured as follows. In Section 2, the problem is formally described after which a detailed look is taken at the developed metaheuristic in Section 3. Our approach is tested on instances of different sizes. In Section 4.2, these instances and the obtained results are discussed. A CluVRP with soft cluster constraints is introduced and compared to the original hard cluster constraint variant in Section 5. In Section 6 the main conclusions are summarised.

2 Problem Definition

We are given a complete undirected graph $G = (V, E)$, where $V = \{0, 1, 2, ..., n\}$ is a set of nodes (Vertices), from which node 0 is the central depot and the labels 1 to n contain the n delivery nodes (customers). The distance $d_N(i, j)$ is associated with each Edge $(i, j) \in E$, connecting two nodes. Furthermore, we also define c clusters c_k with $k = \{0, 1, ..., c\}$, from which cluster 0 contains only one node, the depot. All other clusters contain at least one customer. In the distribution area, every cluster is represented by its *cluster center*. The cluster center is equal to the average euclidean coordinate of all customers inside the cluster. The distance between two clusters is then reduced to the euclidean distance between the two cluster centers, and denoted by $d_C(i, j)$. Furthermore, for every cluster, the total demand q_k, aggregated over all customers in the cluster, is given. A homogeneous fleet of t vehicles is assumed, each of them can ship

Q goods. All vehicles start and end their trip at the central depot and should visit at least one cluster. Table 1 summarises the different symbols used in this paper.

Table 1: *List of symbols*

n	number of distribution nodes
c	number of clusters
t	number of vehicles available
Q	maximum vehicle capacity
q_k	demand at cluster k
$d_N(i,j)$	distance between nodes i and j
$d_C(i,j)$	distance between clusters i and j
i,j,k	indices

For every vehicle in the CluVRP, a trip is to be defined in such a way that all customers are served without violating the capacity constraint of each vehicle. Furthermore, some extra (hard) cluster constraints are added to this classical VRP formulation. These state that clients that are clustered together should be visited by the same vehicle in the same path. In other words, when a customer is served, the vehicle should first serve all other customers belonging to the same cluster before leaving the cluster to either return to the depot or to visit another cluster. In our solution approach we aim at minimizing the total distance travelled. A visualisation of the CluVRP with hard cluster constraints can be seen in Figure 1. For a detailed Integer Programming formulation, we refer to Battarra et al. (2014).

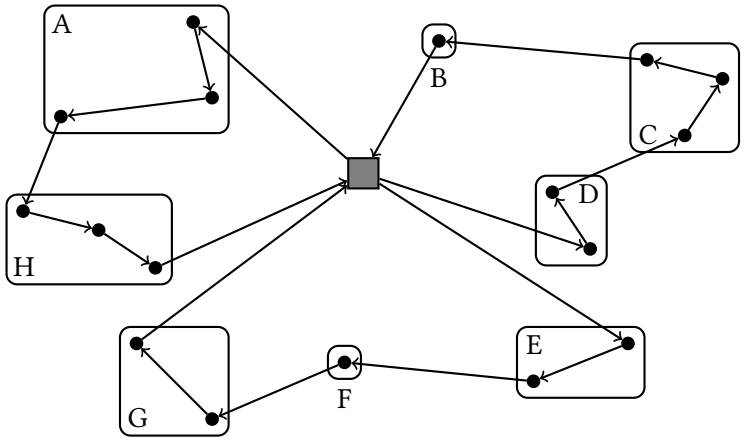

Figure 1: *Visualisation of the CluVRP with hard cluster constraints. All three vehicles depart from the central depot (gray rectangle) to serve all the customers (black bullets). All customers within the same cluster are served consecutively by the same vehicle.*

Stated by Lenstra and Kan (1981), the vehicle routing problem without clustering is NP-hard. Any VRP can be reduced to a CluVRP either with one customer in each cluster or with one cluster containing all customers. As the complexity of this reduction is linear with respect to

the number of customers, the CluVRP is also NP-hard (Barthélemy, 2012).

3 A metaheuristic approach for the CluVRP

In order to tackle the CluVRP, we propose a metaheuristic approach that explores the solution space at two different levels, denoted as *cluster level* — the higher, more global level — and *node level* — taking every single customer into account. At both cluster and node level, a variable neighbourhood search (VNS) is in charge of finding a local optimum. In order to facilitate the search process, the result obtained at the cluster level is used as an input for the node level VNS. During the diversification phase, a reverse move is made by going back from node level to cluster level. This is called a *zoom-out*. The overall framework of our metaheuristic is shown in Algorithm 1.

In the following sections, we take a closer look at the different operators and their implementation.

3.1 Construction phase

The objective of the construction phase is to provide the metaheuristic with an initial solution at *cluster level*. This means that for every cluster, the individual customers are disregarded and replaced by the cluster center. The different clusters, and their corresponding demands, should be allocated to the set of t vehicles. This problem can be identified as a *one-dimensional bin packing problem with given number of bins*. A set of items (clusters) with a given weight (demand) are to be packed into a set of bins (vehicles) with a predefined maximal load (vehicle capacity Q). Where in the original bin packing problem the objective usually consists of minimizing the number of bins used, here it is sufficient to obtain a feasible solution, in which every vehicle contains at least one cluster. In order to be able to compare our results with the literature, we impose that every vehicle should be used to visit at least one cluster.

The one-dimensional bin packing problem is shown to be strong NP-complete (Garey and Johnson, 1978). Because we are not interested in the optimal bin packing solution, but a solution for our CluVRP, we prefer a fast algorithm that provides us with a feasible result to start from. The *first-fit decreasing* and the *best-fit decreasing* algorithms are most commonly applied in the literature. In this paper, the best-fit decreasing strategy is adapted in order to comply with the our objective for the CluVRP.

Algorithm 1 Pseudo code of the two-level metaheuristic approach for solving the CluVRP.

1: **Step 0: Initialization**
2: S_c = solution at cluster level
3: S_i = solution at individual client level and $f(S_i)$ its objective value
4: S_i^* = best solution found so far and $f(S_i^*)$ its objective value
5: *mutationProb* = probability that the mutation operator is chosen instead of the perturbation operator
6: nNoImprovement = 0; iteration = 0; $f(S_i^*) \leftarrow \infty$
7: intra-cluster-sweep-heuristic()
8: **Step 1: Construction phase**
9: $S_c \leftarrow$ add-unvisited-clusters() *(=Bin packing approach)*
10: **while** not-all-clusters-allocated **do**
11: Redistribution-algorithm()
12: $S_c \leftarrow$ add-unvisited-clusters()
13: **end while**
14: **Step 2: Intensification phase**
15: $S_c' \leftarrow$ VNS-at-cluster-level(S_c)
16: $S_i \leftarrow$ conversion-operator(S_c')
17: $S_i' \leftarrow$ VNS-at-client-level(S_i)
18: **if** $f(S_i') < f(S_i^*)$ **then**
19: $S_i^* \leftarrow S_i'$; $f(S_i^*) \leftarrow f(S_i')$
20: nNoImprovement = 0
21: **else**
22: $nNoImprovement \leftarrow nNoImprovement + 1$
23: **if** $nNoImprovement = maxNoImprovement$ **then**
24: $iteration \leftarrow iteration + 1$
25: **if** $iteration = nRestart$ **then**
26: return S_i^*
27: stop the algorithm
28: **end if**
29: **end if**
30: **end if**
31: **Step 3: Diversification phase**
32: $r \leftarrow$ random number [0,1]
33: **if** $r < mutationProb$ **then**
34: $S_c \leftarrow$ mutation(S_c')
35: **go to** 16
36: **else**
37: $S_c \leftarrow$ perturbation(S_c')
38: $S_c \leftarrow$ add-unvisited-clusters()
39: **while** not-all-clusters-allocated **do**
40: Redistribution-algorithm()
41: $S_c \leftarrow$ add-unvisited-clusters()
42: **end while**
43: **go to** 14
44: **end if**

The traditional best-fit bin packing algorithm places each item (cluster), in succession, into the fullest bin (vehicle) in which it fits (Fleszar and Hindi, 2002). For a simple bin packing problem this is satisfying, but when solving the CluVRP, it is important that efficient routes can be constructed with the clusters that have been allocated to the same vehicle afterwards. Therefore, we have adapted the definition of *best fit*. While allocating the clusters to a specific vehicle, the developed algorithm keeps track of the *vehicles' center of gravity*, calculated as the average euclidean coordinate of all clusters already allocated to this vehicle. For every cluster, sorted in decreasing order according to their demand , the bin (vehicle) that has its center of gravity the closest to this cluster is preferred. In this way, the algorithm is more likely to combine different clusters that are located in the same part of the distribution area into one vehicle. Once a vehicle has departed from the depot in a certain direction, it is more likely that other clusters in that same direction are also allocated to this vehicle.

To ensure that a wide area of the solution space is explored by the algorithm, the idea of the GRASP heuristic (Greedy Randomized Adaptive Search Procedure) in included iteratively in the algorithm, as some randomness is introduced by not selecting the closest vehicle but the *nBest* closest vehicles from which one is chosen ad random. By doing so, new possible solutions are explored when the algorithm is executed multiple times in a row (multi-start). The value of the *nBest* parameter is defined based on the total number of vehicles available, *nVehicles*, for the instance according to Equation (1).

$$\text{nBest} = \lfloor \frac{\text{nVehicles}}{2} \rfloor + 1 \tag{1}$$

3.1.1 Redistribution algorithm

A fixed number of vehicles is imposed for every instance. However, as the bin packing algorithm is a heuristic procedure that will not guarantee the optimal solution and although some randomness is included into the construction procedure, a point can be reached where no vehicle has enough capacity left to store the next cluster.

In order to cope with these situations, a specific repair operator, that tries to optimize the current capacity distribution, is built in the solution algorithm. This is done by swapping the vehicle of two clusters in order to increase the load factor of one of the vehicles, preferably up to 100%. As a consequence, more capacity becomes available in the other vehicle that can now be assigned to the extra cluster(s). The working of the redistribution algorithm is also shown in Figure 2.

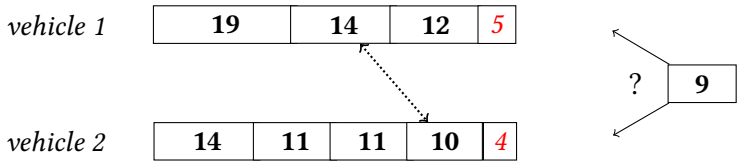

Figure 2: *Visualisation of an operation performed by the redistribution algorithm. By swapping the orders of 14 and 10 items, free space is created in vehicle 1 where the additional order of 9 can be placed.*

Two vehicles are considered with a capacity of 50 items each. Eight clusters, with demand equal to 19, 14, 14, 12, 11, 11, 10 and 9, should be allocated to one of the vehicles. At a certain moment during the construction phase, the situation might appear where vehicle 1 has a capacity of 5 left, and vehicle 2 has room for 4 more items, while the cluster with a demand of 9 still needs to be allocated. At this moment, the redistribution algorithm is called in order to optimise the load factor of one vehicle, resulting into a higher capacity availability in the other one. The preferred move is to swap the vehicle assigned to the clusters with a demand of 14 and 10, as this will result in a 100% utilisation of vehicle 2, creating enough spare capacity in vehicle 1 to put this last cluster.

3.2 Intensification phase

After the construction phase, the actual core of our metaheuristic — called the intensification phase — is responsible for improving the initial solution until a local optimum is reached. First, the cluster level solution is further improved at cluster level (by using a first VNS), obtaining a local optimal sequence of clusters for every vehicle. Afterwards, the cluster centers are replaced by the individual client nodes, resulting in a good initial solution at client level that is then used as the input for a second VNS for further improvement. The intensification phase can therefore be subdivided into *two intensification blocks* — one at cluster level, a second one at client level — with their own VNS. These two blocks are executed sequentially.

3.2.1 Intensification at cluster level

The first part of the intensification phase is executed at the level of the clusters. By ignoring all individual client nodes in this *global approach*, the problem size and complexity is reduced. This higher level routing problem, with the objective of finding an optimal cluster sequence for every vehicle, is solved by a variable neighbourhood search, based on seven local search operators commonly used in vehicle routing. Both *intra and inter vehicle neighbourhoods* are explored. The intra vehicle operators involve a single vehicle and try to minimize the total

distance of a single trip. In inter vehicle operators more than one vehicle is involved, with the aim of optimizing the global cost (total distance) of the solution by exchanging one or a set of clusters between different vehicles. All local search operators discussed are shown in Table 2 and have complexity $O(n^2)$.

Table 2: *List of all intra and inter vehicle local search operators, implemented in the variable neighbourhood search at cluster level*

Intra vehicle operators	
Swap	Swap the position of two clusters in a single trip.
Relocate	Remove one cluster and insert it at a different position in the vehicle.
Two-Opt	Remove two edges and replace them by two new edges to close the tour.
Or-Opt	Remove N sequential clusters and insert them at a different position in the vehicle. (with $N = \{2, 3, 4\}$)
Inter vehicle operators	
Swap	Swap the vehicle of two clusters.
Relocate	Remove one cluster and insert it in another vehicle.
Or-Opt	Remove N sequential clusters and insert them into another vehicle. (with $N = \{2, 3, 4\}$)

The different neighbourhoods are checked sequentially, by first exploring the intra vehicle operators, as they are less expensive in terms of calculating time, especially for the instances that involve a higher number of vehicles. From the moment that no improvement can be found in the current neighbourhood, the algorithm moves to the next neighbourhood. Every time an improvement is found, the algorithm returns to the first neighbourhood. This is repeated until none of the neighbourhoods is able to improve the current solution any further. We then consider that a local optimum is reached at cluster level.

3.2.2 Conversion operator

The optimal cluster sequence for every vehicle, obtained during the intensification phase at cluster level should be converted into a solution at client level before sending it to the client level intensification phase. This is done by the conversion operator, described in this section.

For every node, the angle linked to the polar coordinate representation, computed as the arctangent value of the node with respect to the location of the depot, is calculated during preprocession. By then sorting the nodes inside the same cluster according to this values, an open Hamiltonian path — used as an initial intra cluster path — is obtained for every cluster. This approach is called *the sweep heuristic*, introduced by Gillett and Miller (1974).

By pre-computing the intra cluster routes, the required time for the conversion operator is reduced. As the conversion operator is called in every iteration, this contributes to the overall

performance of the algorithm. We can remark that for small cluster sizes the pre-computation of the individual intra cluster routes can even be done by an exact approach.

During the conversion phase, the intra cluster routes are connected in the order given by the obtained solution at cluster level. The inter cluster connections are constructed based on a *nearest neighbour approach*. When moving to the next cluster, we go to the closest node, that is either the start or the ending of the Hamiltonian path obtained by the sweep heuristic. These connections might not be optimal. This, however, is taken care of during the intensification at client level.

3.2.3 Intensification at client level

The solution produced by the conversion operator, will be improved further in this second intensification phase, in which all individual client nodes are taken into account. Similar to the VNS discussed in 3.2.1, a set of neighbourhoods is explored in the search for a local optimal solution.

The cluster constraints, however, impose that all clients belonging to the same cluster remain grouped together. Therefore, the neighbourhood implementation should be done with care and infeasible moves should be avoided. Two main groups of neighbourhoods can be distinguished: the *intra cluster* and the *inter cluster* neighbourhoods. The first group is responsible for improving the Hamiltonian path within the different clusters. It should be noticed that the optimality of these intra cluster routes is also dependent on the cluster sequence, as this might affect the optimal edge to enter or leave the cluster. Secondly, the inter cluster neighbourhoods question again the cluster order as obtained by the variable neighbourhood search at cluster level. As no client information was taken into account at cluster level, a modified cluster sequence might be beneficial. The inter cluster operators can be both intra or inter vehicle, as the performed moves can involve a single vehicle (e.g. two entire clusters swap within the same vehicle), or multiple vehicles (e.g. two entire clusters in different vehicles are swapped). The neighbourhoods of this second VNS are described in Table 3.

Similar to the first VNS, the applied neighbourhoods are checked sequentially. From the moment that an improvement is found, the algorithm will start again by exploring the first neighbourhood. We continue until none of the neighbourhoods contains a better solution than the current one and therefore a local optimum is reached.

Table 3: *The different VNS Neighbourhoods at individual client level*

Intra cluster operators	
Swap	Swap the position of two clients within the same cluster in a single trip.
Relocate	Remove one client and insert it at a different position within the same cluster.
Two-Opt	Remove two edges and replace them by two new edges to close the tour.
Or-Opt	Remove N sequential clients and insert them at a different position within the same cluster in the vehicle. (with $N = \{2, 3, 4\}$)
Inter cluster operators *(intra vehicle)*	
Swap	Swap the position of two clusters within the same vehicle.
Relocate	Remove all clients of a single cluster and insert them sequentially at a different position in the same vehicle.
Inter cluster operators *(inter vehicle)*	
Swap	Swap the vehicle of two clusters.
Relocate	Remove all clients of a single cluster and insert them sequentially in another vehicle.

3.3 Diversification phase

After evaluating the obtained solution at client level, the algorithm has possession of three diversification strategies to continue the search and explore a wider part of the solution space. Out of the first two diversification operators — a *perturbation* and a *mutation* — one is selected ad random. If no improvement is found for a predefined number of iterations in a row, the third diversification operator — a *full restart* — is executed. In what follows, we take a closer look at all diversification strategies.

3.3.1 Perturbation

When the *perturbation* diversification operator is called, the algorithm performs a *zoom-out* by returning to the local optimal solution found at cluster level during this iteration. A random part of the solution, denoted by the *perturbationRate*, is destroyed, stacking the removed clusters in a separate list. A repair operator is called afterwards, that allocates again all removed clusters to a random vehicle. If no feasible vehicle can be found for a certain cluster, the redistribution operator (described in Section 3.1.1) is called.

From the moment that a new solution is obtained, the algorithm will continue the search at the intensification phase at cluster level (see Section 3.2.1).

3.3.2 Mutation

As both intensification phases are executed sequentially, the VNS at individual client level always uses an optimal cluster sequence as an input solution. However, it might be the case that for the global optimal solution, the corresponding client sequence is not locally optimal. The mutation operator helps in overcoming this problem.

Similar to the perturbation operator, the idea of *zoom-out* is used to go back to the last solution at cluster level. From this solution, two clusters are swapped randomly, turning the cluster level solution into a more inferior one.

Instead of starting the intensification phase now again cluster level, the conversion operator is called immediately. If not, the variable neighbourhood search at cluster level would reverse the mutation move and return again the previous local optimum. In this way, the solution will in the end be optimised locally at client level, starting from a non-optimal cluster sequence.

3.3.3 Restart

If, for a predefined number of consecutive perturbations or mutations, no better solution is found by the algorithm, the third diversification operator is called. The restart operator clears the complete solution and the algorithm is restarted again by constructing a new initial solution.

4 Parameter tuning and experimental results

4.1 Parameter tuning

The different parameters of our algorithm, listed in Table 4, are tuned on a selection of large-size instances (see Section 4.2 for a more elaborate presentation of the instances). By increasing the number of iterations that the complete search procedure is repeated, it can be expected that the solution quality will improve while calculation times will increase. We therefore assume the first two parameters (nNoImprovement and nRestart) to be a fixed value in order to focus on the impact of the other parameters on the solution quality, which is studied by means of a full factorial statistical experiment. For the perturbation rate, a value of 0.1 is obtained. The solution quality tends to be higher (lower GAP) for a small mutation probability, however, at the expense of increasing calculation times. As a compromise between calculation time and

solution quality, the value 0.2 is chosen for the mutation probability. This optimal parameter setting, denoted in the last column, is used to perform the experimental results, presented in the following sections.

Table 4: *Results of parameter tuning on a small subset of the large-size instances*

Parameter	Definition	Tested Values	#	Optimal
perturbationRate	Percentage of the solution that is randomly discarded by the perturbation diversification operator.	0.01, 0.05, 0.1, 0.2, …, 0.6	8	**0.1**
mutationProbability	Probability that the mutation operator is chosen instead of the perturbation operator	0.01, 0.05, 0.1, 0.2, 0.3	6	**0.2**

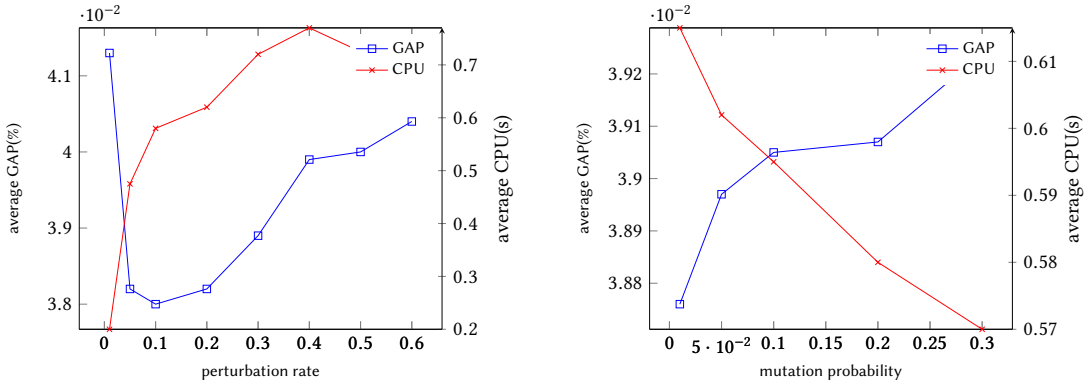

Figure 3: *Solution quality, measured by the average optimality gap and the average calculation time, for different parameter settings. Results are obtained by performing a full factorial experiment.*

4.2 Experimental results

The metaheuristic is tested extensively on benchmark instances of different sizes provided by Battarra et al. (2014) as discussed in their paper on *exact algorithms for the Clustered Vehicle Routing Problem*. The computational results are obtained by using an Intel(R) Core(TM) i5-3320M with 2.60GHz and 8GB of RAM, operating in Windows. Because some random components are implemented in the algorithm, the instances are solved multiple times (20 runs per instance).

First, the algorithm is tested on a set of 79 small and medium sized test instances, denoted as GVRPθ3. These GVRP instances, proposed by Bektas et al. (2011), are adaptations of existing CVRP instances[1]. The transformation is achieved by creating the clusters using a seed-based

[1]available at `http://branchandcut.org/vrp/data`

algorithm and by replacing the number of required vehicles by the solution (number of bins) of the bin packing problem for each instance. The resulting number of clusters can be up to 88, while the average number of customers per cluster is three.

Furthermore, we incorporated a set of 20 large-size instances, provided by Golden et al. (1998) with 240 to 483 customers and transformed by Battarra et al. (2014) into CluVRP instances by using the same transformation as described above for different values of θ. With an average of up to 15 nodes per cluster, these instances are more challenging for our algorithm.

The obtained results for the GVRPθ3 instances are summarised in Table 5. By using their Branch and Cut method, Battarra et al. (2014) are able to solve 78 out of 79 instances exactly within reasonable time limits. It should be mentioned that for these methods the preprocessing times, which are between 3 and 8 seconds, are not included in the calculation times (CPU). This preprocessing step consists of the calculation of all possible Hamiltonian Paths inside each cluster. Afterwards, while running the branch and cut approach, these results are used to define the optimal inter cluster connections at customer level for a given sequence of clusters.

Table 5: *Results for the GVRPθ3 instances. Comparison between the branch and cut and price (BCP), branch and cut (BC) (Battarra et al., 2014) and the variable neighbourhood approach (VNS) proposed in this paper.*

		BCP		**BC**		**VNS**			
		Opt.	CPU (s)	Opt.	CPU (s)	Opt.	CPU (s)	Best GAP	Avr. GAP
A	31 - 79 cust.	27/27	42.52	27/27	4.84	14/27	0.036	0.00%	0.44%
B	30 - 77 cust.	23/23	7.69	23/23	4.99	13/23	0.039	0.00%	0.37%
P	15 - 100 cust.	24/24	0.48	24/24	3.77	10/24	0.040	0.00%	0.46%
M+G	100 - 261 cust.	2/5	157.25	4/5	25.44	0/5	0.48	0.50%	1.67%
total		76/79		78/79		37/79			
average			26.87		5.86		0.066		0.49%

Our variable neighbourhood search is able to solve 37 instances to optimality, while reducing the calculation times significantly. Even if the optimal solution is not found by the algorithm, a high solution quality is guaranteed. For the instances of class A, B and P, the optimality gaps remain very small – below 1% of the optimal solution value. For the medium-sized instances in class M and G, with number of nodes ranging from 101 to 262 (G), the average optimality gap is only 1.67% with a maximum of 2.73%. For the instance (G-n262-k25-C88-V9.gvrp) that could not be solved by the exact approach of Battarra et al. (2014), a heuristic solution is obtained after less than a second.

For the large-size Golden instances, the preprocessing step of the exact approaches becomes more complex, resulting in calculation times that stay on average below one hour, but can peak

to 20,541 seconds (almost six hours). In order to improve the performance of the exact branch and cut algorithm that solves the CluVRP afterwards, Battarra et al. (2014) include a graph reduction and feed the solver also with an initial upper bound to increase the speed.

In Table 6, the results of this branch and cut algorithm (preprocessing excluded) are compared to our Variable Neighbourhood metaheuristic (VNS) according to the θ value of the instances, with θ representing the average number of customers per cluster. The reported calculation times are averaged.

Table 6: *Results for the Golden instances. Comparison between the branch and cut algorithm with graph reduction and initial solution as upper bound (Battarra et al., 2014) and the variable neighbourhood search proposed in this paper. Reported values are the averaged over all instances in the set.*

	BC (GR+UB)		**VNS**			
θ	Opt.	CPU (s)	Opt.	CPU (s)	best GAP	average GAP
5	17/20	363.00	0/20	0.96	2.73%	3.01%
6	19/20	86.65	0/20	0.92	2.80%	3.21%
7	19/20	109.71	0/20	0.72	2.61%	3.02%
8	19/20	93.86	0/20	0.65	2.61%	3.09%
9	19/20	92.18	0/20	0.56	2.91%	3.33%
10	20/20	48.82	0/20	0.66	2.69%	3.19%
11	20/20	72.23	0/20	0.69	2.53%	3.07%
12	20/20	48.71	0/20	0.61	2.49%	3.13%
13	20/20	40.20	0/20	0.55	2.60%	3.27%
14	20/20	39.18	0/20	0.62	2.81%	3.54%
15	20/20	23.09	0/20	0.71	3.11%	3.92%
total	213/220		0/220			
average		88.66		0.70	2.72%	3.25%

With their branch and cut approach, Battarra et al. (2014) are able to solve 213 out of 220 instances to optimality. The VNS is not able to find the optimal solution for any of the given large-size instances. However, good quality solutions are obtained in very small computation times with a optimality gap of 2.72%. All gaps found lie between 0.36% and 6.65% of the optimal value.

The results per instance, averaged over all different values of θ, are visualised in Figure 4. A linear relationship between the instance size — expressed as the number of customers — and total calculation time can be observed. We also expect larger remaining optimality gaps for increasing instance sizes. This relationship is less notable, probably due to the fact that the size of this gap also highly depends on the value of θ, as discussed before.

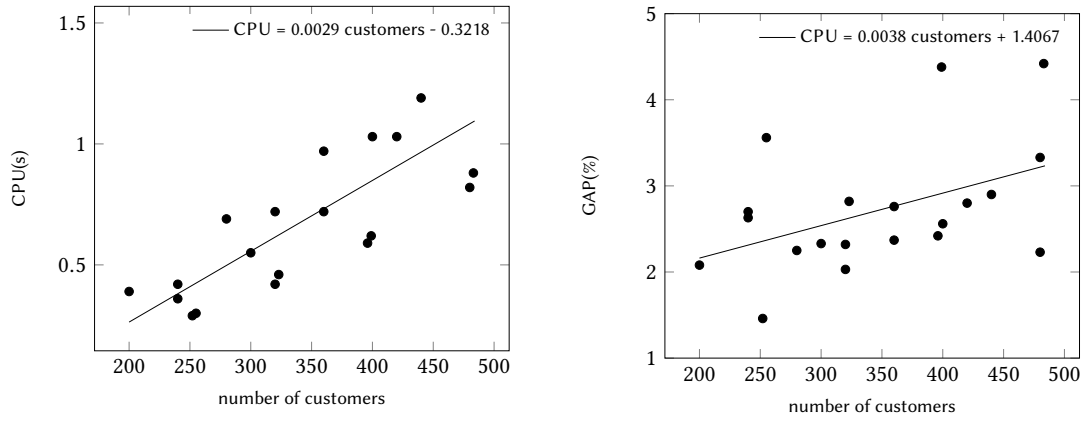

Figure 4: *Optimality gap and total computation time for the Golden instances as a function of the total number of customers. Results are averaged over all different values of θ.*

5 Soft cluster constraints

In the previous sections, we imposed for every instance that all clients inside a single cluster should be served by the same vehicle, before this vehicle either continues its trip to the next cluster or returns to the central depot. We will refer to this scenario as the *CluVRP with hard cluster constraints*. In many applications, however, it might not be necessary to stay within the cluster until all clients are served. It might be profitable to leave the cluster to serve first another cluster (partially) and return to the initial cluster afterwards to visit the remaining clients. In parcel delivery, e.g., the zones are used to allocate the parcels to a vehicle while it is not necessary that each driver distributes each zone individually.

In this section, the GVRPθ3 and Golden instances are solved as a *CluVRP with soft cluster constraints*. A vehicle is now allowed to leave and re-enter a cluster multiple times. We impose however that, in the end, all clients belonging to the same cluster should be served by the same vehicle. In order to solve the CluVRP with soft cluster constraints, small changes are made to our algorithm in order to expand the neighbourhoods of the VNS at individual client level.

The obtained results are compared to the ones discussed in Section 4.2. As seen in Table 7, the objective value (total distance travelled) is improved for all instances when soft cluster constraints are assumed. This is denoted by a negative relative difference in objective value, compared to the scenario with hard cluster constraints. As our algorithm contains some random elements, every instance is solved 20 times. The results in Table 7 are obtained by taking either the best or the average solution for every instance. The reported values are then averaged for every set of instances.

It can be noticed that larger improvements can be realized if the instance size is larger and when the number of clients per cluster is higher. This is not surprising. By adding more clients to the distribution area, the recombination possibilities increase and it becomes more likely that clients are located closer to each other. Also, if clusters are containing more clients, denoted by θ, the number of possible inter cluster connections increases which might result in a lower total distance travelled.

Due to an expansion of the VNS neighbourhoods at client level — *intra* cluster operators become *inter* cluster operators —, an increase in calculation time is observed. However, with average calculation times around 2 to 3 seconds, the algorithm still performs well.

Table 7: *Results for the GVRPθ3 and Golden instances with soft cluster constraints. Instances are solved by using the VNS algorithm. Reported values are the averaged over all instances in the set, compared to the obtained results with hard cluster constraints.*

GVRPθ3				**Golden instances**			
instance set	CPU (s)	Best difference	Avr. difference	θ	CPU (s)	Best difference	Avr. difference
A	0.05	-2.73%	-2.27%	5	2.98	-3.77%	-3.02%
B	0.06	-1.46%	-1.11%	6	3.04	-4.03%	-3.08%
P	0.06	-4.89%	-4.66%	7	2.94	-4.05%	-3.22%
M+G	0.57	-3.83%	-2.46%	8	2.97	-4.50%	-3.58%
				9	2.88	-4.65%	-3.76%
				10	3.64	-4.94%	-4.06%
				11	3.61	-5.09%	-4.30%
				12	3.16	-5.75%	-4.75%
				13	2.56	-5.91%	-4.89%
				14	2.70	-6.36%	-5.28%
				15	2.72	-6.33%	-5.13%
average	0.09	-3.09%	-2.67%		3.02	-5.03%	-4.10%

6 Conclusion

In this paper, the Clustered Vehicle Routing Problem is tackled by means of a fast and high performing metaheuristic. The heuristic approaches proposed in the literature only make use of a transformation of the CluVRP towards a classical CVRP. The multi-level VNS, developed by the authors, however, solves the CluVRP as a distinct problem. Due to this, the specific clustered structure of the problem can be used to reduce the complexity of the full problem and by integrating a global (clusters) and local (clients) approach, high quality solutions are obtained.

The algorithm is tested on benchmark instances from the literature with different sizes and

diverse complexities. Many of the small and medium-sized problems are solved to optimality in a very short time. Even if the optimal solution was not found, an optimality gap below 1% was obtained. For the large-sized problems, no optimal solutions were found. However, with the obtained optimality gaps around 2 to 3% and very low computation times, high quality and competitive solutions are obtained by our metaheuristic approach.

Furthermore, we distinguished two types of CluVRPs. The traditional CluVRP with hard cluster constraints in which it is not allowed to leave a cluster before having served all clients within the cluster, and a variant with soft cluster constraints. In the second problem, multiple inter cluster connections are allowed, as long as all clients belonging to the same cluster are served by the same vehicle. For the tested instances, the total distance travelled decreases with 1.46% to 6.36% when going from hard to soft cluster constraints.

References

T. Barthélemy. A bi-objective Inventory Routing Problem with Periodic and Clustered deliveries. Master's thesis, Universit de Nantes, France, 2012.

T. Barthélemy, A. Rossi, M. Sevaux, K. Sörensen, et al. Metaheuristic approach for the clustered vrp. In *EU/MEeting: 10th Anniversary of the Metaheuristics Community - Université de Bretagne Sud, France*, 2010.

M. Battarra, G. Erdoğan, A. Subramanian, and T. Vidal. Metaheuristics for the clustered vehicle routing problem. In *EURO INFORMS 26th European conference on operational research - Rome - proceedings*, 2013.

M. Battarra, G. Erdogan, and D. Vigo. Exact algorithms for the clustered vehicle routing problem. *Operations Research*, 2014.

T. Bektas, G. Erdogan, and S. Røpke. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, 45(3):299–316, 2011.

J. Chisman. The Clustered Traveling Salesman Problem . *Computers & Operations Research*, 2 (2):115 – 119, 1975.

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

C. Ding, Y. Cheng, and M. He. Two-level genetic algorithm for clustered traveling salesman problem with application in large-scale tsps. *Tsinghua Science & Technology*, 12(4):459 – 465, 2007.

K. Fleszar and K. S. Hindi. New heuristics for one-dimensional bin-packing. *Computers & operations research*, 29(7):821–839, 2002.

M. Garey and D. Johnson. "strong " np-completeness results: Motivation, examples, and implications. *Journal of the ACM*, 25(3):499–508, July 1978.

G. Ghiani and G. Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, 122(1):11–17, 2000.

B. Gillett and L. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2):340–349, 1974.

B. Golden, E. Wasil, J. Kelly, and I.-M. Chao. *The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results*. Centre for Research on Transportation. Springer US, 1998. ISBN 978-1-4613-7637-8.

J. Janssens, J. V. den Bergh, K. Srensen, and D. Cattrysse. Multi-objective microzone-based vehicle routing for courier companies: From tactical to operational planning. *European Journal of Operational Research*, 242(1):222 – 231, 2015.

G. Laporte, J. Potvin, and F. Quilleret. A tabu search heuristic using genetic diversification for the clustered traveling salesman problem. *Journal of Heuristics*, 2(3):187–200, 1997. ISSN 1381-1231.

G. Laporte, U. Palekar, and Correspondence. Some applications of the clustered travelling salesman problem. *Journal of the Operational Research Society*, 53(9):972–976, 2002.

J. K. Lenstra and A. H. G. R. Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981. ISSN 1097-0037.

P. Pop, I. Kara, and A. Marc. New mathematical models of the generalized vehicle routing problem and extensions. *Applied Mathematical Modelling*, 36(1):97–107, 2012.

J. Potvin and F. Guertin. The clustered traveling salesman problem: A genetic approach. In *Meta-Heuristics*, pages 619–631. Springer, 1996.

M. Sevaux, K. Sörensen, et al. Hamiltonian paths in large clustered routing problems. In *Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing, EU/ME'08*, pages 411–417, 2008.

Acknowledgements

This research is supported by the Research Foundation - Flanders (FWO - Ph.D. fellowship) and the Interuniversity Attraction Poles (IAP) Programme initiated by the Belgian Science Policy Office (COMEX project).

Furthermore, the authors would like to thank Maria Battarra (University of Southampton) for sharing the test instances and optimal results.